

**MULTI-COMMODITY DYNAMIC FLOW PROBLEMS
WITH INTERMEDIATE STORAGE AND VARYING
TRANSIT TIMES**



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
DURGA PRASAD KHANAL

September, 2023

**MULTI-COMMODITY DYNAMIC FLOW PROBLEMS
WITH INTERMEDIATE STORAGE AND VARYING
TRANSIT TIMES**



**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
DURGA PRASAD KHANAL**

September, 2023



TRIBHUVAN UNIVERSITY
Institute of Science and Technology

DEAN'S OFFICE

Kirtipur, Kathmandu, Nepal

Institute of Science & Technology
Dean's Office
Kirtipur 2045

EXTERNAL EXAMINERS

Reference No.:

The Title of Ph.D. Thesis: "**Multi-Commodity Dynamic Flow Problems With Intermediate Storage and Varying Transit Times**"

Name of Candidate: **Durga Prasad Khanal**

External Examiners:

- (1) Prof. Dr. Chinta Mani Pokharel
Institute of Engineering
Tribhuvan University, NEPAL

- (2) Prof. Dr. Jayanta Kumar Dash
Department of Mathematics
Institute of Management and Information Technology
Biju Patnaik University of Technology
INDIA

- (3) Prof. Dr. Sergio R. Canoy Jr.
Department of Mathematics & Statistics,
Mindanao State University,
IIT, PHILIPPENES



February 21, 2024

Dr. Surendra Kumar Gautam
Asst. Dean

Declaration

This thesis entitled “Multi-commodity Dynamic Flow Problems with Intermediate Storage and Varying Transit Times” 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. Urmila Pyakurel, Central Department of Mathematics, Tribhuvan University and co-supervised by Prof. Dr. Tanka Nath Dhamala, Central Department of Mathematics, Tribhuvan University and 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.

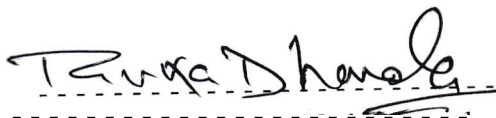


Durga Prasad Khanal
September, 2023

Recommendation

This is to recommend that Mr. Durga Prasad Khanal has carried out research entitled “Multi-commodity Dynamic Flow Problems with Intermediate Storage and Varying Transit Times” 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 University, Kirtipur for the submission of the thesis for the award of Ph.D. degree.

Professor Dr. Urmila Pyakurel
Supervisor
Central Department of Mathematics
Tribhuvan Unviersity
Kirtipur, Kathmandu, Nepal



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



Professor Dr. Stephan Dempe
Co-Supervisor
Fakultät für Mathematik und Informatik
Technische Universität Bergakademie Freiberg
09599, Freiberg, Germany

September, 2023



TRIBHUVAN UNIVERSITY

CENTRAL DEPARTMENT OF MATHEMATICS



KIRTIPUR, KATHMANDU
NEPAL

Date: Feb. 21, 2024

Ref

Letter of Approval

On the recommendation of **Prof. Dr. Urmila Pyakurel, Prof. Dr. Tanka Nath Dhamala and Prof. Dr. Stephan Dempe**, this Ph.D. thesis submitted by **Mr. Durga Prasad Khanal**, entitled "**Multi-commodity Dynamic Flow Problems with Intermediate Storage and Varying Transit Times**" is forwarded by Central Department Research Committee (CDRC) to the Dean, IOST, T.U..

Dr. Chet Raj Bhatta

Professor,

Head,

Central Department of Mathematics

Tribhuvan University

Kirtipur, Kathmandu

Nepal

Acknowledgements

I would like to offer my special thanks to the late supervisor Prof. Dr. Urmila Pyakurel for manifesting my interest in the field of mathematical optimization, invaluable guidance and support until she drew her last breath even though she was struggling with health problem for last seven years and the draft of thesis was send to her just a week before her last breath. I am grateful to my co-supervisor Prof. Dr. Tanka Nath Dhamala for motivating me to be eagerness on tackling different aspects of the problems. Similarly, I am grateful to Prof. Dr. Stephan Dempe, who agreed to be a co-supervisor, provide homely environment for two years of research stay at Technische Universität Bergakademie Freiberg and encouraged me to analyze the problem instances with different solution strategies. This work would not have been possible without their continuous encouragement, inspiration and guidance.

I am thankful to German Academic Exchange Service (DAAD) for awarding me with Research Grants - Bi-nationally Supervised Doctoral Degrees/Cotutelle, 2021/22. This support is not only fruitful to enhance my research but also to know the academic, social and cultural aspects of Germany. I would also like to thank Prof. Dr. Ingo Schiermeyer, Professor of Applied Discrete Mathematics, for facilitating the administrative activities after the retirement of Prof. Dr. Stephan Dempe. Similarly, I am thankful to Mrs. Caterina Löschner, secretary at Institut für Numerische Mathematik und Optimierung, TU Bergakademie Freiberg, for making my research stay very comfortable via administrative supports.

I am also thankful to University Grants Commission Nepal for providing me a Ph.D. Fellowship, Nepal Mathematical Society for NMS fellowship, Saraswati Multiple Campus, Thamel and Tribhuvan University for granting me a study leave. Similarly, I am thankful to the Central Department of Mathematics, research committee and administrative staffs for providing a friendly academic environment.

I am hearty thankful to my family members, specially my wife Nirmala Bhattarai, relatives and well-wishers for their continuous support and encouragement. Last but not least, I would like to thank my friend Mr. Dipak Babu Amgain for the technical support to carryout research implementations in python program.

Durga Prasad Khanal
September, 2023

शोध सार

सामान्यतया कमोडिटीहरू (एकल वा बहु-कमोडिटी) लाई स्रोतबाट गन्तव्यमा पठाउनको लागि प्रयोग गरिने सञ्जाल (Network) संग सम्बन्धित प्रवाह समस्याहरू लाई सञ्जाल प्रवाह भनिन्छ। एकल कमोडिटी प्रवाह समस्यामा कमोडिटीहरूलाई समान मानिन्छ र एक स्रोतबाट गन्तव्यमा पठाइन्छ (बहु स्रोत-गन्तव्यको अवस्थामा यसलाई भर्चुअल स्रोत र गन्तव्य प्रदान गरेर एकल स्रोत-गन्तव्यमा घटाउन सकिन्छ) जबकि बहु-कमोडिटी प्रवाह समस्यामा विभिन्न कमोडिटीहरू सम्बन्धित स्रोतहरूबाट सम्बन्धित गन्तव्यहरूमा पठाइन्छ। त्यसै गरी, मध्यवर्ती भण्डारणसहितको प्रवाह एक सञ्जाल प्रवाह समस्या हो जसमा स्रोतबाट प्रवाहित कमोडिटीहरू गन्तव्यमा मात्र नभई उपयुक्त मध्यवर्ती आश्रयहरूमा पनि पठाइन्छ ताकि स्रोतबाट कुल प्रवाह अधिकतम होस। अर्कोतर्फ, कन्ट्राप्ले दुई-तर्फी सञ्जालमा प्रवाह वृद्धिको लागि प्रयोग हुने विधि हो जसमा विपरीत दिशाका समानान्तर अर्कहरू गन्तव्यतर्फ उल्टाइन्छ।

बहु-कमोडिटी प्रवाह (Multi-commodity flow) मा मध्यवर्ती भण्डारणको साथ प्रवाहको विस्तारको रूपमा हामी पोलिनोमीयल (Polynomial) समयमा अधिकतम स्थिर (Maximum static) बहु-कमोडिटी प्रवाह समस्या र सुडो-पोलिनोमीयल (Pseudo-polynomial) समयमा अधिकतम गतिशील (Maximum dynamic) बहु-कमोडिटी प्रवाह समस्या समाधान गर्छौं। पोलिनोमीयल समय अनुमानको लागि हामी प्राथमिकतामा आधारित अधिकतम गतिशील बहु-कमोडिटी प्रवाह प्रस्तुत गर्दछौं जुन विपद् व्यवस्थापनमा उपयोगी हुन सक्छ। त्यसैगरी, हामी पोलिनोमीयल समयमा समानुपातिक क्षमता साझेदारी प्रविधि र सुडो-पोलिनोमीयल समयमा प्रवाह-निर्भर क्षमता साझेदारी विधि प्रयोग गरेर बन्डल (साझा) अर्कहरूमा क्षमता साझेदारी गरेर अधिकतम (Maximum) र द्रुत (Quickest) बहु-कमोडिटी प्रवाह समस्याहरूको समाधान प्रदान गर्दछौं। हामी Length bound र Delta-condensed विधिहरू प्रयोग गरेर अंशिक कन्ट्राप्लेको साथ प्रवाह-निर्भर द्रुत बहु-कमोडिटी प्रवाह समस्याको पोलिनोमीयल समयमा समाधान गर्छौं।

सञ्जाल प्रवाह मोडेलहरूको विभिन्न अनुप्रयोगहरू बाहेक हाम्रो मुख्य लक्ष्य भनेको हाम्रा समस्याहरूलाई विपद् व्यवस्थापन परिदृश्यहरूसँग सम्बन्धित गर्नु हो। त्यसकारण हामी स्रोत/हरूलाई खतरा क्षेत्र/हरू, गन्तव्य/हरूलाई सुरक्षित क्षेत्र/हरू र मध्यवर्ती आश्रयहरूलाई स्रोत/हरू भन्दा तुलनात्मक रूपमा सुरक्षित मान्दछौं। एकल कमोडिटी प्रवाह समस्या बहु-कमोडिटी प्रवाह समस्या को एक विशेष परिस्थिति हो। हामी एकल कमोडिटी अधिकतम गतिशील प्रवाह (Maximum Dynamic Flow (MDF)) र प्रारम्भिक आगमन प्रवाह (Earliest arrival flow (EAF)) समस्याहरू क्रमशः सामान्य सञ्जाल र श्रृंखला-समानान्तर (Series-parallel) सञ्जालमा मध्यवर्ती भण्डारणको साथ पोलिनोमीयल समयमा प्रवाहको पुनरावृत्ति (Temporally repeated) प्रयोग गर्दै समाधान गर्छौं। त्यसैगरी, विपरीत-समानान्तर अर्कहरूमा सममित (Asymmetric) ट्रान्जिट समयहरूसँग कन्ट्राप्ले समस्या समाधान गर्न हामी विपरीत-समानान्तर पथ गठन (Anti-parallel path decomposition) प्रविधिको विकास गर्छौं। मध्यवर्ती भण्डारण सहितको अस्थायी पुनरावृत्ति सममित र कन्ट्राप्ले सञ्जालमा विपरीत-समानान्तर पथ गठन विधि प्रयोग गर्दै MDF को मामला अध्ययन दृष्टान्त (Case illustration) को लागि हाम्रो समाधान रणनीतिलाई काठमाडौंको सडक सञ्जालमा लागू गर्छौं। शोध प्रबन्धको क्रमिक विकासको लागि, हामी एकल कमोडिटी प्रवाह समस्याबाट सुरु गर्छौं र बहु-कमोडिटी मामलामा फर्कन्छौं।

नोड-अर्क रूपमा नभई पथहरूले स्विक गर्ने गुण सहित एलिमेन्ट-पथ रूपमा कमोडिटी प्रवाह लाई अमूर्त (Abstract) सञ्जाल प्रवाह भनिन्छ। हामी अमूर्त (Abstract) सञ्जालमा मध्यवर्ती भण्डारणको साथ प्रवाहलाई पोलिनोमीयल समयमा स्थिर (Static), लेक्सिकोग्राफिक स्थिर (Lexicographic static) र गतिशील (Dynamic) प्रवाह समस्याहरू समाधान गर्छौं। यसले गैर-क्रसिड पक्षहरूमा पथहरूलाई मोडै र मध्यवर्ती आश्रयहरू (एलिमेन्टहरू) मा अतिरिक्त प्रवाह भण्डारण गरेर भीड हटाउन मद्दत गर्दछ। अमूर्त सञ्जालमा प्रवाह सुधार गर्न हामी अंशिक स्विचिंग (Partial switching) विधि प्रस्ताव गर्दछौं र पोलिनोमीयल समयमा अधिकतम र द्रुत प्रवाह समस्याहरू समाधान गर्दछौं।

सुविधा बाँडफाँड (Facility allocation) समस्या सञ्जाल प्रवाह समस्याको अर्को महत्त्वपूर्ण क्षेत्र हो जसको उद्देश्य उपयुक्त स्थानहरूमा सुविधाहरूको प्रतिस्थापनसँगै प्रवाह प्रसारणलाई अधिकतम बनाउनु हो। हामी समस्याको द्वि-स्तरीय सूत्र दिन्छौं जसमा माथिल्लो तहले सुविधाको स्थानको लागि उपयुक्त स्थान खोज्छ र तल्लो तहले अधिकतम प्रवाह समस्याको इष्टतम समाधान फेला पार्छ। Big-M र epsilon-Bound विधिको साथ Karush-Kuhn-Tucker (KKT) रूपान्तरण समस्या समाधान गर्न प्रयोग गरिने समाधान उपायहरू हुन्।

Abstract

Network flow problems, with single or multiple commodity, are commonly used to transship the objects from the source to the destination. In single commodity flow problem, objects are considered to be uniform and are sent from a source to a sink (in case of multiple source-sink, it can be reduced to single source-sink by assigning virtual source and sink) whereas in multi-commodity flow problem, different commodities are transshipped from respective sources to corresponding sinks. Similarly, flow with intermediate storage is a network flow problem in which flow from the source is not only sent to the sink but also at appropriate intermediate shelters so that total flow out from the source is maximized. On the other hand, contraflow is very well known and commonly used technique of flow increment in two-way network topology in which oppositely directed anti-parallel arcs are reversed towards the destination.

As an extension of the flow with intermediate storage in multi-commodity flow (MCF), we solve the maximum static MCF problem in polynomial time and maximum dynamic MCF problem in pseudo-polynomial time. For the polynomial time approximation, we present priority based maximum dynamic MCF which can be useful in disaster management. Similarly, we provide the approximate solutions to maximum and quickest MCF problems by sharing the capacity in bundle (common) arcs using proportional capacity sharing technique in polynomial time and flow-dependent capacity sharing technique in pseudo-polynomial time. We also discuss the polynomial time approximations of inflow-dependent quickest MCF problem with partial contraflow configuration using length bound and Δ -condense approaches.

Besides the different applications of network flow models, our main concern is to relate our problems to the evacuation scenarios. So, we consider source/s as the danger zone/s, sink/s as the safe zone/s and intermediate shelters comparatively safer than the source/s. As single commodity flow problem is a special case of multi-commodity flow problem, we solve the single commodity maximum dynamic flow (MDF) and earliest arrival flow (EAF) problems with intermediate storage in general network and series-parallel network, respectively, by using temporal repetition of the flow in polynomial time complexity. Similarly, to solve the contraflow problem with asymmetric transit times in anti-parallel arcs, we introduce anti-parallel path decomposition technique. For the implementation of temporally repeated solution to MDF with intermediate storage and anti-parallel path decomposition to asymmetric contraflow network,

we apply our solution strategies to the real road network of Kathmandu, Nepal as the case illustrations. For the sequential development of the thesis, we start with single commodity flow problem and turn to the multiple commodity case.

Abstract network flow concerns with shifting of the flow not in node-arc form but in element-path form in which paths must satisfy the switching property. We incorporate the flow with intermediate storage in abstract network and solve the static, lexicographic static and dynamic flow problems in polynomial time complexity. It helps to eliminate the congestion by diverting the flow in non-crossing sides and storing the excess flow at intermediate shelters (elements). To improve the flow in abstract network, we propose the partial switching technique and solve maximum and quickest flow problems in polynomial time.

The facility allocation problem is another important area of network flow problem whose objective is to maximize the flow transmission along with placement of the facilities at appropriate locations. We give the bi-level formulation of the problem in which upper level problem searches an appropriate location for the placement of the facility and lower level problem finds the optimal solution of maximum flow problem. A naive approach and Karush-Kuhn-Tucker (KKT) transformation with big- M constant and ϵ bound method are solution approaches used to solve the problem.

Contents

1	Introduction	1
1.1	Literature Review	2
1.2	Rationale of the Study	7
1.3	Objectives	8
1.4	Structure of the Thesis	8
2	General Network Flow	9
2.1	Notations and Terminologies	10
2.2	General Network Flow Models	12
2.2.1	Static Flow Model	12
2.2.2	Dynamic Flow Model	13
2.3	Flow with Intermediate Storage	15
2.3.1	Maximum Static Flow with Intermediate Storage	16
2.3.2	Maximum Dynamic Flow with Intermediate Storage	18
2.3.3	Temporally Repeated MDF with Intermediate Storage	19
2.3.4	Temporally Repeated EAF with Intermediate Storage	28
2.4	Contraflow	31
2.4.1	Contraflow with Symmetric Transit Times	33
2.4.2	Asymmetric Contraflow with Orientation-dependent Transit Times	35
2.4.3	Asymmetric Contraflow with Anti-parallel Path Decomposition	36
3	Abstract Network Flow	49
3.1	Abstract Network Flow with Intermediate Storage	49
3.1.1	Notations	50
3.1.2	Abstract Maximum Static Flow with Intermediate storage	51
3.1.3	Lexicographic Abstract Maximum Static Flow with Intermediate Storage	57
3.1.4	Abstract Maximum Dynamic Flow with Intermediate Storage	59
3.1.5	Abstract TRF with Intermediate Storage	65
3.2	Abstract Network Flow with Partial Switching	66
3.2.1	Abstract Static Flow with Partial Switching	66

3.2.2	Abstract Dynamic Flow with Partial Switching	68
3.2.3	Abstract Quickest Flow with Partial Switching	71
4	Multi-commodity Network Flow	73
4.1	Maximum Multi-commodity Flow	74
4.1.1	Commodity Prioritized Maximum Multi-commodity Flow	76
4.1.2	Maximum MCF with Proportional and Flow-dependent Capacity Sharing	83
4.1.3	Maximum Multi-commodity Flow with Intermediate Storage	87
4.2	Quickest Multi-commodity Flow	96
4.2.1	Quickest MCF with Proportional and Flow-dependent Capacity Sharing	96
4.2.2	Inflow-dependent Quickest MCF with Partial Contraflow	103
5	Bi-level Facility Allocation Problem	115
5.1	Basic Notations	116
5.2	Bi-level Problem Formulation with Facility Allocation	116
5.3	Solution Procedure	119
5.3.1	Solution by Naive Approach	119
5.3.2	Solution by KKT Transformation	120
6	Summary and Conclusions	122
A	Data for Case Illustration I	131
B	Data for Case Illustration II	133
C	List of Publications	135
D	List of Presentations	137

List of Abbreviations

MSF	Maximum static flow
MDF	Maximum dynamic flow
QF	Quickest flow
EAF	Earliest arrival flow
MSCF	Maximum static contraflow
MDCF	Maximum dynamic contraflow
QCF	Quickest contraflow
EACF	Earliest arrival contraflow
MCF	Multi-commodity flow
TRF	Temporally repeated flow

List of Symbols

\emptyset	Empty set
\mathbb{R}	Set of real numbers
\mathbb{R}_0^+	Set of positive real numbers including zero
\mathbb{Z}	Set of integers
\mathbb{Z}_0^+	Set of positive integers including zero
Π	Network
Π^T	Time expanded network of Π
Π^*	Modified network of Π
Π_i^*	Modified network of Π with respect to commodity i
$\bar{\Pi}$	Auxiliary network of Π
$\bar{\Pi}^b$	Bow graph of network of $\bar{\Pi}$
Π'	Extended network of Π with anti-parallel path decomposition
Π^r	Residual network of Π
\mathcal{N}	Set of nodes or vertices ($u \in \mathcal{N}$)
\mathcal{N}^T	Set of nodes in Π^T
\mathcal{A}	Set of arcs ($a = (u, v) \in \mathcal{A}$)
$\bar{\mathcal{A}}$	Set of arcs in auxiliary network
$\bar{\mathcal{A}}^b$	Set of bow arcs in $\bar{\mathcal{A}}$
\mathcal{A}_M	Set of movement arcs in time expanded network Π^T
\mathcal{A}_H	Set of holdover arcs in time expanded network Π^T
\mathcal{A}^T	Set of all arcs in Π^T , i.e., $\mathcal{A}_M \cup \mathcal{A}_H$
\mathcal{A}'	Set of arcs in extended network with anti-parallel path decomposition
κ	Arc capacity function
ν	Node capacity function
τ	Transit time function
$\hat{\tau}$	Maximum of arc transit times in the network
τ_0	Transit time of free flow
T	Time horizon
\mathcal{T}	Set of time ($\theta \in \mathcal{T} = \{0, \dots, T\}$ or $[0, T + 1)$)
s	Source or origin node

t	Sink or destination node
\mathcal{I}	Set of intermediate nodes, i.e., $\mathcal{N} \setminus \{s, t\}$
S	Set of multiple sources
D	Set of multiple sinks
Γ_u^{out}	Set of outgoing arcs from node u
Γ_u^{in}	Set of incoming arcs to node u
$tail(a)$	Tail node of arc a
$head(a)$	Head node of arc a
ϕ	Static flow function
$\hat{\phi}$	Static excess flow function
ϕ^i	Static flow function with respect to commodity i
ϕ_P	Static flow along $s - t$ path P
$\phi_{P_{[s \rightarrow u]}}$	Static flow along intermediate path $P_{[s \rightarrow u]}$
ϕ^w	Weakly inflow preserving flow
ϕ_0	Free flow function
$ \phi $	Value of static flow
Φ	Dynamic flow function
$\tilde{\Phi}$	Continuous time dynamic flow function
c	Cost function
c'	Maximum of arc costs, i.e., $\max\{c_a : a \in \mathcal{A}\}$
U	Maximum of arc capacities. i.e., $\max\{\kappa_a : a \in \mathcal{A}\}$
π	Node potential
$c_a(\pi)$	Reduced cost on arc a with node potential π
\mathcal{X}	Set of minimum cut arcs
\mathcal{P}	Set of directed paths (chains)
\mathcal{P}^c	Set of circulation paths in route based evacuation
\mathcal{P}_a	Set of paths through arc a
\mathcal{C}	Set of cycles
M	Set of prioritized nodes
λ	Static excess flow value
γ	Binary variable $\{0, 1\}$
K	Set of multiple (h) commodities $K = \{1, \dots, h\}$
d_i	Demand/supply of each commodity $i \in K$
B^i	Budget for commodity $i \in K$
\mathbb{P}	Priority ordering function
u^*	Dummy node of u
a^*	Dummy arc (u, u^*)
d	Shortest distance
L	Set of feasible locations

η	Size of the facility
Θ, ζ	Dual variables
w_a	Reward function of upper level objective
ϕ^*	Static flow after placement of facility
\mathcal{L}	Lagrangian function

List of Figures

2.1	Directed network with capacity and transit time.	9
2.2	Time expanded network of Figure 2.1 with $T = 5$	11
2.3	Given network with arc capacity, transit time and node capacity.	22
2.4	Network with excess flow λ at each node.	22
2.5	Updated network after balancing the flow on path \bar{P}_1	23
2.6	Updated network after balancing the flow on path \bar{P}_2	23
2.7	Updated network after balancing the flow on path \bar{P}_3	23
2.8	Updated network after balancing the flow on path \bar{P}_4	24
2.9	Evacuation zone with 69 nodes. Nodes 0 (red) and 68 (blue) represent the source and sink, respectively, whereas green areas are intermediate shelters. (Source: https://www.greattibettour.com/nepal-tours/maps-of-kathmandu.html)	27
2.10	A general non-series-parallel network and its excess flow computation at nodes.	29
2.11	A series-parallel network and its excess flow computation at nodes.	30
2.12	Flow balancing paths of series-parallel network in Figure 2.11.	31
2.13	Two-way networks with (i) symmetric and (ii) asymmetric transit times.	32
2.14	Auxiliary network of Figure 2.13(i).	33
2.15	Auxiliary network of Figure 2.13(ii).	35
2.16	Anti-parallel path decomposition (b) of given arc (a) with capacity and transit time.	36
2.17	Contraflow with anti-parallel path decomposition.	39
2.18	(b) represents the $t - s - t$ path of (a) as one-way network flow.	46
2.19	Evacuation zone with 50 nodes. Nodes 0 (red dashed lines) and 49 (entry point of green rectangular area) represent the source (emergency area) and sink (safe shelter), respectively.	47
2.20	Comparison of no. of evacuees with and without contraflow.	48
2.21	Comparison of quickest time with and without contraflow.	48
3.1	Network with movement capacity, cost between elements and storage capacity at elements.	53

3.2	Reconfigured network with a compatible sequence of sinks (dummy elements) after priority ordering on Figure 3.1.	53
3.3	Storage of flow at dummy elements and sink before switching the paths (general). . .	56
3.4	Storage of flow at dummy elements and sink after switching the paths (abstract). . . .	57
3.5	Solution in (a) before switching of paths and (b) after switching of paths.	57
3.6	Dynamic network with transit time.	61
3.7	Flow pattern in three cases: without switching, with complete switching and with partial switching.	66
3.8	Dynamic network with movement capacity and transit time of each element which is used to transship flow to it's adjacent element.	70
4.1	(a) Multi-commodity network, (b) Solution with priority order.	80
4.2	Dynamic network with κ_a and τ_a	81
4.3	The time expanded network of Figure 4.2. Blue, black and red paths are for commodity-1, commodity-2 and commodity-3, respectively.	82
4.4	Network with mix commodity at D_1 and D_2	83
4.5	Partitioning of destination with mix commodity in Figure!4.4 into commodity-wise separate destination in (a) and its prioritized static solution in (b).	84
4.6	(b) represents the time expanded layer graph Π^T of given network (a).	87
4.7	Two-commodity network (a) with arc capacity, cost and storage capacity. Figure (b) is its modified network with prioritized dummy nodes.	91
4.8	Super network $\tilde{\Pi}^*$ with dummy nodes and virtual sinks.	94
4.9	Time expanded multi-commodity network flow with intermediate storage, where black and red colors are for commodity-1 and commodity-2, respectively. Dotted arcs represent the dummy arcs.	95
4.10	(b) represents the network with proportional capacity sharing of (a).	101
4.11	Flow-dependent transit time function and transformation to the piecewise constant.	104
4.12	Bow graph of arc a with respect to the step function $\tau_a^{step}(\phi)$	107
4.13	Two-commodity two-way network with symmetric inflow-dependent transit times. 110	
4.14	(a) represents the two-commodity auxiliary network and (b) represents the static flow with saved arc capacities.	111
4.15	Condensed auxiliary network with capacity $\bar{\kappa}'$ and transit time τ^{\uparrow}	113
4.16	Graphical representation of inflow-dependent quickest time after and before lane reversals taking $\Delta = 2$. Due to lane reversals, quickest time is reduced significantly.	114

List of Tables

2.1	Maximum static path flow with anti-parallel path decomposition.	38
2.2	Comparison of no. of evacuees reaching to the destination with and without contraflow configuration.	48
2.3	Comparison of quickest time (in minutes) with and without contraflow configuration.	48
3.1	General and abstract path flows with intermediate storage	56
3.2	Abstract flow with intermediate storage in each time θ	64
3.3	Flow pattern of Figure 3.8 with time horizon $T = 10$	70
3.4	Maximum flow obtained from Figure 3.8 in different time horizon.	71
4.1	Multi-commodity flow with intermediate storage in each time θ	96
4.2	Inflow-dependent transit times on paths of given network without lane reversals.	110
4.3	Inflow-dependent transit time on each path.	111
4.4	Inflow-dependent transit times of paths after contraflow with $\Delta = 2$	113
4.5	Inflow-dependent transit times of paths before contraflow with $\Delta = 2$	113
A.1	Arcs/road segments with respective no. of lanes and vehicle transit times.	131
B.1	Arcs/road segments with respective capacities and vehicle transit times.	133

Chapter 1

Introduction

In mathematics, graph theory is the study of graphical structures of a physical phenomena with points and lines which is used to model pairwise relations between them. The line segments joining the points are considered as arcs and the points of intersection of arcs are considered as nodes. Graphs can be directed or undirected as per the direction on arcs are assigned or not. Two special nodes, known as source and sink, are the nodes from which flow started to move (origin) and flow stopped (destination), respectively. Alternatively, in directed graph, a source node is a node with no incoming arcs from other nodes, while a sink node is a node with no outgoing arcs. The rest of the nodes are termed as intermediate nodes. Note that a network consisting exactly of one source and one sink is said to be two-terminal network. The entities transshipped from the source to the sink through a network is considered as the network flow.

The possibly best practice to manage and improve the performance of the graphical network using different strategies is a network optimization. Network flow problems are the class of optimization problems with objective of the maximization of flow or minimization of time or cost. The maximum flow problem consists of maximization of flow out from the source to the destination without violating the capacity constraints (flow must be less or equal to the capacity) on the arcs and flow conservation constraints (inflow = outflow) on the nodes, where total amount of flow reaching to and out from the node is called inflow and outflow of the node, respectively. An earliest arrival flow problem concerns with maximization of flow not only within the given time horizon but it maximizes the flow at each point of time. Similarly, transshipment of the given amount of flow in minimum possible time is considered as the quickest flow problem, whereas minimum cost flow problem consists of obtaining the cheapest possible routes of sending a certain amount of flow through the network. The multi-commodity flow problem is a more complex problem than a single commodity one, in which different commodities are to be transshipped from respective sources to the corresponding sinks without violating the capacity and flow conservation constraints (Ahuja et al. (1993); Ford & Fulkerson (1962)).

1.1 Literature Review

General Network Flow

In network flow, if the flow transshipment problems are modeled with respect to node-arc form then it is termed as general network flow or classical network flow or simply, network flow. It is most commonly used method to represent the flow models and their solution strategies. L. R. Ford and D. R. Fulkerson are the pioneers of network flow problems. The Ford-Fulkerson algorithm is a very well known algorithm that tackles the max-flow min-cut problem (Ford & Fulkerson (1956), Ford & Fulkerson (1962)). They also have introduced and solved the maximum dynamic flow (MDF) problem, which concerns the shifting of maximum amount of flow from the origin node (source) to the destination node (sink) within the given time horizon, using temporally repeated flow (TRF) and time expanded network. Later, many researchers have presented their algorithms to improve the results such as shortest augmenting path algorithm of Edmonds & Karp (1972), blocking flow algorithm of Dinic (1970), push-relabel algorithm of Goldberg & Tarjan (1988) and many more. For detailed illustrations of maximum flow problems and their solution strategies, we refer to the book of Ahuja et al. (1993), book series of Kotsireas et al. (2018), survey papers of Aronson (1989); Dhamala et al. (2018); Kotnyek (2003) and the references therein.

An optimal flow over time problem, in which the amount of flow reaching the sink is maximized at each time step, is an earliest arrival flow (EAF) problem. Gale (1959) has shown that earliest arrival flows exist for a single source single sink network with constant transit time. Miniéka (1973) and Wilkinson (1971) designed the algorithms for finding earliest arrival flows by using successive shortest paths. Hoppe & Tardos (1994) presented the first polynomial time approximation algorithm to solve the problem. More detailed information on it can be found in Fleischer (2001); Ogier (1988); Ruzika et al. (2011); Tjandra (2003) and references therein. Similarly, another aspect of the flow problem is minimum cost flow problem in which total cost of flow transmission is to be minimized by using minimum cost paths. As an application of the problem in industry, finding the best delivery route from a factory to a warehouse by taking the road network with capacity and cost is commonly used. Most of other relevant problems can be modeled as a minimum cost flow problem, since it can be solved by using network simplex algorithm, and is very fundamental among all flow and circulation problems.

The inverse of maximum dynamic flow problem is the quickest flow (QF) problem, which concerns the minimization of time to transship the preassigned flow value. By applying a binary search to the MDF algorithm of Ford and Fulkerson, Burkard et al. (1993) provided the first polynomial time algorithm for the quickest flow problem. To provide strongly polynomial time algorithm, they upgraded the algorithm by incorporating a parametric approach to the minimum cost flow problem. By using the single quickest path, Chen & Chin (1990) and Rosen et al. (1991) transshipped the given amount of flow from the source to the sink in shortest possible

time. Lin & Jaillet (2015) solved the quickest flow problem by applying the cost-scaling algorithm of Goldberg & Tarjan (1990) within the same time complexity. For continuous time settings, Fleischer & Tardos (1998) provided the first polynomial time algorithm for the quickest flow problem by using a natural transformation.

Day to day traveling on the busy roads of urban cities, specially in office hour, every one can realize that the constant (fix) transit times on arcs/roads are not realistic but may vary as per the change on the flow, known as flow-dependent transit time. Merchant & Nemhauser (1978) were the first to provide the model for flow-dependent transit times, where flow-dependent cost function and exit function were considered in each arc. This model being non-linear and non-convex, Carey (1986) slightly improved the model with convex programming by replacing the maximum outflow with actual outflow. Flow-dependent models can be classified in two ways: inflow-dependent transit time and load-dependent transit time. Köhler et al. (2002) introduced the quickest single source and single sink flow problem in the setting of inflow-dependent transit time in which the current rate of flow is measured when it enters on the arc and moves with constant speed throughout the arc. Köhler & Skutella (2005) introduced a model in which, at any moment of time, the actual speed of the flow depends on the current amount of the flow on the arc, known as load.

Abstract Network Flow

A generalization of general (or classical) network is an abstract network which is associated with the set of elements and linearly ordered subset of elements, known as paths. In this network flow model, paths must satisfy the switching property: when two paths cross at an element then there must be a path that is a subset of the first path up to the crossing element and a subset of the second path after the crossing element (McCormick (1996)). More precisely, flows on crossing paths of an abstract network are diverged to the different directions by switching the flows on non-crossing sides (e.g., by means of traffic diversion, barricades, etc.).

The concept of abstract flow was first introduced in Hoffman (1974) by reviewing the first proof of the max-flow-min-cut theorem of Ford & Fulkerson (1956) considering the flows in terms of paths rather than on arcs. McCormick (1996) provided a polynomial time algorithm by using an oracle where the input is an arbitrary subset of elements whose output is either a path contained in that subset or states that no such path exists. The augmenting path structure is used in his path construction which satisfy the complementary slackness condition: every positive path meets the cut set exactly at one common element and every element of the cut is saturated. Martens (2007) computed the unsplittable and k -splittable abstract network flows for single as well as multi-commodity flows using shortest path oracle. Considering an additional attribute of weight on paths, Martens & McCormick (2008) extended the result of McCormick (1996) in more general case. Similarly, a polynomial time algorithm for lexicographic abstract

maximum flow and its use to prove the existence of abstract earliest arrival flow can be found in the PhD thesis of Kappmeier (2015).

Flow with Intermediate Storage

Except at the source and sink nodes, most of the network flow models allow the flow conservation constraints where inflow must be equal to outflow. Flow with intermediate storage is a new trend of research where flow out from the source is stored at some intermediate nodes if the flow cannot move forward due to either non-conservation of the flow or insufficient time to reach the next node from the current node. The flow with intermediate storage was first introduced and modeled by Pyakurel & Dempe (2020) by holding the excess flow at intermediate nodes. They introduced the maximum static and maximum dynamic flow problems and presented polynomial time solution strategies to solve them using lexicographically maximum flow approach. Similarly, efficient algorithms for the universal maximum flow problem with intermediate storage in two-terminal series-parallel networks can be found in Pyakurel & Dempe (2021). The maximum dynamic flow model for hesitant fuzzy evacuation with intermediate storage at nodes can be found in Gerasimenko et al. (2022). In an abstract network topology, Pyakurel et al. (2022) introduced the maximum static, lexicographic maximum static and maximum dynamic flow problems with intermediate storage and presented algorithms to solve the problems in polynomial time complexity. They have introduced the temporally repeated flow (TRF) technique on switched paths to obtain the flow with intermediate storage. Recently, Dhamala et al. (2024) introduced the generalized maximum static and dynamic flow problems with intermediate storage in lossy network and present efficient algorithms for the solutions.

Contraflow

A network in which at least a pair of nodes connected with anti-parallel arcs (oppositely directed parallel arcs) is called a two-way network. In such a network topology, contraflow (lane reversal) is one of the best and widely used flow improvement technique in which direction of opposite arcs are reversed in such a way that it increase the amount of flow and reduce the traversal time. Because of the complexity on deciding the flipping of edges in back or forth direction for the best solution, the contraflow problem becomes \mathcal{NP} -hard. Kim & Shekhar (2005) proved that contraflow problem with bounded evacuation time is \mathcal{NP} -complete by transforming the problem to 3SAT problem. The polynomial solutions to this problem is due to reversing the direction of arcs at initial time. By incorporating the road capacity constraints, multiple sources, congestion, and scalability, Kim et al. (2008) presented the first macroscopic approach based on graph theory for the solution of a contraflow network reconfiguration. Rebennack et al. (2010) presented a polynomial time algorithm to solve the maximum flow problem in two-terminal network by reverting the direction of arcs at time zero and keeping them fixed afterwards. Different aspects of contraflow configurations with various mathematical models, heuristics, optimization

and simulation techniques can be found in Arulsevan (2009); Baumann & Köhler (2007); Kim & Shekhar (2005); Pyakurel et al. (2017). Using the natural transformation of Fleischer & Tardos (1998), Pyakurel & Dhamala (2016) introduced the continuous time dynamic contraflow model and presented efficient algorithms to solve the maximum, quickest, and earliest arrival flow problems.

The maximum dynamic contraflow (MDCF) problem with intermediate storage and its polynomial solution can be found in Pyakurel & Dempe (2020). Similarly, in general as well as two-terminal series-parallel networks, efficient algorithms for the universal maximum flow and contraflow problems with intermediate storage can be found in Pyakurel & Dempe (2021). Pyakurel et al. (2019) introduced the concept of partial contraflow, in which only necessary arc capacities are reversed to increase the flow value and remaining arc capacities are saved for other emergency purpose like facility allocation and logistic supports. For these contraflow problems, networks are considered with symmetric transit times in anti-parallel arcs. For the contraflow with different transit times on anti-parallel arcs, Bhandari & Khadka (2020) studied the solution strategies for maximum and earliest arrival flow with symmetric reversal of each arc, where the auxiliary network in their modified network is an undirected network and symmetric reversal exists in the arcs after contraflow configuration as similar to Rebennack et al. (2010). Nath et al. (2021) investigated on the concept of contraflow with orientation dependent transit times and solved the maximum dynamic and quickest contraflow problems in polynomial time. Similarly, maximum dynamic contraflow problem in a general network and earliest arrival contraflow problem in two-terminal series-parallel network with asymmetric transit times on anti-parallel arcs allowing the intermediate storage of flow can be found in Khanal et al. (2021a).

Multi-commodity Flow

Multi-commodity flow (MCF) problem concerns with shipment of several different commodities from respective sources to corresponding sinks through a network without violating the capacity constraints associated with the arcs. Many network routing and network design problems such as message routing in telecommunication, production scheduling and planning, supply chains network, scheduling and routing in logistics and transportation, distribution system design, etc. are some multi-commodity network flows. The static and dynamic multi-commodity flow problems and their solution procedures can be found in Ahuja et al. (1993); Ali et al, (1980); Assad (1978); Kennington (1978). The static multi-commodity flow problem is polynomial time solvable by using the ellipsoid or interior point method, whereas dynamic multi-commodity flow problem is \mathcal{NP} -hard, Hall et al. (2007a); Hall et al. (2007). By using time expanded network, Kappmeier (2015) provided the solution of maximum dynamic multi-commodity flow problem and multi-source single sink multi-commodity earliest arrival transshipment problem in pseudo-polynomial time complexity.

The maximum multi-commodity flow over time problem with partial contraflow is presented by Pyakurel et al. (2020). Dhamala et al. (2020) presented algorithms to solve the quickest multi-commodity contraflow problem with partial reversal of arcs, where given amount of flow is to be transshipped in minimum possible time. Khanal et al. (2021) presented the pseudo-polynomial time algorithms for maximum multi-commodity flow and contraflow problems with intermediate storage where contraflow is configured for the network with symmetric as well as asymmetric transit times on arcs. Priority based static multi-commodity flow problem and its polynomial time solution strategy is presented in Khanal et al. (2020).

Flow with Facility Allocation

The facility allocation problem concerns with allocation of the facilities at appropriate locations and optimization of the flow value on the facilitated network. The impact of the placement of facility is that it decreases the capacity of arc by the size of the facility. Weber (1909) introduced the first location flow theory with the application in industries. Different discrete location models and algorithms with applications can be found in Daskin (1997). Later on, Hamacher et al. (2013) introduced the single and multiple flow location (FlowLoc) problems by combining the network flows and locational analysis, and presented polynomial time algorithms for the 1-FlowLoc problem and polynomial time heuristics for the q-FlowLoc problem. By incorporating the contraflow problem, Dhungana & Dhamala (2019) presented the polynomial time solution strategies to solve the maximum static and maximum dynamic ContraFlowLoc problems. Similarly, Nath et al. (2021) solved the quickest flow location problem with single and multiple facilities and presented polynomial time algorithm and polynomial time heuristics, respectively.

Bi-level Optimization

Bi-level optimization problem, also known as leader-follower problem, is hierarchical optimization problem having two decision makers. The leader problem is considered as upper level whereas follower problem is considered as lower level. This problem was first introduced by von Stackelberg (1934) in his habilitation thesis, where the problem was originated on economic game theory. Bracken & McGill (1973) has given the first formal definition of bi-level problem for the military application. The basic models and the characterizations of the problem, areas for application and the existing solution approaches can be found in the review papers of Wen & Hsu (1991); Vicente & Calamai (1994). In addition, we refer to the papers of Anandalingam & Friesz (1992); Ben-Ayed (1993), survey papers of Colson et al. (2005); Colson et al. (2007) and the books of Bard (1998); Dempe (2002); Dempe (2020); Dempe & Zemkoho (2020) and the references therein for the detailed illustrations.

1.2 Rationale of the Study

In day to day life, every individual probably has realized the transshipment of more than one different commodities, directly or indirectly. Our main concern in this thesis is to deal with the maximum and quickest flow problems and their solution strategies on multi-commodity flow network, whose special case is the single commodity flow with only one commodity. Similarly, improvement of the flow in existing network topology is very essential task when instant extension of the topological structure is impossible, especially for post disaster or peak hour traffic managements. In this research study, intermediate storage of flow, congestion minimization by crossing elimination and contraflow configuration with a novel technique are used as the key tools of flow improvement. For the convenient of the study and sequential development of the tasks, we start with single commodity flow problems and move to the multi-commodity one.

Excess flow is the flow out from the source but unable to reach the destination due to capacity constraints of the arcs or insufficient time to reach the next node. Instead of returning back, storage of such flow at intermediate shelters plays an important role to improve the flow transmission. Theoretic and computational results on the flow with intermediate storage by introducing temporally repeated solution strategy is a fruitful achievement of the study. Incorporating the intermediate storage, we solve the maximum multi-commodity flow problem. Similarly, at the time of disasters, every individual may not hurt equally. So, the study of priority based evacuation with respect to case sensitivity using multi-commodity flow model is a useful task.

On solving single as well as multi-commodity flow problems, contraflow with symmetric or orientation dependent transit times improves the amount of flow significantly. By adopting this technique, the quickest multi-commodity flow with partial lane reversal and inflow-dependent quickest multi-commodity flow with partial lane reversal are the reasonable areas of our study. Together with this, introducing the new concept to solve the contraflow problem in asymmetric network using anti-parallel path decomposition and its computational findings for maximum and quickest contraflow problems for the single commodity network flow is remarkable. Similarly, route-based evacuation procedure presented for asymmetric network using anti-parallel path decomposition is also an achievement of the study.

At the time of disasters, every individual desires to leave the danger zone as quickly as possible, which causes extensive congestion at the crossing of the roads. Abstract network flow strictly prohibits the crossing of the flow at intersections by switching the flow towards the non-crossing sides. It can be a milestone to save the lives at the time of evacuation and also helpful in distribution or supply chain management. An important aspect of this study is to integrate the concept of abstract network flow and intermediate storage, and develop the mathematical models and their solution strategies. Similarly, providing the essential facilities like foods and medicines to the evacuees is another important issue. We design the model and present the solution strategy to solve the problem using bi-level optimization.

1.3 Objectives

The general objective of this study is to construct mathematical models and develop solution procedures for single as well as multi-commodity flow with intermediate storage. As a special case of the multi-commodity flow, we start our study from single commodity flow with intermediate storage and turn to the multi-commodity one. We aim to study the flow problems with different aspect of transit times on arcs/paths such as constant, flow-dependent, symmetric and asymmetric. The specific objectives of the study are as follows.

- Develop the mathematical models for single as well as multi-commodity flow problems and connect them with evacuation planning problems.
- Extend the mathematical models of above problems by incorporating the intermediate storage of flow and study the importance in evacuation planning problems.
- Design the solution strategies to solve above problems and test the computational performance of the algorithms.

1.4 Structure of the Thesis

The structure of the thesis is as follows. Chapter 2 concerns with general (classical) network flow which is divided in four sections - notations and terminologies, general network flow models, flow with intermediate storage and contraflow. The temporally repeated solution strategy for MDF and EAF problems with intermediate storage and the asymmetric contraflow with anti-parallel path decomposition are novel work of this research study.

In Chapter 3, network flow problems in abstract network are studied. We divide the chapter in two sections - abstract network flow with intermediate storage and abstract network flow with partial switching. For the abstract flow with intermediate storage, we solved the maximum static, lexicographic maximum static and maximum dynamic flow problems whereas for abstract flow with partial switching, maximum and quickest flow problems are introduced and solved in polynomial time.

Chapter 4 refers to the multi-commodity network flow problems and their solution strategies. It is divided in two sections - maximum and quickest multi-commodity flows. Priority based maximum MCF, maximum and quickest MCF problems with proportional as well as flow-dependent capacity sharing, maximum MCF with intermediate storage and inflow-dependent quickest MCF with partial contraflow configuration are the findings of our research study.

The bi-level formulation of the facility allocation problem is presented in Chapter 5. Two solution approaches - a naive approach and Karush-Kuhn-Tucker (KKT) transformation are used to solve the problem. The thesis is summarized and concluded in Chapter 6.

Chapter 2

General Network Flow

In this chapter, we discuss the basic notations used throughout the thesis which are essential for the development of mathematical models. As the fundamental approach of the study is based on the network optimization, basically linear programming, graphical representation of general network with set of nodes (junctions) and arcs (links) having specific attributes are considered.

In mathematics, network is often referred to as graph which reflect the physical phenomena with a set of objects (nodes or vertices) that are connected together. Network with different attributes is used as an input of mathematical optimization problem. It is used to analyze and design the mathematical model of large systems such as communication, transportation, manufacturing networks, etc. and provide optimal solution. A network is called a directed network if the links (arcs) joining the pair of nodes are pointed in only one direction. If all the links are bidirectional or undirected, then the network is an undirected network. For example, Figure 2.1 presented alongside is a directed network having four nodes $\{s, x, y, t\}$ and five arcs $\{(s, x), (s, y), (y, x), (x, t), (y, t)\}$ with two attributes, a capacity and a transit time, on each arc. Here, s and t are two special nodes, the source node with no incoming arcs and the sink node with no outgoing arcs, respectively. Rest of the nodes are called intermediate nodes. For any arc (u, v) , node u stands for the tail node and v , the head node. Our assumption is that the network does not contain parallel arcs (i.e., two or more arcs with the same tail and head nodes) except for the contraflow configuration.

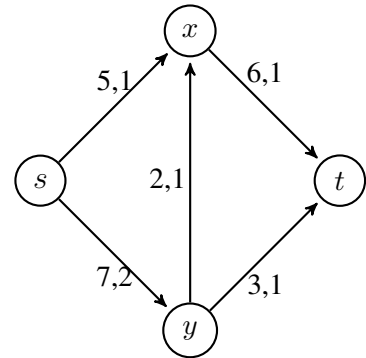


Figure 2.1: Directed network with capacity and transit time.

For any arc (u, v) , node u stands for the tail node and v , the head node. Our assumption is that the network does not contain parallel arcs (i.e., two or more arcs with the same tail and head nodes) except for the contraflow configuration.

In this chapter, we first fix the mathematical notations used throughout the thesis and present the mathematical models. Thereafter, flow problems with intermediate storage and the TRF solutions will be discussed. Symmetric and asymmetric contraflow problems and route-based evacuation will be discussed in the last section of this chapter.

2.1 Notations and Terminologies

Consider a directed network Π having a set of nodes \mathcal{N} with cardinality $|\mathcal{N}| = n$ and a set of arcs \mathcal{A} (i.e., set of ordered pair of nodes) with cardinality $|\mathcal{A}| = m$. Most commonly, the nodes are also termed as vertices, elements or junction points and the arcs as links, branches or edges. Each arc $a = (u, v) \in \mathcal{A}$ has an initial point $u = tail(a)$ and a final point $v = head(a)$. Each arc consists of two attributes, non-negative integer capacity function $\kappa : \mathcal{A} \rightarrow \mathbb{Z}_0^+$ that limits the flow on arc and the non-negative real valued transit time function $\tau : \mathcal{A} \rightarrow \mathbb{R}_0^+$ that measures the transmission time from u to v . Any particle starting at node u at time θ reaches to v at time $\theta + \tau_a$. Similarly, we denote

$$\Gamma_u^{out} = \{a \in \mathcal{A} : tail(a) = u\} \text{ and } \Gamma_u^{in} = \{a \in \mathcal{A} : head(a) = u\}$$

as the set of outgoing arcs from and incoming arcs to the node u , respectively. The time horizon of the flow transmission T is represented in discrete time setting as $\mathcal{T} = \{0, \dots, T\} \subset \mathbb{Z}_0^+$. In continuous time setting, flow is transmitted continuously over the time and is denoted by $\mathcal{T} = [0, T) \subset \mathbb{R}_0^+$. However, this study is mainly focused on discrete time setting, except for inflow-dependent transit time. We represent source and sink nodes by s and t , respectively, so that $\Gamma_s^{in} = \emptyset$ and $\Gamma_t^{out} = \emptyset$, except for the two-way network. Rest of the nodes are termed as intermediate nodes denoted by $\mathcal{I} = \mathcal{N} \setminus \{s, t\}$. The collection $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$ represents the dynamic network topology that captures the graphical structure of physical phenomena.

In some aspect of the flow transmission, we also have to consider the capacity function on the nodes defined by $\nu : \mathcal{N} \rightarrow \mathbb{Z}_0^+$ so that the dynamic network becomes $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$. The cost function $c : \mathcal{A} \rightarrow \mathbb{R}_0^+$ can also be included in the network as per the necessity, which represents the per unit cost of flow transmission. In a network, if the time components τ and T are absent, then the network becomes a static one and is represented by $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, s, t)$. Network defined above is a directed network because each arc is directed from a fix tail node to a fix head node. If the head and tail nodes of the arcs are undefined, then the network becomes undirected. A two-terminal network is one which consists exactly one source and one sink where as multi-terminal network is one which contains more than one source and/or sink.

Chain, Path and Cycle. Consider a subset $\{u_1, \dots, u_l\} \subseteq \mathcal{N}$ ($l \geq 2$) of distinct nodes of the network Π such that the ordered pair $a_i = (u_i, u_{i+1})$ forms an arc in \mathcal{A} , $\forall i = 1, \dots, l-1$. Then the sequence of nodes and arcs

$$u_1, a_1, u_2, \dots, a_{l-1}, u_l \tag{2.1}$$

is called a chain from u_1 to u_l . If $u_1 = s$ and $u_l = t$, then it is termed as a $s-t$ chain. Similarly, if first and last components of the sequence are same (i.e. $u_1 = u_l$), then it forms a cycle. The chain formed in equation 2.1 is said to be a path from u_1 to u_l if $a_i = (u_i, u_{i+1})$ or $a_i =$

$(u_{i+1}, u_i), \forall i = 1, \dots, l - 1$. Thus, the direction of arcs in a path may be forward or backward. The chain is also termed as standard chain or even commonly, directed path. A path which is not a chain is also known as non-standard chain. We denote the set of directed paths (chains) and cycles by \mathcal{P} and \mathcal{C} , respectively.

Time Expanded Network. For the given network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$ with time horizon T , a time expanded network of Π is denoted by $\Pi^T = (\mathcal{N}^T, \mathcal{A}^T, \kappa, \tau, s(\theta), t(\theta), T)$ in which the original network is splitted over the time T with duplication of each node $u \in \mathcal{N}$ at every time period by $u(0), \dots, u(T)$. Thus, \mathcal{N}^T contains $T + 1$ copies of original nodes. Arc set $\mathcal{A}^T = \mathcal{A}_M \cup \mathcal{A}_H$ consists the arcs of two types, movement arcs $\mathcal{A}_M = \{(u(\theta), v(\theta + \tau_a)) : a = (u, v) \in \mathcal{A}, \theta \in \mathcal{T}\}$ and holdover arcs $\mathcal{A}_H = \{(u(\theta), u(\theta + 1)) : u \in \mathcal{N}, \theta \in \mathcal{T}\}$. Movement arcs carry the objects from tail node to the head node beyond the limit of their predefined capacities and transit times whereas holdover arcs hold the flow for unit time step with infinite (sufficiently large) capacity. Here, Figure 2.2 represents the time expanded network of Figure 2.1 with time horizon $T = 5$ where solid arrows represent the movement arcs and dashed arrows represent the holdover arcs.

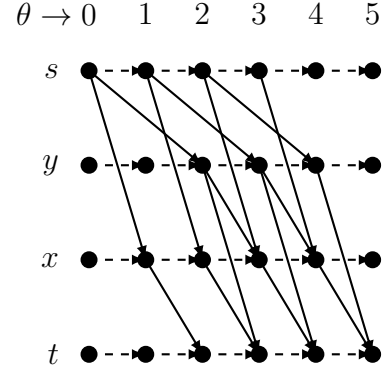


Figure 2.2: Time expanded network of Figure 2.1 with $T = 5$.

Flow Network. In graph theory, a flow is an object or a material that is to be shipped from one point to another point. The amount of flow on an arc can not exceed its capacity, known as capacity constraint. A directed graph with capacitated arcs which transships the flow from a fixed node (source) to another fixed node (sink) without violating the capacity constraint on each arc is called flow network. Flow network can be static or dynamic according to absence or presence of the time component on it.

Network Flow Problems. Different classes of computational problems in which input is the flow network and whose output is the execution of flow are called network flow problems. At each arc, the capacity constraint needs to be satisfied, and at each node except source and sink, flow conservation (inflow = outflow) has to be satisfied. Some commonly used network flow problems are maximum flow, minimum cost flow, quickest flow, earliest arrival flow, multi-commodity flow, etc.

Maximum flow problem concerns with shipment of maximum amount of flow from the source to the sink. For the flow network with cost in arcs, minimum cost flow problem seeks to find the minimum possible cost to transship the given amount of flow. A similar variant of the problem in dynamic network is the quickest flow problem in which minimum possible time is to be executed to transship the given amount of flow from source to the sink. The calculation of maximum flow, not only in time horizon T but at each time step $\theta \in \mathcal{T}$, is an earliest arrival

flow problem. Similarly, a flow problem with more than one different commodities in which each commodity is transshipped from respective source to corresponding sink with total flow amounts together respects the capacity constraint in each arc is known as multi-commodity flow problem.

2.2 General Network Flow Models

In this section, a brief explanation of the flow models in static as well as dynamic network topology will be considered. These models are based on general network where each equation is defined on node-arc form. Apart from this, only in Chapter 3, the model in element-path form will be formulated for an abstract network flow.

2.2.1 Static Flow Model

Consider a static network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, s, t)$, where symbols have their usual meaning. The static flow ϕ is defined by the function $\phi : \mathcal{A} \rightarrow \mathbb{R}_0^+$ satisfying the following constraints.

$$\sum_{a \in \Gamma_u^{out}} \phi_a - \sum_{a \in \Gamma_u^{in}} \phi_a = \begin{cases} |\phi| & \text{for } u = s \\ -|\phi| & \text{for } u = t \\ 0 & \text{for } u \in \mathcal{I} \end{cases} \quad (2.2a)$$

$$0 \leq \phi_a \leq \kappa_a, \quad \forall a \in \mathcal{A} \quad (2.2b)$$

Here, equation (2.2a) represents the flow conservation constraint for each intermediate node and non-conservation of flow at source and sink. The positive and negative signs of $|\phi|$ in right hand side stands for supply from the source and demand to the sink, respectively. Equation (2.2b) represents the capacity constraint for each arc where flow is non-negative and must not exceed the capacity of the arc. The notation $|\phi|$ refers to the value of the flow ϕ . Flow function satisfying only the capacity constraint is called pseudo-flow whereas if it satisfy both flow conservation and capacity constraints, then the flow is feasible. The objective functions can be considered depending on the goal of the problem types as follows.

Maximum Static Flow. If the goal of the mathematical formulation is to maximize the flow from s to t , then the objective function with flow value $|\phi|$ is

$$\max |\phi| = \sum_{a \in \Gamma_s^{out}} \phi_a = \sum_{a \in \Gamma_t^{in}} \phi_a. \quad (2.3)$$

Thus, maximize $|\phi|$ subject to the constraints (2.2a–2.2b) is a maximum static flow (MSF) problem. Analogously, we say that flow function ϕ is maximum flow function if there does not

exist a flow function ϕ^* such that $|\phi^*| > |\phi|$.

Minimum Cost Flow. Minimum cost flow problem concerns with the minimization of the cost of given flow not exceeding the maximum flow. Thus, if the goal is to minimize the total cost of flow transmission, then the objective function becomes

$$\min c(\phi) = \sum_{a \in \mathcal{A}} c_a \phi_a, \quad (2.4)$$

and equation (2.4) together with constraints (2.2a–2.2b) is a minimum cost flow problem.

Minimum Cut. The dual problem of maximum flow problem is a minimum cut problem. Let (Π_1, Π_2) be a partition of network Π in to two sub-networks with corresponding partition of nodes \mathcal{N}_1 and \mathcal{N}_2 such that $s \in \mathcal{N}_1, t \in \mathcal{N}_2, \mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ and $\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$. Define a set \mathcal{X} of minimum cut arcs as

$$\mathcal{X} = \left\{ a = (u, v) \in \mathcal{A} : u \in \mathcal{N}_1, v \in \mathcal{N}_2, \sum_a \kappa_a \text{ is minimum} \right\}.$$

The sum of capacities of the minimum cut arcs in \mathcal{X} equals the maximum static flow in Π .

Path Flow Decomposition. Let \mathcal{P} be a set of directed paths (chains) from the source to the sink and $P \in \mathcal{P}$. Define a binary variable γ with respect to the arc a by

$$\gamma_a = \begin{cases} 1 & \text{if } a \in P \\ 0 & \text{otherwise.} \end{cases}$$

Then the flow function $\phi_P : \mathcal{P} \rightarrow \mathbb{R}_0^+$ satisfying the path capacity constraint

$$\sum_{P \in \mathcal{P}: a \in P} \gamma_a \phi_P \leq \kappa_a \quad (2.5)$$

is a static arc-path flow function from the source to the sink. The total flow function on $s - t$ paths is $\phi = \sum_{P \in \mathcal{P}} \phi_P$. With this argument, every arc flow can be decomposed to the path flow.

Observation 2.1 (Ford & Fulkerson (1962)). If ϕ is a node-arc flow from source node s to the sink node t with flow value $|\phi| > 0$, then there exists the path (chain) flow from s to t such that every arc of the path has flow $\phi > 0$.

2.2.2 Dynamic Flow Model

Dynamic flow, also known as flow over time, is an accumulation of the static flow over the time which can be reached to the sink within the given time horizon T . The dynamic flow function Φ , with flow value $|\Phi|$, defined on the dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$ is the collection

of non-negative arc flow functions $\Phi_a : \mathcal{A} \times \mathcal{T} \rightarrow \mathbb{R}_0^+$ satisfying the following constraints.

$$\sum_{a \in \Gamma_u^{out}} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta) - \sum_{a \in \Gamma_u^{in}} \sum_{\beta=0}^{\theta} \Phi_a(\beta - \tau_a) \leq 0, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.6a)$$

$$\sum_{a \in \Gamma_u^{out}} \sum_{\theta=\tau_a}^T \Phi_a(\theta) - \sum_{a \in \Gamma_u^{in}} \sum_{\theta=0}^T \Phi_a(\theta - \tau_a) = \begin{cases} |\Phi| & \text{for } u = s \\ -|\Phi| & \text{for } u = t \\ 0 & \text{for } u \in \mathcal{I} \end{cases} \quad (2.6b)$$

$$0 \leq \Phi_a(\theta) \leq \kappa_a, \quad \forall a \in \mathcal{A}, \theta \in \mathcal{T} \quad (2.6c)$$

Here, $\Phi_a(\theta)$ represents the flow starting the tail node of arc a at time θ . Equation (2.6a) represents the weak flow conservation at intermediate nodes in time step θ , where outflow may not exceeds the inflow. The flow conservation at time horizon T is presented in the third condition of equation (2.6b), whereas the first and second conditions are the supply and demand, respectively. The satisfiability of capacity constraint in each arc at any time step θ is represented by equation (2.6c), where capacity is an upper bound of the flow.

As in static flow problem, the objective function of the dynamic flow problem depends on the goal of the problem. The objective of flow maximization problem within the given time horizon T is

$$\max |\Phi| = \sum_{a \in \Gamma_s^{out}} \sum_{\theta=0}^T \Phi_a(\theta) = \sum_{a \in \Gamma_t^{in}} \sum_{\theta=\tau_a}^T \Phi_a(\theta - \tau_a), \quad (2.7)$$

where $|\Phi|$ represent the value of the flow induced by the flow function Φ . Equation (2.7) together with constraints (2.6a–2.6c) is known as maximum dynamic flow (MDF) model. The dynamic flow function Φ is a maximum dynamic flow function if no any other dynamic flow function Φ^* can be found satisfying $|\Phi^*| > |\Phi|$.

In the quickest flow (QF) problem, the objective is to minimize the makespan for the shipment of the given amount of flow. Thus, the quickest flow problem is to minimize T satisfying the constraints (2.6a–2.6c), for which $|\Phi|$ is given amount of flow to be shipped from the source (i.e., supply). Analogously, if T is the quickest time to satisfy the supply $|\Phi|$ and denoted by $T(|\Phi|)$, then there does not exist T^* such that $T^*(|\Phi|) < T(|\Phi|)$.

An earliest arrival flow (EAF) problem is to obtain the maximum flow out from the source which is to be reached maximally to the sink at each time step $\theta \in \mathcal{T}$. Thus, The mathematical model for an EAF problem is

$$\max |\Phi(\theta)| = \sum_{a \in \Gamma_s^{out}} \sum_{\beta=0}^{\theta} \Phi_a(\beta) = \sum_{a \in \Gamma_t^{in}} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta - \tau_a), \quad \forall \theta \in \mathcal{T} \quad (2.8)$$

satisfying constraints (2.6a–2.6c).

Temporally Repeated Flow (TRF). On solving dynamic flow problems, most commonly used technique is a TRF in which constant rate of static flow is repeated along the decomposed paths (chains) within the time frame. For a feasible static flow ϕ , let ϕ_P be its flow decomposition to the directed paths $P \in \mathcal{P}$. Also, let $\tau_P = \sum_{a \in P} \tau_a$ be path time and T be given time horizon. Flow sending from the source with constant flow rate ϕ_P along the decomposed paths $P \in \mathcal{P}$ repeatedly during the time steps 0 to $T - \tau_P$ executes the dynamic flow, known as temporally repeated flow (TRF). Throughout the thesis, we use the term ‘paths’ for flow carrying directed paths with length not exceeding the time horizon T , i.e., $\sum_{a \in P} \tau_a \leq T, \forall P \in \mathcal{P}$.

Observation 2.2 (Ford & Fulkerson (1962); Skutella (2009)). Let ϕ_P be the path flow decomposition of a feasible static s - t flow ϕ such that $\phi_P = 0$ for $\tau_P > T$. Then the value of the corresponding TRF Φ is

$$|\Phi| = \sum_{P \in \mathcal{P}} (T + 1 - \tau_P) \phi_P = (T + 1)|\phi| - \sum_{a \in \mathcal{A}} \tau_a \phi_a \quad (2.9)$$

Continuous Time Dynamic Flow. To this point, we discussed on discrete time maximum dynamic flow problems, where dynamic flow function Φ assigns the flow from source node at each time step $\theta = 0, \dots, T$ satisfying the capacity constraints. Here, we discuss the flow in continuous time setting. A continuous dynamic flow function $\tilde{\Phi}$ is defined as the flow rate per unit time $[\theta, \theta + 1)$ that leaves from the source at each moment of time without violating the capacity constraints on the arcs.

Fleischer & Tardos (1998) established the strong relation between discrete and continuous flow models by using natural transformation. This natural transformation defines the continuous dynamic flow for time interval $[\theta, \theta + 1)$ with $\tilde{\Phi}_a[\theta, \theta + 1) = \Phi_a(\theta)$, where $\Phi_a(\theta)$ is the discrete dynamic flow entering arc $a \in \mathcal{A}$ at each time step $\theta = 0, \dots, T$. The discrete time flow can be transformed to continuous time flow as follows: any discrete flow over time Φ_a with integral time horizon T is equivalent to the continuous flow over time $\tilde{\Phi}_a[\theta, \theta + 1)$ by incorporating the flow Φ entering arc a at time step $\theta \leq T - \tau_a$ as a constant flow rate on arc a during the unit time interval $[\theta, \theta + 1)$. Mathematically,

$$\int_{\theta}^{\theta+1} \tilde{\Phi}_a(\beta) d\beta = \Phi_a(\theta), \quad \forall a \in \mathcal{A}.$$

2.3 Flow with Intermediate Storage

For the flow with intermediate storage, network topology $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$ with node capacity ν is considered, where ν_u represent the holding capacity of the intermediate node u .

For the existence of a feasible solution, the capacity of intermediate node is considered at least the sum of capacity of the incoming arcs, i.e., $\nu_u \geq \sum_{a \in \Gamma_u^{in}} \kappa_a$, $\forall u \in \mathcal{I}$ (Pyakurel & Dempe (2020)). Similarly, capacity of source and sink are considered sufficiently large, say infinity. Flow with intermediate storage is applicable in various supply chain management situations like industrial product distribution, water supply, electricity supply, etc. and also in evacuation planning and disaster management. In this section, we discuss on maximum static and dynamic flow problems with intermediate storage and their solution procedures in brief.

2.3.1 Maximum Static Flow with Intermediate Storage

For a given network Π , the objective of the maximum static flow problem with intermediate storage is to maximize the flow leaving from source s that is to be sent to the sink t via $s-t$ paths by allowing the storage of maximum excess flow at intermediate nodes with storage capacity

$$\nu_u \geq \sum_{a \in \Gamma_u^{in}} \kappa_a, \quad \forall u \in \mathcal{I}.$$

Flow Model. For the static flow function ϕ on the given network Π , define an arc flow function $\phi_a : \mathcal{A} \rightarrow \mathbb{R}_0^+$ that transships the flow on arcs and the excess flow function $\hat{\phi}_u : \mathcal{I} \rightarrow \mathbb{R}_0^+$, $\forall u \in \mathcal{I}$ that holds the flow at intermediate nodes. As in Pyakurel & Dempe (2020), the linear programming formulation of static flow with intermediate storage is as follows.

$$\max |\phi| \tag{2.10a}$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \phi_a = |\phi| = \sum_{a \in \Gamma_t^{in}} \phi_a + \sum_{u \in \mathcal{I}} \hat{\phi}_u \tag{2.10b}$$

$$\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a = \hat{\phi}_u, \quad \forall u \in \mathcal{I} \tag{2.10c}$$

$$0 \leq \phi_a \leq \kappa_a, \quad \forall a \in \mathcal{A} \tag{2.10d}$$

$$0 \leq \hat{\phi}_u \leq \nu_u, \quad \forall u \in \mathcal{I} \tag{2.10e}$$

Here, equation (2.10a) is an objective function which intends to maximize the flow. Equation (2.10b) represents that the static flow with intermediate storage is the total flow out from the source which is equal to the sum of inflow at the sink and the excess flow at the intermediate nodes. The excess flow at each intermediate node is represented by equation (2.10c). The constraint in (2.10d) represents the capacity constraint on each arc which means that the arc flow is bounded by its capacity. Similarly, constraints (2.10e) represent the boundedness of the excess flow at each intermediate node by the storage capacity of the node.

Solution Procedure. To solve the problem, first step is to decide the priority order of nodes,

which can vary as per the problem instance. For example, if the problem instance is to solve the problem for distribution network, then order priority can be fixed as ‘higher in demand higher in priority’. Similarly, if the problem instance is for the evacuation network in which source is a danger zone and sink is a safe zone, then ‘farther the distance from the source higher in priority’ can be set with assumption that the places far from the danger zone are relatively safer. Pyakurel & Dempe (2020) used the distance from the source to fix the priority order of intermediate nodes as their problem was the disaster management.

As considered in Pyakurel & Dempe (2020), the sink is most appropriate place to store the flow with sufficient (infinite) node capacity. So, possible maximum flow is shifted to their respective sink as the first priority node. To store the excess flow at intermediate nodes, they set the priority order with respect to the distance or cost. For this, they have calculated the shortest distance $d_{[s,u]}$ (considering cost as distance) of each $u \in I$ by using algorithm of Dijkstra (1959). The minimum cost path is considered as the shortest path and the priority is given to the farthest node among the nodes with shortest distance. That is, if $d_{[s,u_1]} > d_{[s,u_2]}$, then u_1 has higher priority than u_2 and is denoted by $u_1 \succ u_2, \forall u_1, u_2 \in I$. The equality in distance implies the mutability in the priority order. We adopt the same technique for the priority.

The second step of the solution procedure is to construct a modified network Π^* by creating dummy node u^* of each prioritized node $u \in \mathcal{I}$, denoted by $\Pi^* = (\mathcal{N}^*, \mathcal{A}^*, \kappa, \nu, c, s, D)$, where $\mathcal{N}^* = \mathcal{N} \cup \{u^*\}$. A dummy arc $a^* = (u, u^*)$ is created to connect dummy node u^* with cost $c_{a^*} = 0$ and capacity $\kappa_{a^*} = \nu_u = \nu_{u^*}$ so that $\mathcal{A}^* = \mathcal{A} \cup \{a^*\}$. The priority of dummy node u^* is same as of u . The network so created is a single source multi-sink network with $D = \{t\} \cup \{u^* : u \in \mathcal{I}\}$. This construction obviously helps to satisfy the flow conservation at each intermediate node. Finally in third step, lexicographically maximum flow with priority order is obtained by using the algorithm of Minieka (1973). The algorithmic framework, as in Pyakurel & Dempe (2020), is presented in Algorithm 1.

Algorithm 1: Maximum static flow algorithm with intermediate storage.

Input : Given static network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, c, s, t)$.

Output: Maximum static flow with intermediate storage in Π .

1. Fix the priority order of nodes.
 2. Construct a modified network $\Pi^* = (\mathcal{N}^*, \mathcal{A}^*, \kappa, \nu, c, s, D)$.
 3. Compute the lexicographic maximum static flow in Π^* with priority order using Minieka (1973).
 4. Transform the solution to the original network Π by removing the dummy nodes and the dummy arcs, and shifting the flow at dummy nodes to their respective original nodes.
-

Theorem 2.1 (Pyakurel & Dempe (2020)). *Algorithm 1 solves the maximum static flow problem with intermediate storage optimally in polynomial time complexity.*

2.3.2 Maximum Dynamic Flow with Intermediate Storage

For a given dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$, the objective of the maximum dynamic flow problem with intermediate storage is to maximize the flow leaving the source s that is to be sent to the sink t via $s - t$ paths by allowing the storage of maximum excess flow at the intermediate nodes with storage capacity $\nu_u \geq \sum_{a \in \Gamma_u^{in}} \kappa_a, \forall u \in \mathcal{I}$ within time horizon T .

Flow Model. Consider a dynamic flow function Φ , defined on a dynamic network Π , as the collection of non-negative arc flow function $\Phi_a : \mathcal{A} \times \mathcal{T} \rightarrow \mathbb{R}_0^+$, where $\Phi_a(\theta)$ represents a flow starting from the tail of arc a at time θ . Similarly, let us define an excess flow function $\hat{\Phi}_u : \mathcal{I} \times \mathcal{T} \rightarrow \mathbb{R}_0^+$ that stores the flow at intermediate node u . Let $|\Phi|$ be the total flow value stored at sink as well as intermediate nodes within the given time horizon. The mathematical model for the maximum dynamic flow problem by allowing the intermediate storage is a linear programming formulation which can be defined as follows.

$$\max |\Phi| \quad (2.11a)$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \sum_{\theta=0}^T \Phi_a(\theta) = |\Phi| = \sum_{a \in \Gamma_t^{in}} \sum_{\theta=\tau_a}^T \Phi_a(\theta - \tau_a) + \sum_{u \in \mathcal{I}} \hat{\Phi}_u(T) \quad (2.11b)$$

$$\sum_{a \in \Gamma^{in}(u)} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta - \tau_a) - \sum_{a \in \Gamma^{out}(u)} \sum_{\beta=0}^{\theta} \Phi_a(\beta) = \hat{\Phi}_u(\theta), \quad u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.11c)$$

$$0 \leq \Phi_a(\theta) \leq \kappa_a, \quad \forall a \in \mathcal{A}, \theta \in \mathcal{T} \quad (2.11d)$$

$$0 \leq \hat{\Phi}_u(\theta) \leq \nu_u, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.11e)$$

Equation (2.11a) is an objective function that intends to maximization of the dynamic flow value. Equation (2.11b) represents that the dynamic flow with intermediate storage is the total flow out from the source that must be equal to the sum of inflow at sink and the total excess flow at intermediate nodes within the given time horizon T . The excess flow stored at intermediate nodes is represented by equation (2.11c) where the constraints in (2.11d) and (2.11e) represent the capacity constraints of arcs and nodes, respectively.

Solution Procedure. As in static case, fixing the priority of nodes and constructing modified network Π^* are similar for the dynamic flow with intermediate storage. The lexicographically maximum dynamic flow is computed in Π^* by using algorithm of Hoppe & Tardos (2000). The transformation of the original network Π by removing the dummy nodes and the dummy arcs, and shifting the flow of dummy node to their respective original nodes are also same as in static case. Solution obtained in this procedure provides an optimal flow with intermediate storage which can be obtained in polynomial time complexity (Pyakurel & Demepe (2020)).

2.3.3 Temporally Repeated MDF with Intermediate Storage

The TRF at sink of a two-terminal network using $s - t$ paths was first introduced by Ford & Fulkerson (1962) to solve the maximum dynamic flow problem in general network, where intermediate storage of the flow was prohibited. In previous Subsection 2.3.2, we discussed the dynamic flow with intermediate storage in which solution procedure depends on the lexicographic approach. This subsection is mainly focused on an alternative approach of finding maximum dynamic flow with intermediate storage, known as temporally repeated approach. This technique was first introduced in the preprint of Khanal et al. (2022) for general network flow and Pyakurel et al. (2022) for the abstract network (c.f. Chapter 3 for abstract flow).

Here, to solve the MDF problem with intermediate storage in general network by using TRF, we proceed the solution strategy in three stages, (Khanal et al. (2022)).

- (a) Fix the priority order of nodes.
- (b) Calculate the excess flow and construct the flow balancing paths.
- (c) Use temporally repeated flow (TRF) at each node.

A necessary condition for the existence of a solution to MDF with intermediate storage is that the storage capacity of intermediate nodes must be T times the sum of capacity of incoming arcs, i.e., $\nu_u \geq T \sum_{a \in \Gamma_u^{in}} \kappa_a$, $\forall u \in \mathcal{I}$ because the flow may have to be held at intermediate nodes for T time horizon. Any intermediate node which has no aforementioned capacity is considered as a part of the path segment.

(a) Priority Order of Nodes. To solve the MDF problem with intermediate storage, we begin the procedure by fixing the priority order of nodes. As we assume the problem instance of disaster management, the sink is most appropriate place to send the flow and so the first priority is given to the sink to transship as much flow as possible. The storage of excess flow at the intermediate nodes with priority order is fixed as follows. For each $u \in \mathcal{I}$ with storage capacity $\nu_u \geq T \sum_{a \in \Gamma_u^{in}} \kappa_a$, let $d_{P_{[s \rightarrow u]}}$ be the shortest distance of node u from the source s obtained by using algorithm of Dijkstra (1959). Set the priority order with respect to the distance so that farthest intermediate node has highest priority, i.e., if $d_{P_{[s \rightarrow u_1]}} > d_{P_{[s \rightarrow u_2]}}$ for $u_1, u_2 \in \mathcal{I}$, then u_1 has higher priority than u_2 and is denoted by $u_1 \succ u_2$. The equality in distance can have mutability in priority order. If the priority order of $n-2$ intermediate nodes be $u_1 \succ u_2 \succ \dots \succ u_{n-2}$, then priority order including sink is $M = \{t \succ u_1 \succ u_2 \succ \dots \succ u_{n-2}\} = \mathcal{N} \setminus \{s\}$.

(b) Excess Flow and Flow Balancing Path Decomposition. While shipment of the flow in a network, flow stored at prioritized nodes due to the capacity constraint or the time constraint is known as excess flow. If the incoming flow at a node is more than the outgoing flow, then the amount of flow held at the node is an excess flow due to the capacity constraint. Similarly in case of dynamic flow, if the flow is unable to reach the successor node of a path from the current

(predecessor) node due to insufficient transit time, then the flow held at the current node is an excess flow due to the time constraint.

For the TRF with intermediate storage, we first have to find the static excess flow λ_u at each prioritized node $u \in M$ that is caused by capacity constraints in the static network topology. Next, we send the maximum static flow $\lambda_t = |\phi|_t$ at sink by using min-cost max-flow and then decompose the flow on $s - t$ paths. We then obtain the residual capacity $(\kappa_a - \phi_a)$ on each arc. Afterwards, we pick the next prioritized node u_1 , calculate the static flow at u_1 via $s - u_1$ paths on residual network, denote it by λ_{u_1} and update the residual capacity as in previous step. Similar procedure is used to find the static excess flow at each intermediate node $\lambda_u = |\hat{\phi}|_u$ with respective priority order.

As a node may lay in more than one path, we indicate the excess flow of node at the decomposed paths taken in successive shortest order by balancing the flow on arcs and nodes. A path P obtained after path decomposition of min-cost max-flow is a ‘flow balancing path with excess flow’ if it satisfies (i) outflow from source along the path P is equal to sum of flows at sink and intermediate nodes and (ii) inflow at each intermediate node of path P is equal the sum of outflow from the node and excess flow hold at the node. The flow balancing paths with excess flow at nodes can be obtained by Algorithm 2 as follows.

Algorithm 2: Flow balancing paths with excess flow.

Input : Given a dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$.

Output: Flow balancing paths with excess flow at nodes.

1. Fix the first priority to the sink t and the priority order of intermediate nodes $u \in \mathcal{I}$ with farther in distance from the source higher in priority.
 2. Calculate the min-cost max-flow on static network by considering transit time as cost. Indicate the flow as excess of sink λ_t . Decompose the arc flows to $s - t$ path flow. Also, indicate the residual capacity on arcs.
 3. Calculate the static flow at intermediate nodes by min-cost max-flow and decompose the intermediate paths $P_{[s \rightarrow u]}$ from the residual network, as in Step 2 with successive priority order of nodes $u \in \mathcal{I}$. Indicate it as excess flow λ_u on the original network.
 4. Obtain the ‘flow balancing paths with excess flow’ as follows.
 - (a) On the decomposed $s - t$ path, balance the flow on path from source to the sink without violating the capacity constraints on arcs and excess flow at intermediate nodes so that outflow from source along the path is equal to sum of flows at sink and intermediate nodes.
 - (b) Update the network by reducing the used capacity on arcs and the excess flow on the nodes obtained in Step 4(a).
 - (c) Continue the process as long as there exist a decomposed $s - t$ path of positive flow in Step 2 as well as intermediate path in Step 3 with priority order of nodes.
-

Edmonds & Karp (1972) introduced the scaling technique by modifying successive shortest path algorithm to obtain the capacity scaling algorithm, which is the first polynomial time algorithm for minimum cost flow problem. For min-cost max-flow calculation, we use this scaling

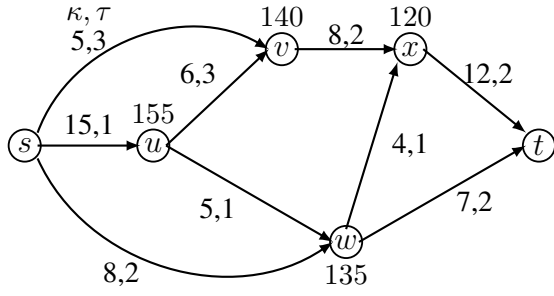
algorithm by considering the transit time as cost so that the algorithm computes the solution in polynomial time. Again, each path must transship at least one unit of flow from the source, so number of flow balancing paths in Algorithm 2 is bounded by the sum of the capacity of outgoing arcs from the source, i.e., $|\mathcal{P}| \leq \sum_{a \in \Gamma_s^{out}} \kappa_a$.

Theorem 2.2. *Algorithm 2 provides the optimal static excess flow at each prioritized node using decomposed paths.*

Proof. We consider the transit time on arcs as distance/cost to calculate the distance of each node from the source. Static flow at sink can be obtained by using min-cost max-flow algorithm optimally, where maximum static flow is calculated at first and minimum cost paths are obtained to satisfy the demand of the maximum flow. This flow is indicated as the excess flow of the sink λ_t . By obtaining the residual capacity of arcs in each iteration, we calculate the min-cost max-flow to the prioritized intermediate nodes in the residual network and indicate it as an excess flow λ_u . Thus the static excess flow obtained in each node is feasible as well as optimal. Next, we decompose the flow on paths with intermediate storage by using successive shortest paths (i.e., minimum cost paths) as follows: We take a $s - t$ path (shortest among all $s - t$ paths) obtained while finding the excess flow and balance the flow from sink to the source with intermediate storage without violating the capacity constraints in such a way that the sum of inflow and excess flow equal the outflow. By updating the flow on arcs and nodes, balancing process is continued as long as there is any successive shortest path with positive flow. Due to updated residual capacity obtained by reducing the used capacity on arcs as well as the excess flow at nodes in each iteration, algorithm provides the feasible flow at each node of the flow balancing path. It is important to emphasize that the flow stored at an intermediate node is permanently stored at the node. \square

Example 2.1. Consider a dynamic network with capacity and transit time on each arc as shown in Figure 2.3, where storage capacity of nodes are $\nu_u = 155$, $\nu_v = 140$, $\nu_w = 135$, $\nu_x = 120$ and the sink has sufficient capacity, say $\nu_t = \infty$. The shortest distance of intermediate nodes from the source are $d_{P_{[s \rightarrow v]}} = 3$, $d_{P_{[s \rightarrow x]}} = 3$, $d_{P_{[s \rightarrow w]}} = 2$, $d_{P_{[s \rightarrow u]}} = 1$. As the first priority is given to the sink t , the priority order nodes for the given network is $t \succ v \succ x \succ w \succ u$. Five $s - t$ paths obtained by min-cost max-flow algorithm are P_1, P_2, P_3, P_4 and P_5 (see Figure 2.3) with excess $\lambda_t = 19$. In the network with residual capacity, the next priority is for node v with a single shortest path, excess flow on the residual network to this node is 3 units. Continuing the process for each prioritized nodes, the excess flow at x, w and u are 0, 2 and 4 units, respectively. The network with excess flow at each node is presented in Figure 2.4.

As our aim is to find the dynamic flow with intermediate storage using TRF, the static path flow with their excess at nodes is essential. As an intermediate node lies in more than one paths, we fix the excess of that node for the successive shortest decomposed paths obtained in Step 4 of



Priority Order : $t \succ v \succ x \succ w \succ u$.

Successive shortest (min-cost max-flow) $s - t$ paths:

$P_1 : s - w - t$, $\tau_1 = 4$, flow=7.

$P_2 : s - w - x - t$, $\tau_2 = 5$, flow=1.

$P_3 : s - u - w - x - t$, $\tau_3 = 5$, flow=3.

$P_4 : s - v - x - t$, $\tau_4 = 7$, flow=5.

$P_5 : s - u - v - x - t$, $\tau_5 = 8$, flow=3.

Prioritized intermediate paths with positive flow:

Intermediate path : $s - u - v$, flow=3

Intermediate path : $s - u - w$, flow=2

Intermediate path : $s - u$, flow=4

Figure 2.3: Given network with arc capacity, transit time and node capacity.

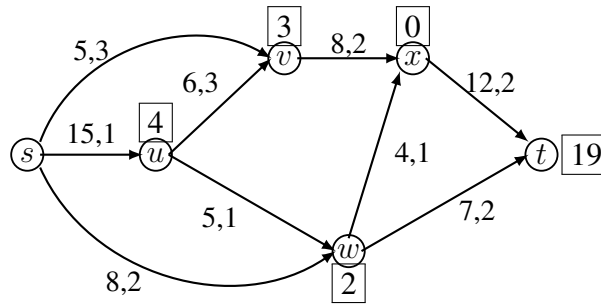
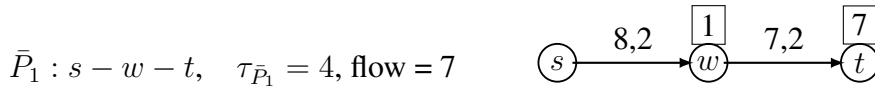


Figure 2.4: Network with excess flow λ at each node.

Algorithm 2 without violating the capacity constraints on arcs as follows.

Flow Balancing Path Decomposition of Static Flow with Excess Flow λ at Nodes. As P_1 is the shortest $s - t$ path obtained after path decomposition, so we start to balance the flow on this path so that 7 units of flow reaches to t with rename it as $\bar{P}_1 : s - w - t$ where, flow of 8 units started from s reaches only 7 units to t by holding 1 unit at w .



Flow balancing path \bar{P}_1 with excess flow λ

Now, the arc capacity and excess along this path of Figure 2.4 are updated by reducing the used capacity at arcs and nodes to obtain Figure 2.5.

As capacity of arc (s, w) is saturated, path P_2 is not used in flow balancing path. We select the next shortest decomposed path $\bar{P}_2 : s - u - w - x - t$ with $\tau_{\bar{P}_2} = 5$ and path flow value 4 at sink. Then, we balance the flow at nodes and arcs. The flow balancing path and updated network obtained from this step is presented in Figure 2.6.

Continuing this process, we select the next shortest decomposed path $\bar{P}_3 : s - v - x - t$ with $\tau_{\bar{P}_3} = 7$ and path flow value 5 at sink. Then, the flow is balanced at nodes and arcs. The flow

Updated network with excess flow:

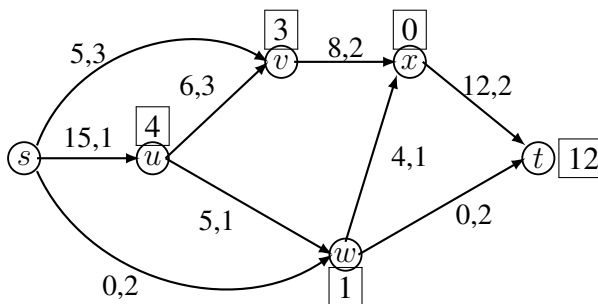
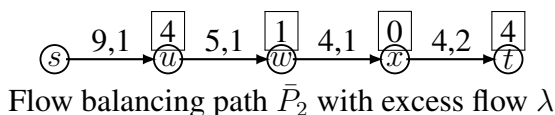


Figure 2.5: Updated network after balancing the flow on path \bar{P}_1 .

$\bar{P}_2 : s - u - w - x - t, \tau_{\bar{P}_2} = 5, \text{ flow} = 4$



Flow balancing path \bar{P}_2 with excess flow λ

Updated network with excess flow:

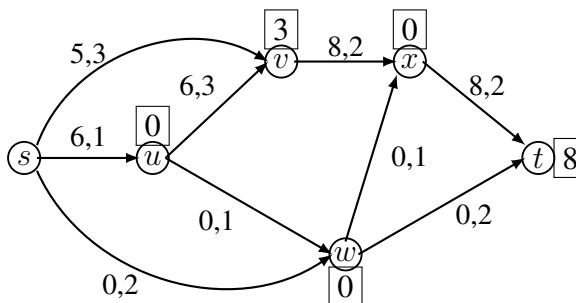
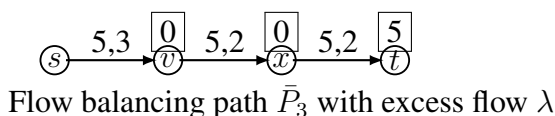


Figure 2.6: Updated network after balancing the flow on path \bar{P}_2 .

balancing path and updated network are obtained as presented in Figure 2.7.

$\bar{P}_3 : s - v - x - t, \tau_{\bar{P}_3} = 7, \text{ flow} = 5$



Flow balancing path \bar{P}_3 with excess flow λ

Updated network with excess flow:

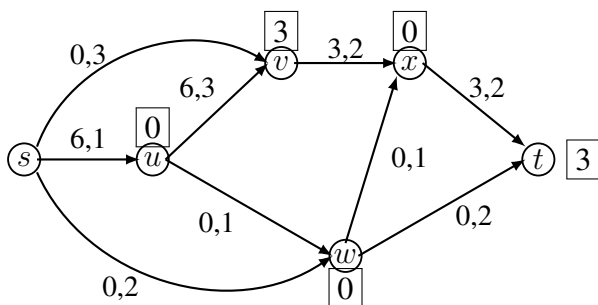


Figure 2.7: Updated network after balancing the flow on path \bar{P}_3 .

Finally decomposed path $\bar{P}_4 : s - u - v - x - t$ with $\tau_{\bar{P}_4} = 8$ and path flow value 3 at sink are obtained in Figure 2.8 as follows.

$\bar{P}_4 : s - u - v - x - t$, $\tau_{\bar{P}_4} = 8$, flow = 3

Flow balancing path \bar{P}_4 with excess flow λ

Updated network with excess flow:

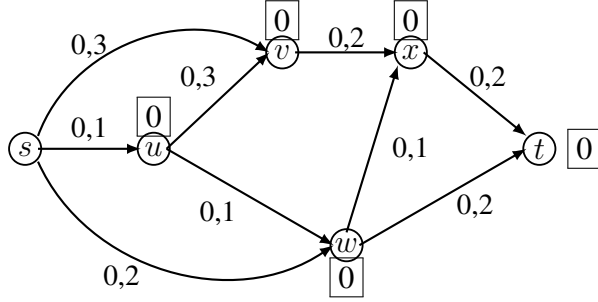


Figure 2.8: Updated network after balancing the flow on path \bar{P}_4 .

As no $s - t$ paths and intermediate paths with positive flow exist, so the process is terminated with four flow balancing paths \bar{P}_1 , \bar{P}_2 , \bar{P}_3 and \bar{P}_4 . Hereafter, we derive the formula for TRF with intermediate storage and apply it on these four paths to obtain the maximum flow with intermediate storage. \square

(c) Temporally Repeated Flow (TRF) with Intermediate Storage. Ford and Fulkerson are the pioneers of network flow who have introduced the TRF at sink using source-sink paths with fixed transit times by repeating the constant rate of flow on paths to provide the optimal flow at sink. We are concerning with the maximization of flow out from the source by holding the excess flow at intermediate nodes which can not reach to the sink by using temporal repetition. To deal with this problem, we present the TRF that transship the maximum flow to the sink as well as to the prioritized intermediate nodes simultaneously for the general network topology.

The TRF at sink t or the last node of intermediate path with respect to the flow balancing path \bar{P} within time horizon T is,

$$|\Phi|_t = \sum_{\bar{P} \in \mathcal{P}} (T - \tau_{\bar{P}} + 1) \cdot \phi_{\bar{P}} \quad (2.12a)$$

Similarly, if u_i and u_j are any two successive nodes in flow balancing path \bar{P} with $\bar{P}_{[s \rightarrow u_i]} \subseteq \bar{P}_{[s \rightarrow u_j]}$, the TRF at u_i is,

$$|\Phi|_{u_i} = \sum_{\bar{P}: \bar{P}_{[s \rightarrow u_i]} \subseteq \bar{P}} \left[(T - \tau_{\bar{P}_{[s \rightarrow u_j]}} + 1) \cdot \lambda_{u_i} + (\tau_{\bar{P}_{[s \rightarrow u_j]}} - \tau_{\bar{P}_{[s \rightarrow u_i]}}) \cdot \phi_{\bar{P}_{[s \rightarrow u_i]}} \right] \quad (2.12b)$$

where, λ_{u_i} represents the excess flow at u_i and $\phi_{\bar{P}_{[s \rightarrow u_i]}}$ is the static path flow from s to the node u_i along the flow balancing path \bar{P} . Equation (2.12b) shows that the storage of the flow at intermediate nodes is the sum of two flows, excess flow λ_{u_i} at node u_i due to non-conservation of the flow on balancing paths and the path flow up to the node u_i which can not move forward

due to insufficient time.

The generalize equation (2.12a) in term of equation (2.12b) is as follows. Replace the node u_i in equation (2.12b) by t and its predecessor node u_j along the path as t itself (as sink has no any successor). Again for the sink node, excess flow λ_t is considered as the source-sink path flow $\phi_{\bar{P}}$. With these arguments, equation (2.12b) reduce to equation (2.12a).

Theorem 2.3. *The TRF in equations (2.12a) and (2.12b) provides the optimal solution to the MDF with intermediate storage in polynomial time.*

Proof. As proven in Theorem 2.2, Algorithm 2 provides the optimal excess flow using min-cost max-flow solution on each prioritized node. In equations (2.12a) and (2.12b), we use the flow balancing path with excess flow λ obtained by using Algorithm 2. As in Ford & Fulkerson (1956), equation (2.12a) provides the optimal flow at sink. Similarly, each prioritized intermediate node has storage capacity at least T times the sum of incoming arc capacities and can holds as much excess flow as possible within the time horizon by using equation (2.12b). Intermediate node holds excess flow due to capacity constraint as long as the flow can move to the successor node and if the flow is unable to move to the successor node due to time constraint, then it holds full path flow up to the node. It is to be noted that the excess flow due to capacity constraint stored at intermediate node is not possible to proceed toward the sink because at every moment of time, inflow at the intermediate node is greater than or equals to the outflow and the excess flow is accumulated at the node within the time horizon. The flow out from the source with saturating arc capacity within time horizon are either reaching to the sink or to the prioritized intermediate nodes, the TRF obtained by equations (2.12a) and (2.12b) are optimal.

The complexity of maximum dynamic flow solution with intermediate storage depends on the complexity of Algorithm 2 because the complexity of TRF computed on the nodes using equations (2.12a) and (2.12b) is $O(n)$. The time complexity of Dijkstra's algorithm is $O(n^2)$. Similarly, maximum flow can be found by shortest augmenting path algorithm in $O(nm \log(U))$ time and generic cost-scaling algorithm solves the minimum cost flow in $O(n^2m \log(nc'))$ time, where, $U = \max\{\kappa_a : a \in \mathcal{A}\}$ and c' is the maximum non-negative cost (Ahuja et al. (1993)). Thus, the MDF with intermediate storage can be solved using TRF in polynomial time. \square

Example 2.2. On Figure 4.1 of Example 2.1, consider the time horizon $T = 8$. By using the TRF with intermediate storage on the set of flow balancing paths $\{\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4\}$ with excess flows at the paths. As the priority order of nodes is $t \succ v \succ x \succ w \succ u$, we calculate flow values as per the order of nodes as follows.

$$\begin{aligned} |\Phi|_t &= \sum_{i=1}^4 (T - \tau_{\bar{P}_i} + 1) \cdot \phi_{\bar{P}_i} = 64 \quad (\because \text{Node } t \text{ is last node in all four Paths}) \\ |\Phi|_v &= [(T - \tau_{\bar{P}_3[s \rightarrow x]} + 1) \cdot \lambda_{v(\bar{P}_3)} + (\tau_{\bar{P}_3[s \rightarrow x]} - \tau_{\bar{P}_3[s \rightarrow v]}) \cdot \phi_{\bar{P}_3[s \rightarrow v]}] + \end{aligned}$$

$$\begin{aligned}
& [(T - \tau_{\bar{P}_4[s \rightarrow x]} + 1) \cdot \lambda_v(\bar{P}_4) + (\tau_{\bar{P}_4[s \rightarrow x]} - \tau_{\bar{P}_4[s \rightarrow v]}) \cdot \phi_{\bar{P}_4[s \rightarrow v]}] \\
= & 31 \quad (\because \text{Node } v \text{ lies in Paths } \bar{P}_3 \text{ and } \bar{P}_4) \\
|\Phi|_x = & [(T - \tau_{\bar{P}_2[s \rightarrow t]} + 1) \cdot \lambda_x(\bar{P}_2) + (\tau_{\bar{P}_2[s \rightarrow t]} - \tau_{\bar{P}_2[s \rightarrow x]}) \cdot \phi_{\bar{P}_2[s \rightarrow x]}] + \\
& [(T - \tau_{\bar{P}_3[s \rightarrow t]} + 1) \cdot \lambda_x(\bar{P}_3) + (\tau_{\bar{P}_3[s \rightarrow t]} - \tau_{\bar{P}_3[s \rightarrow x]}) \cdot \phi_{\bar{P}_3[s \rightarrow x]}] + \\
& [(T - \tau_{\bar{P}_4[s \rightarrow t]} + 1) \cdot \lambda_x(\bar{P}_4) + (\tau_{\bar{P}_4[s \rightarrow t]} - \tau_{\bar{P}_4[s \rightarrow x]}) \cdot \phi_{\bar{P}_4[s \rightarrow x]}] \\
= & 24 \quad (\because \text{Node } x \text{ lies in Paths } \bar{P}_2, \bar{P}_3 \text{ and } \bar{P}_4) \\
|\Phi|_w = & [(T - \tau_{\bar{P}_1[s \rightarrow t]} + 1) \cdot \lambda_w(\bar{P}_1) + (\tau_{\bar{P}_1[s \rightarrow t]} - \tau_{\bar{P}_1[s \rightarrow w]}) \cdot \phi_{\bar{P}_1[s \rightarrow w]}] + \\
& [(T - \tau_{\bar{P}_2[s \rightarrow x]} + 1) \cdot \lambda_w(\bar{P}_2) + (\tau_{\bar{P}_2[s \rightarrow x]} - \tau_{\bar{P}_2[s \rightarrow w]}) \cdot \phi_{\bar{P}_2[s \rightarrow w]}] \\
= & 32 \quad (\because \text{Node } w \text{ lies in Paths } \bar{P}_1, \text{ and } \bar{P}_2) \\
|\Phi|_u = & [(T - \tau_{\bar{P}_2[s \rightarrow w]} + 1) \cdot \lambda_u(\bar{P}_2) + (\tau_{\bar{P}_2[s \rightarrow w]} - \tau_{\bar{P}_2[s \rightarrow u]}) \cdot \phi_{\bar{P}_2[s \rightarrow u]}] + \\
& [(T - \tau_{\bar{P}_4[s \rightarrow v]} + 1) \cdot \lambda_u(\bar{P}_4) + (\tau_{\bar{P}_4[s \rightarrow v]} - \tau_{\bar{P}_4[s \rightarrow u]}) \cdot \phi_{\bar{P}_4[s \rightarrow u]}] \\
= & 55 \quad (\because \text{Node } u \text{ lies in Paths } \bar{P}_2, \text{ and } \bar{P}_4)
\end{aligned}$$

Here, the notation $\lambda_v(\bar{P}_3)$ stands for the excess flow at node v with respect to the flow balancing path \bar{P}_3 and so on. The MDF with intermediate storage that is transshipped from source within time horizon $T = 8$ is 206 units and the flow is stored at different nodes with priority order are as follows: flow at first priority node t is 64, flow at second priority nodes v and x are $(31 + 24) = 55$, flow at third priority node w is 32 and flow at fourth priority node u is 55.

It is to be noted that if we choose flow balancing path such as $\bar{P}_1 : s - u - w - t$ with $\tau_1 = 4$ and flow = 5, $\bar{P}_2 : s - w - t$ with $\tau_2 = 4$ and flow = 2, $\bar{P}_3 : s - w - x - t$ with $\tau_3 = 5$ and flow = 4, $\bar{P}_4 : s - v - x - t$ with $\tau_4 = 7$ and flow = 5, and $\bar{P}_5 : s - u - v - x - t$ with $\tau_5 = 8$ and flow = 3, then we get exactly the same maximum dynamic flow with intermediate storage. \square

Case Illustration I

Problem Description. Disasters are unexpected evidences that may be natural like land slide, flooding, earthquake, heat or cold waves, fire, glacier, etc. and may also be caused by human errors like terrorism, nuclear explosions, etc. Nepal is more likely to encounter natural disaster every year. A case of a disaster can also be caused by a mere rumor of bombing. Suppose that, for the sake of an illustration, there is a spiritual mass gathering in a surrounding of the greatest holly temple of Hindus at Gausala area. This occasion is participated in by more than 40,000 people coming from different parts of the country. A group of young people in this event are also gathered to celebrate the festival by fireworks. Unfortunately, loud explosions caused by these fireworks are thought of by some people as as occurrence of bombing. The rumor of bombing spreads swiftly thereby creating panic among the people and quick movement around the place of the event. Instantly and expectedly, traffic officials will try to fix the route of evacuation to minimize the congestion due to crossing and auto-based evacuation is started on the located



Figure 2.9: Evacuation zone with 69 nodes. Nodes 0 (red) and 68 (blue) represent the source and sink, respectively, whereas green areas are intermediate shelters. (Source: <https://www.greattibettour.com/nepal-tours/maps-of-kathmandu.html>)

routes (see Figure 2.9).

To capture the mathematical optimization model with intermediate storage using network optimization, we have created 69 nodes (numbered from 0 to 68), where nodes 0 and 68 are the source (danger zone) and sink (safe shelter), respectively. The objective of the evacuation model is to send maximum amount of flow from danger zone (Gausala area - 0) to safe shelter (Tribhuvan University area - 68) by holding the excess flow at intermediate shelters (Shankha park - 1, Tinkune - 10, BICC Building - 11, Tudikheh - 20, Balaju Industrial Area - 31, Soyambhunath - 32, Chyasal Stadium - 46, UN Park - 48, Institute of Engineering - 49 and ANFA Complex - 51). Rest of the nodes are considered as a part of path segments with no storage of flow. The paths joining nodes are the road segments, considered as arcs. The detailed list of arcs, number of lanes in each arc and their transit times are presented in Appendix A where number of lanes are considered as per the width and transit times as per the length and width of the roads.

Output. We have implemented Algorithm 2 to obtain the flow balancing paths and the temporally repeated formula obtained in equations (2.12a) and (2.12b) are used to find the maximum dynamic flow with intermediate storage. For this, we developed the programming codes in Python of version 3.7 on Dell computer with 64-bit operating system having 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz & 2.42 GHz dual processor and 8 GB RAM. For the network presented in Figure 2.9 with 69 nodes and 135 arcs, average of 7 vehicles per minute are

considered to pass away in each lane of the road. Thus, by natural transformation of continuous flow to discrete time settings, the capacity per minute of an arc is 7 times the number of lanes on the arc. Some pair of nodes contain parallel arcs with same transit times, for example, two arcs joining the node 0 to node 3 with number of lanes 2 in one and 1 in another but have the same transit time of 1 minute. So, in the table we have indicated the number of lanes in arc 0-3 as 2+1. The priority order of shelters is obtained from programming output as $\{68 \succ 51 \succ 32 \succ 31 \succ 49 \succ 46 \succ 48 \succ 1 \succ 10 \succ 11 \succ 20\}$. We set the time of evacuation as $T = 4$ hours = 240 minutes. The number of evacuees shifted to respective priority order of shelters in this time horizon are 24094, 6379, 3178, 147, 27, 47, 31, 284, 197, 58 and 171. Hence, the total of 34,613 evacuees are evacuated from the source in 4 hours. The running time of the program is 0.149 seconds.

2.3.4 Temporally Repeated EAF with Intermediate Storage

In this subsection, our discussion is on earliest arrival flow (EAF) problem with intermediate storage with an objective of transshipping the maximum possible flow from the source to sink by holding the excess flow simultaneously at prioritized intermediate nodes not only at the time horizon $\theta = T$ but in each time step $\theta \in \mathcal{T}$.

The objective function for EAF problem with intermediate storage is,

$$\max |\Phi(\theta)| = \sum_{a \in \Gamma_{out}^g} \sum_{\beta=0}^{\theta} \Phi_a(\beta) = \sum_{a \in \Gamma_t^{in}} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta - \tau_a) + \sum_{u \in \mathcal{I}} \Phi_u(\theta), \quad \forall \theta \in \mathcal{T} \quad (2.13)$$

and the flow conservation and capacity constraints are as in (2.11c–2.11e).

Here, equation (2.13) refers to maximization of the flow not only at $\theta = T$ but in every time step $\theta \in \mathcal{T} = \{0, 1, \dots, T\}$. Thus, it is a multi-objective optimization problem. Some of the solution methods for solving multi-objective optimization problem are global criterion method, utility function method, inverted utility method, bounded objective function method, lexicographic model and global programming method (Chapter 9 of Nayak (2020)). To solve multi-terminal maximum flow problem, Miniéka (1973) used the lexicographic approach by forming the set inclusion with respect to the priority order of terminals. The solution strategy of lexicographic multi-objective optimization is to maximize the objective of T^{th} time step by fixing all earlier time (i.e., $\theta = 0, 1, \dots, T-1$) objective functions to their optimal function value with successive order of time steps and is optimal for time $\theta = T$.

Due to the use of non-standard chain decomposition in EAF computation, Wilkinson (1971) has shown that it is not possible to use TRF for the general network with flow conservation constraints where the storage of flow at intermediate nodes is not considered. Here, we present Example 2.3 to show that the MDF with intermediate storage for general network obtained in

Subsection 2.3.3 by using TRF is not an EAF.

Example 2.3. Consider a network presented in Figure 2.10, where the priority order of nodes with respect to the distance (cost) is $t \succ x \succ y$. Let the storage capacity of both intermediate nodes be 200 and time horizon be $T = 8$. Considering transit time as cost, the first shortest (min-cost) path $s - y - x - t$ sends 2 units of flow at t and then path $s - x - y - t$ in residual network sends 2 more flow units at t . By canceling the cycle at arc (y, x) , flows are decomposed in to two paths $s - y - t$ and $s - x - t$. Similarly, excess flows at x and y are 8 and 0 units, respectively.

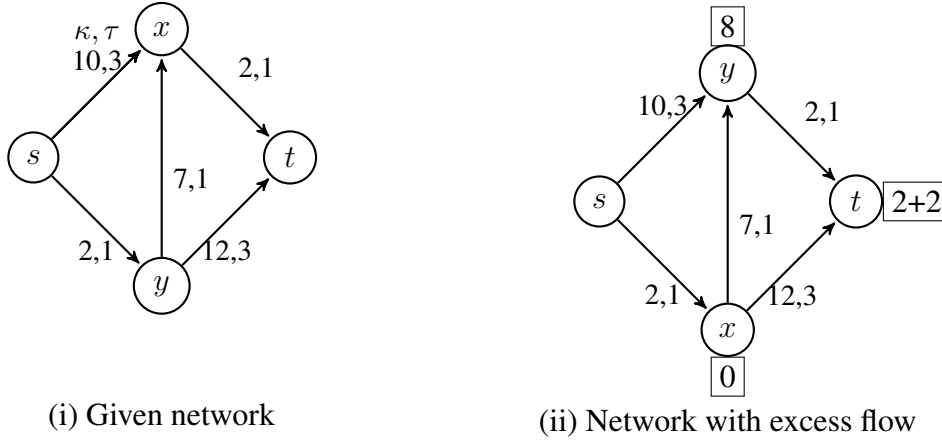


Figure 2.10: A general non-series-parallel network and its excess flow computation at nodes.

Now, to obtain the flow balancing path, if the shortest path $s - y - x - t$ is used, then arcs (s, y) and (x, t) will be saturated with 2 units of flow at t and no $s - t$ path exists for the remaining 2 units of flow. Thus as in Subsection 2.3.3, instead of using all successive shortest paths, use of decomposed paths (min-cost max-flow paths) is essential. As the TRF is used on the flow balancing paths which does not use the shortest path $s - y - x - t$ of length 3 units, it can not provide the maximum flow at time $T = 3$, and so it is not an EAF. \square

Series-parallel graphs are proper subset of acyclic digraphs defined as follows. Graph with single arc is a series-parallel graph. Let Π_1 and Π_2 be two series-parallel graphs with sources s_1 and s_2 , and sinks t_1 and t_2 , respectively. The series composition $\Pi(\Pi_1, \Pi_2)$ obtained by identifying the sink t_1 as the source s_2 is a series-parallel graph with source s_1 and sink t_2 . Similarly, the parallel composition $\Pi(\Pi_1, \Pi_2)$ obtained by merging the source s_1 with s_2 and sink t_1 with t_2 is also a series-parallel graph with source $s_1(= s_2)$ and sink $t_1(= t_2)$.

The existence of a the TRF on two-terminal series-parallel graph having the earliest arrival property was proven by Ruzika et al. (2011) and have presented a polynomial time solution strategy. As their problem does not concerns with the storage of flow at intermediate nodes, we are here to address the temporally repeated EAF not only at the sink but also at the intermediate nodes. As it is not possible to find EAF using TRF in non-series-parallel network, we adopt the series-parallel graph as in Ruzika et al. (2011) and use the procedure as described in Sub-

section 2.3.3. The algorithmic framework for the solution of EAF Problem with intermediate storage by using TRF is present in Algorithm 3.

Algorithm 3: EAF algorithm with intermediate storage using TRF.

Input : Given a two-terminal series-parallel network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$.

Output: EAF with intermediate storage.

1. Fix the first priority to the sink t and the priority order of intermediate nodes $u \in \mathcal{I}$ with farther in distance from the source higher in priority.
 2. Use Algorithm 2 to find the excess flow and flow balancing paths.
 3. Obtain the MDF at sink and intermediate nodes by temporal repetition of flow using equations (2.12a) and (2.12b).
 4. MDF so obtained is an EAF with intermediate storage.
-

Due to the series-parallel graph, no any path forms the cycle and each successive shortest path obtained by Step 5 of Algorithm 2 is a successive shortest decomposed path in Algorithm 3. Thus, MDF obtained in Algorithm 3 is an EAF with intermediate storage for series-parallel graph, which can be obtained in polynomial time complexity.

Theorem 2.4. *For the series-parallel graph, Algorithm 3 provides an optimal solution to EAF with intermediate storage using TRF in polynomial time.*

Example 2.4. Consider a series-parallel network presented in Figure 2.11, where the priority order of nodes is $t \succ v \succ x \succ w \succ u$ and nodes v and x are mutable as they have same shortest distance. The excess flow of t, v, x, w and u are 16, 6, 1, 2 and 0, respectively. Let the storage capacity of each node be 150 and time horizon be $T = 8$.

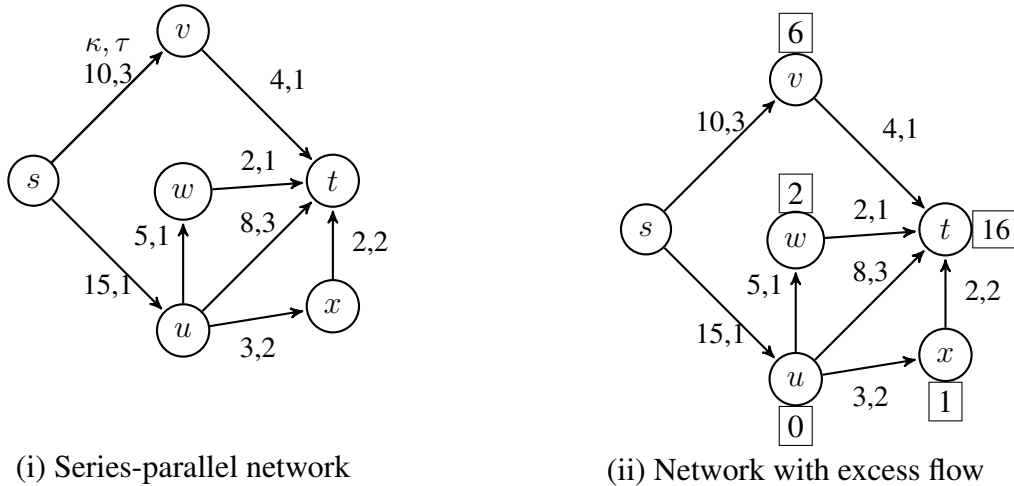


Figure 2.11: A series-parallel network and its excess flow computation at nodes.

In Figure 2.11, there are four successive shortest paths $s - u - w - t$, $s - u - t$, $s - v - t$ and $s - u - x - t$ of min-cost max-flow computation, which are also the flow balancing paths. Flow balancing paths with excess flow at nodes are presented in Figure 2.12.

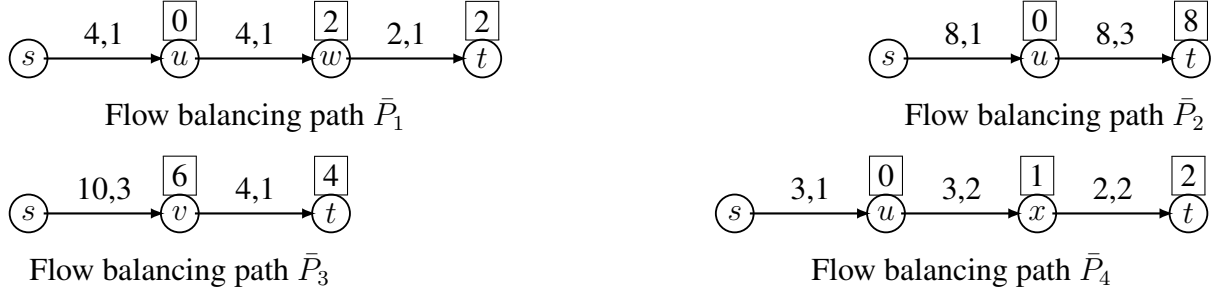


Figure 2.12: Flow balancing paths of series-parallel network in Figure 2.11.

By using equations (2.12a) and (2.12b), total flow reaching to t , v , x , w and u along these paths within $T = 8$ are 80, 40, 10, 16 and 34 units respectively, satisfying the earliest arrival property. Thus, total amount of earliest arrival flow transmission in $T = 8$ is 180 units. \square

2.4 Contraflow

A general network topology in which pair of oppositely directed arcs (anti-parallel arcs) exist in between some pair of nodes is known as two-way network topology. On solving the network flow problems in such network, a very well known technique, known as contraflow configuration, is used to increase the outbound capacity of arcs by the reversal of opposite orientated (backward) arcs towards the destination.

Consider a two-way static network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, s, t)$ in which anti-parallel arcs $a = (u, v)$ and $\overleftarrow{a} = (v, u)$ have respective capacity κ_a and $\kappa_{\overleftarrow{a}}$. To solve the static contraflow problem by lane reversal strategy, the auxiliary network of given two-way network Π , denoted by $\overline{\Pi} = (\mathcal{N}, \overline{\mathcal{A}}, \overline{\kappa}, s, t)$, is constructed where $\overline{\mathcal{A}}$ contains the undirected edges $\overline{\mathcal{A}} = \{(u, v) : (u, v) \text{ or } (v, u) \in \mathcal{A}\}$. The capacity $\overline{\kappa}$ of an arc in the auxiliary network is the sum of capacities of anti-parallel arcs a and \overleftarrow{a} , i.e., $\overline{\kappa}_a = \kappa_a + \kappa_{\overleftarrow{a}}$, where $\kappa_a = 0$ if $a \notin \mathcal{A}$. Based on this graph transformation, Rebennack et al. (2010) presented an algorithm to solve the maximum static contraflow (MSCF) problem in polynomial time complexity.

Static Flow Model. The linear programming model for maximum static contraflow problem can be presented as

$$\max |\phi| \tag{2.14a}$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \phi_a = |\phi| = \sum_{a \in \Gamma_t^{in}} \phi_a \tag{2.14b}$$

$$\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a = 0, \quad \forall u \in \mathcal{I} \quad (2.14c)$$

$$0 \leq \phi_a \leq \bar{\kappa}_a, \quad \forall a \in \bar{\mathcal{A}} \quad (2.14d)$$

Here, equation (2.14d) reflects the capacity bound of the flow after contraflow configuration. Similarly, the static contraflow model with intermediate storage is,

$$\max |\phi| \quad (2.15a)$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \phi_a = |\phi| = \sum_{a \in \Gamma_t^{in}} \phi_a + \sum_{u \in \mathcal{I}} \hat{\phi}_u \quad (2.15b)$$

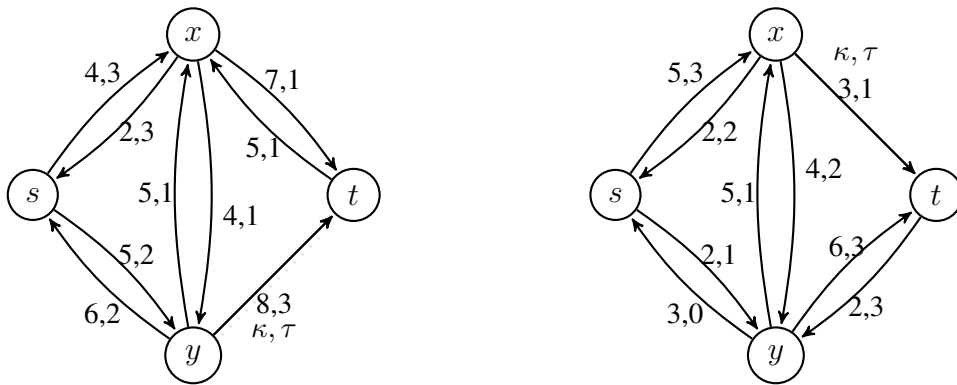
$$\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a = \hat{\phi}_u, \quad \forall u \in \mathcal{I} \quad (2.15c)$$

$$0 \leq \phi_a \leq \bar{\kappa}_a, \quad \forall a \in \bar{\mathcal{A}} \quad (2.15d)$$

$$0 \leq \hat{\phi}_u \leq \nu_u, \quad \forall u \in \mathcal{I} \quad (2.15e)$$

The symbols have their usual meaning as defined previously.

In case of dynamic network, the transit times on the anti-parallel arcs plays important role on reversing the direction of arcs. The reversal of arc is made to increase the flow within given time horizon or to reduce the overall makespan for the given amount of flow. At the time of evacuation, our assumption is that lanes towards the danger zones (sources) are almost empty and the reverse of lanes towards the safe zones (sinks) plays an effective role in evacuation process.



(i) Two-way symmetric network

(ii) Two-way asymmetric network

Figure 2.13: Two-way networks with (i) symmetric and (ii) asymmetric transit times.

Transit times on anti-parallel arcs may be same or different, according to which the network can be classified as the network with symmetric or asymmetric transit times (see Figure 2.13).

Hereafter, we discuss and analyze the contraflow configuration with these two different aspects of transit times between the pair of oppositely directed arcs. For asymmetric network, we focus on the contraflow with orientation-dependent transit times and contraflow with anti-parallel path decomposition.

2.4.1 Contraflow with Symmetric Transit Times

Consider a two-way dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$ in which transit times in anti-parallel arcs $a = (u, v)$ and $\overleftarrow{a} = (v, u)$ are same, i.e., $\tau_a = \tau_{\overleftarrow{a}}$ for all $a, \overleftarrow{a} \in \mathcal{A}$. Let κ_a and $\kappa_{\overleftarrow{a}}$ be the capacity on a and \overleftarrow{a} , respectively. To solve the network flow problem with symmetric transit times, an auxiliary network is constructed as follows.

Auxiliary Network. The auxiliary network of given two-way network Π is denoted by $\overline{\Pi} = (\mathcal{N}, \overline{\mathcal{A}}, \overline{\kappa}, \overline{\tau}, s, t, T)$, where $\overline{\mathcal{A}}$ contains the undirected (or bi-directed) edges $\overline{\mathcal{A}} = \{(u, v) : (u, v) \text{ or } (v, u) \in \mathcal{A}\}$. The capacity $\overline{\kappa}$ of an arc in the auxiliary network is the sum of capacities of anti-parallel arcs a and \overleftarrow{a} , i.e., $\overline{\kappa}_a = \kappa_a + \kappa_{\overleftarrow{a}}$, where $\kappa_a = 0$ if $a \notin \mathcal{A}$. Similarly, the transit time of arc in an auxiliary network $\overline{\tau}$ is obtained by

$$\overline{\tau}_a = \begin{cases} \tau_a & \text{if } a \in \mathcal{A} \\ \tau_{\overleftarrow{a}} & \text{otherwise.} \end{cases}$$

All other parameters of $\overline{\Pi}$ are same as in Π . While solving the network flow problems using some programming languages like python, input of the arcs must be in some direction. So, instead of undirected arcs, bi-directional input is used for the auxiliary network. We must be careful that, if the problem with intermediate storage is to be considered, then one more parameter of storage capacity ν on the nodes is included.

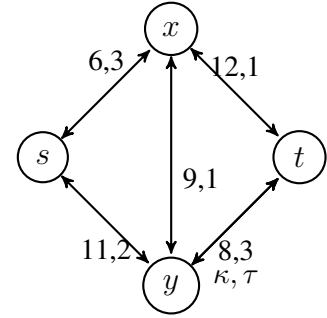


Figure 2.14: Auxiliary network of Figure 2.13(i).

Rebennack et al. (2010) used the TRF on the set of paths obtained from the chain decomposition of the optimal minimum cost flow of auxiliary network to find the solution of maximum dynamic contraflow (MDCF) problem in polynomial time. The linear programming dynamic contraflow model defined on auxiliary network is,

$$\max |\Phi| \tag{2.16a}$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \sum_{\theta=0}^T \Phi_a(\theta) = |\Phi| = \sum_{a \in \Gamma_t^{in}} \sum_{\theta=\overline{\tau}_a}^T \Phi_a(\theta - \overline{\tau}_a) \tag{2.16b}$$

$$\sum_{a \in \Gamma^{in}(u)} \sum_{\beta = \bar{\tau}_a}^{\theta} \Phi_a(\beta - \bar{\tau}_a) - \sum_{a \in \Gamma^{out}(u)} \sum_{\beta = 0}^{\theta} \Phi_a(\beta) \geq 0, \quad u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.16c)$$

$$0 \leq \Phi_a(\theta) \leq \bar{\kappa}_a, \quad \forall a \in \bar{\mathcal{A}}, \theta \in \mathcal{T} \quad (2.16d)$$

Symmetric MDCF with Intermediate Storage Using TRF

The solution strategy of MDCF problem with intermediate storage in symmetric network can be found in Pyakurel & Dempe (2020). Their solution strategy depends on the lexicographic maximum dynamic flow of Hoppe & Tardos (2000). Our concern here is to use TRF to solve MDCF problem with intermediate storage, where storage capacity of intermediate nodes must be $\nu_u \geq T \sum_{a \in \Gamma_u^{in}} \kappa_a$, $\forall u \in \mathcal{I}$.

Flow Model. The linear programming contraflow model with intermediate storage defined on auxiliary network can be presented as

$$\max |\Phi| \quad (2.17a)$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \sum_{\theta = 0}^T \Phi_a(\theta) = |\Phi| = \sum_{a \in \Gamma_t^{in}} \sum_{\theta = \bar{\tau}_a}^T \Phi_a(\theta - \bar{\tau}_a) + \sum_{u \in \mathcal{I}} \hat{\Phi}_u(T) \quad (2.17b)$$

$$\sum_{a \in \Gamma^{in}(u)} \sum_{\beta = \bar{\tau}_a}^{\theta} \Phi_a(\beta - \bar{\tau}_a) - \sum_{a \in \Gamma^{out}(u)} \sum_{\beta = 0}^{\theta} \Phi_a(\beta) = \hat{\Phi}_u(\theta), \quad u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.17c)$$

$$0 \leq \Phi_a(\theta) \leq \bar{\kappa}_a, \quad \forall a \in \bar{\mathcal{A}}, \theta \in \mathcal{T} \quad (2.17d)$$

$$0 \leq \hat{\Phi}_u(\theta) \leq \nu_u, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (2.17e)$$

Solution Procedure. For the solution of MDCF problem with intermediate storage, we first fix the priority order of nodes as defined in Subsection 2.3.3. We then construct an auxiliary network of the given network and apply Algorithm 2 on it to obtain the directed path with excess flow. As the network has symmetric transit times on anti-parallel arcs, the priority order of nodes in the given network remains same as the priority order of nodes in its auxiliary network. The maximum dynamic flow from s to t and then at each intermediate nodes with their respective priority order is obtained by using equations (2.12a) and (2.12b). Here, we present Algorithm 4 to solve the MDCF problem with intermediate storage by using TRF.

Theorem 2.5. *Algorithm 4 solves the MDCF problem with intermediate storage optimally by using temporal repetition of flow in polynomial time.*

Proof. Fixing the priority of nodes and constructing auxiliary network in Steps 1 and 2 are

Algorithm 4: MDCF algorithm with intermediate storage using TRF

Input : Given a two-way dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \nu, \tau, s, t, T)$.

Output: MDCF with intermediate storage.

1. Fix the first priority to the sink t and the priority order of intermediate nodes $u \in \mathcal{I}$ with farther in distance from the source higher in priority.
 2. Construct an auxiliary network $\bar{\Pi}$ of Π .
 3. Use Algorithm 2 in $\bar{\Pi}$ to find directed flow balancing paths with excess flow at the nodes.
 4. Obtain the MDF using TRF presented in equations (2.12a) and (2.12b).
-

both feasible because no flow conservation and capacity constraints are violated. By using Algorithm 2 in auxiliary network, the optimal static flow with intermediate storage and its path decomposition is obtained in Step 3 and in Step 4, the temporal repetition of flow provides the optimal solution to the MDF problem with intermediate storage. Thus, Algorithm 4 provides the optimal solution to MDCF problem with intermediate storage. The time complexity of Algorithm 4 depends on the complexity of Step 3, which is polynomial of $O(m)$ and Step 4, which is polynomial. So, Algorithm 4 solves the MDCF problem with intermediate storage optimally by using temporal repetition of flow in polynomial time. \square

If the contraflow network is series-parallel, then Algorithm 4 provides the earliest arrival contraflow solution by using TRF. The idea behind is from Ruzika et al. (2011) by incorporating the storage of flow at intermediate nodes as presented in Subsection 2.3.4.

2.4.2 Asymmetric Contraflow with Orientation-dependent Transit Times

In a two-way network Π , the traversal time in anti-parallel arcs may not always be same. Network with different transit times in anti-parallel arcs (i.e., $\tau_a \neq \tau_{\bar{a}}$) is known as the network of asymmetric transit times. We can realize the asymmetric transit times on the one-way roads between two places or the curved and inclined/declined roads in mountain regions.

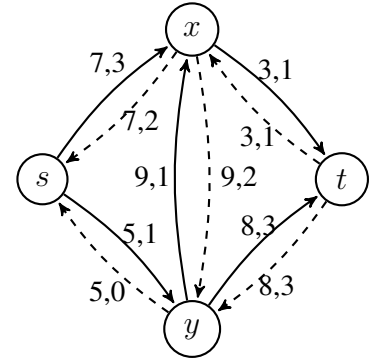


Figure 2.15: Auxiliary network of Figure 2.13(ii).

For the contraflow configuration of such network, if the transit time of reversed arc is taken as the orientation of the arc, then it is called the contraflow with orientation dependent transit times (Nath et al. (2021)). While constructing the auxiliary network $\bar{\Pi} = (\mathcal{N}, \bar{\mathcal{A}}, \bar{\kappa}, \bar{\tau}, s, t, T)$ of Π , transit time $\bar{\tau}$ is obtained as follows.

$$\bar{\tau}_a = \begin{cases} \tau_a & \text{if the orientation is along the arc } a \\ \tau_{\bar{a}} & \text{if the orientation is along the arc } \bar{a} \text{ or } a \notin \mathcal{A}. \end{cases}$$

The other components of $\bar{\Pi}$ are as similar to the contraflow with symmetric transit times. Figure 2.15 is an auxiliary network of Figure 2.13(ii) where capacities in anti-parallel arcs are added but transit times are as per the directions. On solving network flow problems, either dashed or dotted arcs are used for the flow transmission but not both.

Orientation-dependent MDCF with Intermediate Storage Using TRF

The algorithmic framework of MDCF problem with intermediate storage is same in symmetric and orientation-dependent contraflow except on creating auxiliary network and fixing the priority order of nodes. In Step 2 of Algorithm 4, we must use orientation-dependent contraflow. Similarly, for the uniqueness of the solution, priority order of nodes in Step 1 of Algorithm 4 is fixed from the given network topology before the contraflow configuration because of two reasons, no excess flow computation is possible without deciding the priority and the priority of nodes (i.e., distance from source) may vary in the auxiliary network due to orientation dependent transit times (Khanal et al. (2021a)). After construction of auxiliary network in Step 2 with orientation dependent transit times, Algorithm 4 provides solution of the maximum dynamic contraflow problem with intermediate storage using TRF optimally in polynomial time. The solution of EACF problem in series-parallel network is also similar as discussed above.

2.4.3 Asymmetric Contraflow with Anti-parallel Path Decomposition

In this subsection, we present a novel technique of contraflow configuration with asymmetric transit times, termed as contraflow with anti-parallel path decomposition. For each given arc, we create an anti-parallel path of two arcs. If $a = (u, v)$ be an arc of given network with capacity κ_a and transit time τ_a , then an oppositely directed additional path $v-vu-u$ of two arcs $a_1 = (v, vu)$ and $a_2 = (vu, u)$ is created with $\kappa_{a_1} = \kappa_{a_2} = \kappa_a$ and $\tau_{a_1} = r\tau$, $\tau_{a_2} = (1-r)\tau$, $0 \leq r \leq 1$, where vu is an artificial node along the path from v to u (see Figure 2.16).



Figure 2.16: Anti-parallel path decomposition (b) of given arc (a) with capacity and transit time.

An extended network of a given network with n nodes and m arcs obtained in this process contains $m+n$ nodes and $3m$ arcs. This contraflow technique depends on the extended network instead of auxiliary network which helps to solve the contraflow problem with unequal transit times on anti-parallel arcs. In the auxiliary network with orientation dependent transit times, properties of the arcs are modified as per the change in directions. Sometimes, this may not be possible because the transit time for a road remains the same whatever be the direction of

vehicles, e.g. the one-way roads connecting two cities with parallel paths of different length. We form an extended network Π' of given network Π in such a way that for each given arc $a \in \mathcal{A}$, we create an anti-parallel path without removing the original arc as in Figure 2.16. The graphical structure of an extended network is of the form $\Pi' = (\mathcal{N}', \mathcal{A}', \kappa, \tau', s, t, T)$ where, $\mathcal{N}' = \mathcal{N} \cup \{u' : u' = uv \text{ or } vu\}$, $\mathcal{A}' = \mathcal{A} \cup \{a_1\} \cup \{a_2\}$ and $\tau' = \{\tau\} \cup \{\tau_{a_1}\} \cup \{\tau_{a_2}\}$. As the flow is transshipped in one of the direction, flow in the extended network provides the optimal solution to the original network with contraflow configuration. Using this contraflow technique, we solve the maximum dynamic contraflow (MDCF), quickest contraflow (QCF) and the route based evacuation problems hereafter.

(i) Maximum Dynamic Contraflow (MDCF) with Anti-parallel Path Decomposition

The solution strategy of MDCF problem using anti-parallel path decomposition seeks to obtain the maximum flow from the source to the sink within given time horizon T by creating anti-parallel path for each given arc at time zero. The flow model is as similar to maximum dynamic flow defined in Subsection 2.2.2, where the only difference is that it must be defined on $\Pi' = (\mathcal{N}', \mathcal{A}', \kappa, \tau', s, t, T)$ instead of Π .

Solution Procedure. The solution procedure for MDCF problem starts with the construction of an extended network Π' by creating anti-parallel path for each arc. The static flow from the source is transmitted to the sink in the extended network Π' so that flow takes one of the directions and the removal of unused anti-parallel arc/path declares the flow transmission is without any circulation. Finally, network is transformed to the original network with appropriate arc reversals. Now, we present an algorithm to solve the MDCF problem with anti-parallel path decomposition as follows.

Algorithm 5: MDCF algorithm with anti-parallel path decomposition

Input : Given a dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$.

Output: Maximum dynamic contraflow with anti-parallel path decomposition in Π .

1. Construct an extended network Π' with anti-parallel path decomposition.
 2. Obtain the static $s - t$ flow on Π' and remove unused arcs and nodes.
 3. Decompose the static flow on paths and cycles and remove the cycles.
 4. Calculate the dynamic flow with temporally repeated solution.
 5. Remove the artificial nodes of reversed anti-parallel paths to transform the solution in original network.
-

Theorem 2.6. *Algorithm 5 solves the MDCF problem with asymmetric transit times by using anti-parallel path decomposition optimally.*

Proof. In Step 1, the construction of an extended network Π' with $m + n$ nodes and $3m$ arcs from the given network can be done in linear time complexity of $O(m)$. Also, while calculating static $s - t$ flow in Step 1, it takes the arcs in one of the direction but not in both and in Step 3,

flow is decomposed on the paths. These two steps are equivalent to finding minimum cost maximum static flow by considering transit time as cost. Similarly, Step 4 can be obtained by using TRF on the decomposed paths. Thus the solution obtained from Algorithm 5 is feasible. Again, static flow in Step 1 is an optimal solution and its temporal repetition over time is the optimal dynamic flow, so Algorithm 5 provides the optimal solution of MDCF problem with asymmetric transit times by using anti-parallel path decomposition. \square

Since the time complexity of construction of extended network Π' does not effect the complexity of the maximum dynamic flow, the time complexity of maximum static flow is $O(mn \log n)$. The temporally repeated flow executes the dynamic flow in constant time. So, MDCF problem with asymmetric transit times by using anti-parallel path decomposition can be solved in polynomial time.

Lemma 2.7. *Algorithm 5 runs in polynomial time complexity.*

Example 2.5. Consider a two-way network with asymmetric transit times on the arcs as shown in Figure 2.17(a) whose expanded network with anti-parallel path decomposition is presented in Figure 2.17(b) with $r = \frac{1}{2}$. Numbers on each arc represent the capacity and transit time. The static flow is sent in expanded network from the source s to the destination t and flow is decomposed to the paths in which it takes one of the direction. According to Step 1 and Step 3 of Algorithm 5, static flow on expanded network is sent from source s to the sink t using five paths (minimum cost maximum flow paths) as given in Table 2.1.

Table 2.1: Maximum static path flow with anti-parallel path decomposition.

Path	Transit time of path	Static flow (ϕ_P)
$P_1 : s - sx - x - xt - t$	$\tau'_{P_1} = 4$	3
$P_2 : s - sx - x - y - yt - t$	$\tau'_{P_2} = 4$	1
$P_3 : s - sx - x - y - t$	$\tau'_{P_3} = 5$	2
$P_4 : s - y - t$	$\tau'_{P_4} = 6$	2
$P_5 : s - x - xy - y - t$	$\tau'_{P_5} = 7$	1

Let $T = 10$ be given time horizon. To obtain the dynamic flow with anti-parallel path decomposition, we use TRF on each path obtained by static flow decomposition (see in Figure 2.17(c)) so that total maximum dynamic contraflow within time horizon $T = 10$ is $|\Phi| = \sum(T + 1 - \tau'_P)\phi_P = 7 * 3 + 7 * 1 + 6 * 2 + 5 * 2 + 4 * 1 = 54$ units. The transformed network after contraflow configuration is presented in Figure 2.17(d). \square

(ii) Quickest Contraflow with Anti-parallel Path Decomposition

Here, we solve a quickest flow problem in which flow value $|\Phi|$ is given and our concern is to find a minimum possible time that is necessary to transship the given amount of flow from the

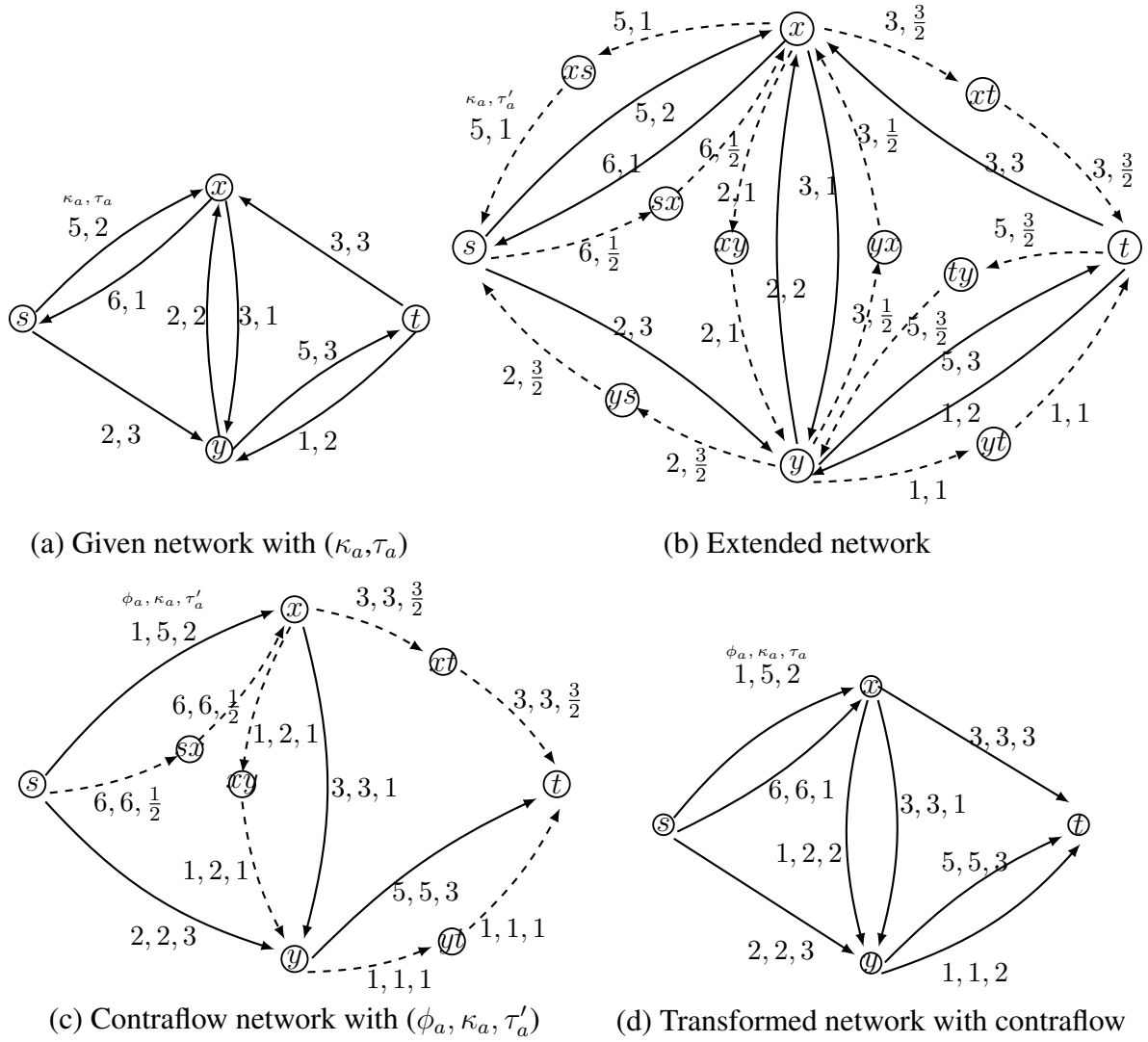


Figure 2.17: Contraflow with anti-parallel path decomposition.

source to the sink by using anti-parallel path decomposition. We apply binary search method to solve the problem in polynomial time.

Flow Model. For the flow over time function Φ with flow value $|\Phi|$ defined on an expanded network Π' , where $\Phi_a(\theta)$ represents the dynamic flow on arc a that moves from tail node of $a \in \mathcal{A}'$ at time θ with arc traversal time τ'_a , the mathematical model for the quickest flow problem is a linear network flow model which is presented as follows.

$$\min T \tag{2.18a}$$

such that,

$$\sum_{a \in \Gamma_s^{out}} \sum_{\theta=0}^T \Phi_a(\theta) = \sum_{a \in \Gamma_t^{in}} \sum_{\theta=\tau'_a}^T \Phi_a(\theta - \tau'_a) \geq |\Phi| \tag{2.18b}$$

satisfying the weak flow conservation constraint in (2.6a) and capacity constraint in (2.6c).

Solution Procedure. The solution procedure starts with the construction of an extended network Π' by creating anti-parallel path for each arc and the static $s - t$ flow is obtained on it. Let \mathcal{P} be set of all paths used to transship the static $s - t$ flow in Π' and P^* be the shortest path among all flow carrying paths $P \in \mathcal{P}$ with time τ_{P^*} and static flow ϕ_{P^*} . If there are more than one shortest paths then choose the path with minimum flow value. For the polynomial time solution, we use binary search on $[T_{min}, T_{max}]$ such that $\Phi(T_{min}) \leq |\Phi| \leq \Phi(T_{max})$. Initially, we set $T_{min} = \tau_{P^*}$ and $T_{max} = \tau_{P^*} + \lceil \frac{|\Phi|}{\phi_{P^*}} \rceil$, $\phi_{P^*} > 0$. In each iteration, the searched interval is halved until there exist T with $\Phi(T)$ converging to $|\Phi|$ and the flow values at extreme points of interval are obtained by using Algorithm 5. The algorithmic framework to solve QCF problem by using anti-parallel path decomposition is present in Algorithm 6.

Algorithm 6: QCF algorithm with anti-parallel path decomposition

Input : Given a dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$.

Output: Quickest contraflow with anti-parallel path decomposition in Π .

1. Construct an extended network Π' with anti-parallel path decomposition.
 2. Obtain the static $s - t$ flow on Π' and remove unused arcs and nodes.
 3. Decompose the static flow on paths and cycles and remove the cycles.
 4. Use binary search on $[T_{min}, T_{max}]$ to find T so that $\Phi(T)$ converges to $|\Phi|$.
 5. $T =$ Quickest time to satisfy given flow value $|\Phi|$.
-

Theorem 2.8. *Algorithm 6 solves the QCF problem in polynomial time using anti-parallel path decomposition.*

Proof. The optimality of Steps (1–3) are as in Algorithm 5. As function $\Phi(T)$ is non-decreasing function of T and searching of minimum T satisfying $\Phi(T) \geq |\Phi|$ using binary search can be made optimally in polynomial time, Algorithm 6 solves the QCF problem in polynomial time complexity. As discrete time T is an integer and the length of interval in k^{th} iteration is $\frac{T_{max} - T_{min}}{2^k}$, continuation of bisection goes unless the length of interval is less or equals to 1. Thus, $\frac{T_{max} - T_{min}}{2^k} \leq 1$ implies $k \geq \log(T_{max} - T_{min})$. This shows that the lower bound for searching a minimum number of iteration k is $\log(T_{max} - T_{min})$, which is polynomial. \square

Example 2.6. Consider the network presented in Example 2.5. Suppose that the given flow value $|\Phi|$ at the source is 80 units. As there are two shortest paths with time 4, we choose the path with minimum flow value so that $\tau_{P^*} = 4$ and flow $\phi_{P^*} = 1$. Initially, $[T_{min}, T_{max}] = [4, 80]$. By using Algorithm 6, quickest time to transship 80 units of flow is $T^* = 13$. \square

The contraflow technique with anti-parallel path decomposition is useful for the route-based evacuation planning because it has the possibility of contraflow on directed paths not only towards the sink but also towards the source. For this purpose, the solution strategy of route-based evacuation problem is presented hereafter.

(iii) Route-based Evacuation Planning Using Anti-parallel Path Decomposition

An evacuation scenario in which every evacuee depend on the public vehicles of predefined route is a route-based evacuation. Evacuees are to be collected at pickup location (source node) s nearby the danger zone and evacuated to the sink t (safe zone) by sending vehicles from the safe zone. Our assumption is that the pickup location s is very near but comparatively safer than the danger zone so that the evacuees can move to the pickup location themselves within arbitrary small (zero) transit time. Also, no individuals have their own vehicles so that evacuation scenario is completely route-based. As the vehicles are first moved from the sink to the source using $t - s$ paths and pickup the evacuees from s to bring back at the sink t by using another $s - t$ path, the network topology must have at least two arcs in either of the cut that separates the source and sink so that the circulation of flow becomes smooth.

Here, we solve the route-based maximum dynamic evacuation planning problem with asymmetric times on anti-parallel arcs by using circulation of flow (vehicles) on the extended network which is obtained by anti-parallel path decomposition. In real evacuation scenarios, the first trip of vehicles can originate from anywhere and form the circulation of the flow from the second trip onward, but this complicates the formulation of mathematical models. So, from the beginning of the evacuation process, we assume that all of the vehicles begin to move from the sink and form a circulation of flow at the sink. For this purpose, we solve the problem with two objectives, one is selection of the circuits with possibly shortest route solving subset sum problem and another one is maximizing the number of evacuees on the route.

Due to the construction of anti-parallel arcs in Π' , every $s - t$ path in Π has its $t - s$ path in Π' . Let $\mathcal{P} = \overleftarrow{\mathcal{P}} \cup \overrightarrow{\mathcal{P}}$ be set of all paths with backward ($t - s$) and forward ($s - t$) directions in Π' with $\overleftarrow{P} \in \overleftarrow{\mathcal{P}}$ and $\overrightarrow{P} \in \overrightarrow{\mathcal{P}}$, respectively. The minimum cut capacity of $\overleftarrow{\mathcal{P}}$ and $\overrightarrow{\mathcal{P}}$ are same because each arc has its anti-parallel path. Hereafter, we denote \mathcal{P}^c to the set of all paths P that forms the circulation $t - s - t$ at t and \mathcal{P}_a be the set of paths that passes through arc $a \in \mathcal{A}'$ i.e., $\mathcal{P}_a = \{P \in \mathcal{P}^c : a \in P\}$. The mathematical model defined on the extended network with circulation of paths is as follows.

Flow Model. The dynamic flow function Φ defined on the network Π' with arc traversal time τ' is the collection of non-negative path flow function $\Phi_P : P \times \mathcal{T} \rightarrow \mathbb{R}^+$. The linear programming model for the dynamic contraflow problem with circulation of flow at sink t can be presented as follows.

$$\max |\Phi| \tag{2.19a}$$

such that,

$$\sum_{P \in \mathcal{P}^c} \sum_{\theta=0}^T \Phi_P(\theta) = |\Phi| \tag{2.19b}$$

$$\sum_{P \in \mathcal{P}_a: a \in \Gamma_u^{in}} \Phi_P(\theta) - \sum_{P \in \mathcal{P}_a: a \in \Gamma_u^{out}} \Phi_P(\theta) = 0, \quad \forall u \in \mathcal{N}', \theta \in \mathcal{T} \quad (2.19c)$$

$$\sum_{P \in \mathcal{P}_a} \Phi_P(\theta) \leq \kappa_a, \quad \forall a \in \mathcal{A}', \theta \in \mathcal{T} \quad (2.19d)$$

$$\Phi_P \geq 0, \quad \forall P \in \mathcal{P}^c \quad (2.19e)$$

Here, equation (2.19a) is an objective function which seeks to maximize the total flow and this flow obtained by circulation at sink t along paths $P \in \mathcal{P}^c$ within the time horizon T is represented in equation (2.19b). Flow conservation at each node lying on the path is represented by equation (2.19c). Equations (2.19d) and (2.19e) represent the capacity constraint in each arc and non-negativity of the flow in each circulation path, respectively.

Solution Procedure. To form a set \mathcal{P}^c of the circulation paths P of maximum flow not exceeding the half of minimum cut capacity in Π' , we proceed as follows. As minimum cut capacity of $s - t$ and $t - s$ paths in Π' are same, we construct two lists $F = \{\phi_{\overleftarrow{P}}\}$ and $\Gamma = \{\tau'_{\overleftarrow{P}}\}$ of static flows and their corresponding path transit times of all $t - s$ paths, respectively, where $\tau'_{\overleftarrow{P}} = \sum_{a \in \overleftarrow{P}} \tau'_a$. Our aim is to search a subset F^* of F in such a way that the sum of minimum cut capacity of $t - s$ paths in F^* is nearly half of the total capacity of $t - s$ paths but not exceeding it, i.e., greatest sum such that $\sum F^* \leq \lfloor \frac{\sum F}{2} \rfloor$ and the corresponding subset Γ^* of Γ with minimum sum i.e., $\sum \Gamma^*$ is minimum (possible shortest routes). The notation $\sum F$ represents the sum of elements in set F and have similar meaning for $\sum F^*$ and $\sum \Gamma^*$. Mathematically, it can be modeled as follows.

$$\min \sum \tau'_{\overleftarrow{P}} \quad (2.20a)$$

such that,

$$\overleftarrow{P} \in \overleftarrow{\mathcal{P}} \text{ and } \tau'_{\overleftarrow{P}} \in \Gamma^* \subset \Gamma. \quad (2.20b)$$

To obtain the $t - s$ paths $\overleftarrow{P} \in \overleftarrow{\mathcal{P}}$ in equations (2.20a–2.20b) with flows in $F^* \subset F$, we model the subset sum problem as follows.

$$\max \sum \phi_{\overleftarrow{P}} \cdot \gamma_{\overleftarrow{P}} \quad (2.21a)$$

such that,

$$\sum \phi_{\overleftarrow{P}} \cdot \gamma_{\overleftarrow{P}} \leq \left\lfloor \frac{\sum F}{2} \right\rfloor, \text{ for } F = \{\phi_{\overleftarrow{P}}\} \quad (2.21b)$$

$$\gamma_{\overleftarrow{P}} = \begin{cases} 1 & \text{if } \overleftarrow{P} \text{ is selected for its flow in } F^* \subset F \\ 0 & \text{otherwise.} \end{cases} \quad (2.21c)$$

The paths \overleftarrow{P} with flow values in F^* satisfying subset sum problem (2.21a – 2.21c) which also satisfy the minimum transit time from equation (2.20a) are used to send the flows from t to s . Then after, their corresponding anti-parallel paths are removed. Again, $s - t$ paths \overrightarrow{P} are obtained so that the sum of path flows is close to $\lfloor \frac{\sum F}{2} \rfloor$ and then, their corresponding anti-parallel paths are removed. Now to form the circulation of path flows in \mathcal{P}^c , we obtain the maximum static flow along $t - s$ paths \overleftarrow{P} at time $\theta = 0$ and continue the same flow on $s - t$ paths \overrightarrow{P} . TRF along the circulation paths \mathcal{P}^c is used to obtain the dynamic flow.

We remark that the process of finding $t - s$ paths from equations (2.20a – 2.21c) can also be defined as a multi-objective optimization problem which aims to search the subset of paths of maximum flow with minimum traversal time. From the set of solutions obtained by equations (2.21a – 2.21c), equation (2.20a) selects the possibly best one. The boundedness in equation (2.21b) seems it to be solvable by bounded objective function method.

Example 2.7. Consider an arbitrary network having 7 paths from t to s with positive flow and their corresponding transit times in Π' as follows.

$$\overleftarrow{P}_1 : \tau'_1 = 2, \phi_1 = 5; \quad \overleftarrow{P}_2 : \tau'_2 = 3, \phi_2 = 3; \quad \overleftarrow{P}_3 : \tau'_3 = 4, \phi_3 = 7; \quad \overleftarrow{P}_4 : \tau'_4 = 4, \phi_4 = 4; \quad \overleftarrow{P}_5 : \tau'_5 = 5, \phi_5 = 2; \quad \overleftarrow{P}_6 : \tau'_6 = 6, \phi_6 = 6; \quad \overleftarrow{P}_7 : \tau'_7 = 6, \phi_7 = 2.$$

Thus, we have $F = \{5, 3, 7, 4, 2, 6, 2\}$ with $\sum F = 29$ and $\Gamma = \{2, 3, 4, 4, 5, 6, 6\}$. The possible subsets F^* with $\sum F^* \leq \lfloor \frac{\sum F}{2} \rfloor = 14$ are as follows.

$$F_1^* = \{5, 3, 4, 2\}, \sum \Gamma_1^* = 14; \quad F_2^* = \{5, 7, 2\}, \sum \Gamma_2^* = 11; \quad F_3^* = \{5, 3, 6\}, \sum \Gamma_3^* = 11; \\ F_4^* = \{3, 7, 4\}, \sum \Gamma_4^* = 11; \quad F_5^* = \{3, 7, 2, 2\}, \sum \Gamma_5^* = 18; \quad F_6^* = \{4, 2, 6, 2\}, \sum \Gamma_6^* = 21.$$

So the possible subsets with minimum path transit times are F_2^* , F_3^* , and F_4^* , and one of the set is used for $t - s$ paths. If F_2^* with paths $\{\overleftarrow{P}_1, \overleftarrow{P}_3, \overleftarrow{P}_5\}$ is used for $t - s$ paths, then their anti-parallel paths are removed. From the network so obtained, $s - t$ paths with maximum static flow computation are obtained and their anti-parallel paths are removed. \square

Now we present Algorithm 7 to solve the maximum flow problem with circulation of flow on paths $P \in \mathcal{P}^c$.

Theorem 2.9. *Solution obtained from the route-based maximum flow circulation in Algorithm 7 is approximate efficient over the circulation of paths.*

Proof. The feasibility on constructing the extended network in Step 1 is from Theorem 2.6. The construction of a subset F^* from the list of $t - s$ paths by using subset sum problem is also feasible because no paths violate the flow conservation and capacity constraints. The maximum static flow obtained in $t - s - t$ circulation is feasible because it is obtained from set of feasible paths. So, the solution obtained in Algorithm 7 is feasible. Again, paths obtained from Steps 4 and 5 in the direction of $t - s$ and then $s - t$ are independent. The selection of $t - s$ paths with capacity $\lfloor \frac{\sum F}{2} \rfloor$ is made by polynomial time approximation scheme of Kellerer et al. (2003) and paths with minimum sum in equation (2.20a) is used to rescue the evacuees as quickly as

Algorithm 7: Route-based maximum flow circulation algorithm

Input : Given evacuation network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, \tau, s, t, T)$.

Output: Maximum flow circulation at sink t with path reversals in Π .

1. Construct extended network Π' with anti-parallel path decomposition having two-way path set $\mathcal{P} = \vec{\mathcal{P}} \cup \overleftarrow{\mathcal{P}}$.
 2. Construct a list F of flow values on each $t - s$ path of Π' with positive flow and a set Γ of respective transit times.
 3. Use subset sum problem in equations (2.21a–2.21c) to find a subset $F^* \subset F$ whose sum is a greatest integer less than or equals to $\lfloor \frac{\sum F}{2} \rfloor$ and whose corresponding sum of transit time in $\Gamma^* \subset \Gamma$ is to be minimized by equation (2.20a).
 4. Save the paths having flow values in F^* as $t - s$ paths $\overleftarrow{\mathcal{P}}$ and remove their anti-parallel paths.
 5. Find $s - t$ paths $\vec{\mathcal{P}}$ from the network obtained in Step (4) so that the sum of path capacities equal to the sum of capacities obtained for the flows in Step (4) and remove their anti-parallel paths.
 6. Obtain the maximum static flow on circulation paths $t - s - t$ obtained from Step 4 and Step 5 at time $\theta = 0$.
 7. Compute the MDF circulation at t within time horizon T by using TRF.
-

possible by saving approximately equal capacity in $s - t$ paths. Similarly, selection of $s - t$ paths can be made by using any path decomposition technique of maximum flow algorithm satisfying the maximum static flow arrived at s by using $t - s$ paths. The TRF on the circulation paths is used to find the dynamic flow. So, Algorithm 7 computes an approximate efficient solution in $t - s - t$ circulation. \square

With the notion of multi-objective optimization, an efficient (or Pareto optimal) solution is one that can't be improved in any of the objectives without compromising at least one of the others. To divide the minimum cut capacity of extended network Π' in to approximately two halves for $t - s$ and $s - t$ paths, we use subset sum problem with minimum of total traversal time on $t - s$ paths. The reason behind dividing the path capacities into two halves is for the optimal use of capacities without wasting unnecessarily while forming the circulation of paths and taking minimum total travel time on $t - s$ paths is to response the evacuees as quickly as possible. So, the solution obtained here is an efficient.

Theorem 2.10. *An efficient solution obtained from Algorithm 7 is executed in polynomial time.*

Proof. Construction of extended network in Step 1 of Algorithm 7 depends on the number of arcs of the given network. In Steps 2, all $t - s$ paths with positive flow can be obtained by cost-scaling in polynomial time by considering transit time as cost, Edmonds & Karp (1972) or any flow decomposition algorithm. Step 4 is saving of $t - s$ paths and removal of their anti-parallel paths and in Step 5, we search $s - t$ paths with maximum static flow using any polynomial time maximum flow algorithm. The TRF in Step 7 for maximum dynamic flow takes the constant time. Similarly, the subset sum problem in Step 3 can be solved by using fully polynomial

approximation scheme of Kellerer et al. (2003) in $O(\min\{n \cdot \frac{1}{\epsilon}, n + \frac{1}{\epsilon^2} \log(\frac{1}{\epsilon})\})$ time, where ϵ is the worst-case relative error. So, the route-based dynamic flow circulation problem can be solved efficiently by Algorithm 7 in polynomial time approximation. \square

Remember that, Algorithm 7 solves the route-based evacuation planning problem by using anti-parallel path decomposition not only for the contraflow network having two-way arcs but also for the general network having one-way arcs with at least two minimum-cut-arcs if reversal of the direction of arc is allowed. In one-way general network, the construction of anti-parallel paths for each arc is similar to two-way network. We can apply Algorithm 7 to solve maximum dynamic flow circulation at sink t as in similar manner. Moreover, this approach also applies to the network with symmetric transit times on the arcs.

Theorem 2.11. *For one-way $s - t$ network with at least two minimum-cut-arcs, Algorithm 7 provides the solution of route-based maximum flow problem with flow circulation by reversing necessary arcs from sink to the source at time zero.*

Example 2.8. Consider a network presented in Figure 2.17(a) of Example 2.5. Let $T = 15$ be given time horizon. The construction of extended network Π' is as in Figure 2.17(b). The $t - s$ paths with positive flow in Π' are

$$\begin{aligned} \overleftarrow{P}_1 &= t - x - s, \tau'_1 = 4, \Phi_1 = 3; & \overleftarrow{P}_2 &= t - y - yx - x - s, \tau'_2 = 4, \Phi_2 = 1; & \overleftarrow{P}_3 &= \\ t - -ty - y - yx - x - s, \tau'_3 &= 5, \Phi_3 = 2; & \overleftarrow{P}_4 &= t - ty - y - ys - s, \tau'_4 = 6, \Phi_4 = 2; \\ \overleftarrow{P}_5 &= t - ty - y - x - xs - s, \tau'_5 = 7, \Phi_5 = 1. \end{aligned}$$

As $F = \{3, 1, 2, 2, 1\}$ with $\sum F = 9$, so we search $t - s$ paths with capacity at most 4 units and having minimum total time by using subset sum problem. Step 3 of Algorithm 7 provides \overleftarrow{P}_1 and \overleftarrow{P}_2 as $t - s$ paths. After removing the anti-parallel paths of \overleftarrow{P}_1 and \overleftarrow{P}_2 , we can find two $s - t$ paths in the remaining network, $s - y - t$ and $s - x - xy - y - t$ with flow value 2 each having transit times 6 and 7 units, respectively.

The one-way representation of Figure 2.18(a) is presented in Figure 2.18(b). While sending static flow on $t - s - t$ circulation, we have three paths $P_1^c = t - x - s - y - t$, $P_2^c = t - x - s - x - xy - y - t$ and $P_3^c = t - y - yx - x - s - x - xy - y - t$ with transit times $\tau'_{P_1^c} = 10$, $\tau'_{P_2^c} = 11$ and $\tau'_{P_3^c} = 11$ which carry the flow of 2, 1 and 1 units, respectively. Total amount of flow circulation in $T = 15$ by using TRF is 22 units. That is, 22 units of vehicles are circulated in $T = 15$ to pickup the evacuees. \square

Case Illustration II

Problem Description. In this case illustration, we are considering an artificial incidence of flooding at the residential area of Kathmandu valley situated at the confluence of two major rivers Bagmati and Manohara. Consider an incidence that there is heavy rainfall in Kathmandu valley and its surrounding mountains at a morning. Meteorological forecasting division release

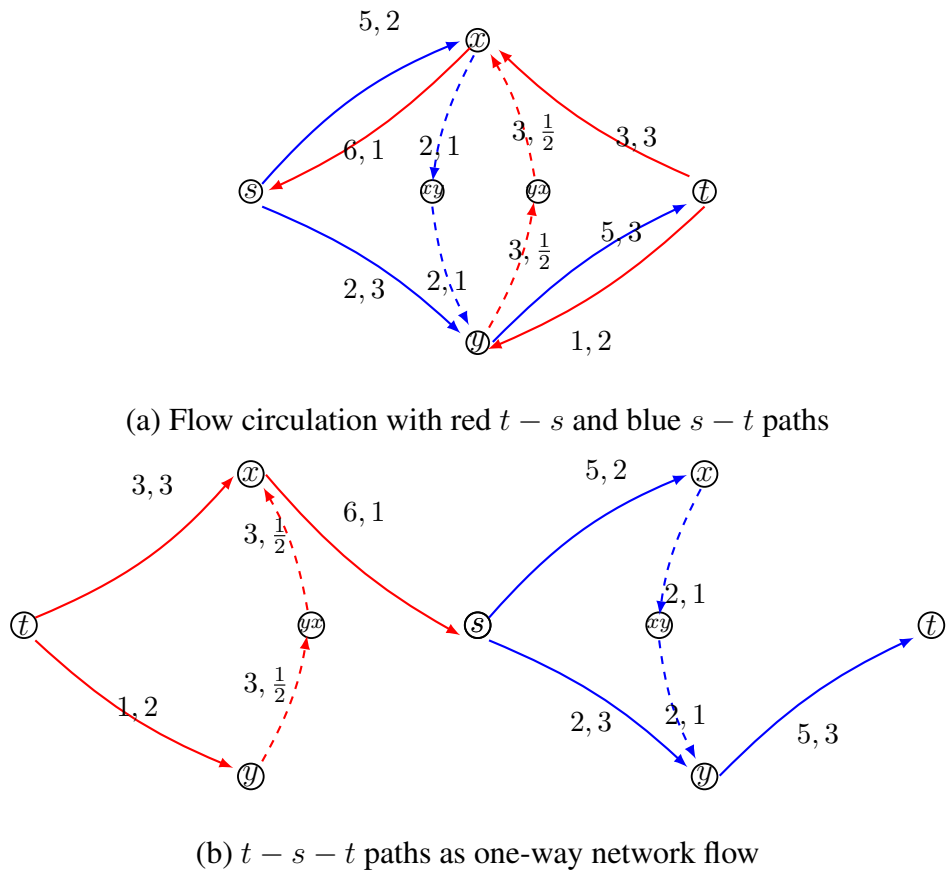


Figure 2.18: (b) represents the $t - s - t$ path of (a) as one-way network flow.

a warn notice that the residential areas of lower part of Buddhanagar situated at the confluence of two rivers Bagmati and Manohara is going to be flooded after 2 hours. Government of Nepal declares the area as an emergency zone and decides to evacuate them at Tudikhel approximately in 1 hour. All the vehicles from Tinkune, Koteshwor, Pulchouk, Kalimati, Jamal and Gausala to this area are diverted to other directions except the vehicles used for the evacuation process. Due to the narrow roads at disaster zone, Traffic Department suggests to use the micro vans which have capacity of 16 passengers at a time. Traffic police has requested micro van service committees for humanitarian support by sending vehicles and managed the routes for these vehicles from Koteshwor, Gwarko and Lagankhel areas to the emergency area. Evacuation zone is presented in the Figure 2.19 below.

To capture the mathematical optimization model of evacuation zone using network optimization, we have created 50 nodes (numbered from 0 to 49) at the crossing points of the roads, where nodes 0 and 49 are the source (emergency area) and sink (safe shelter), respectively. Similarly, paths joining nodes are the road segments, considered as arcs. In Figure 2.19, bold yellow line at the middle and other black dashed lines are paths used for the evacuation process in which some are of one-way. Yellow path from node 2 to 18 is a two-way main road of 8 lanes (4 lanes in each direction) and that from 18 to 47 has 6 lanes (3 lanes in each direction). Similarly, bold dashed road from 21 to 24 is of 6 lanes (3 lanes in each direction) where as from

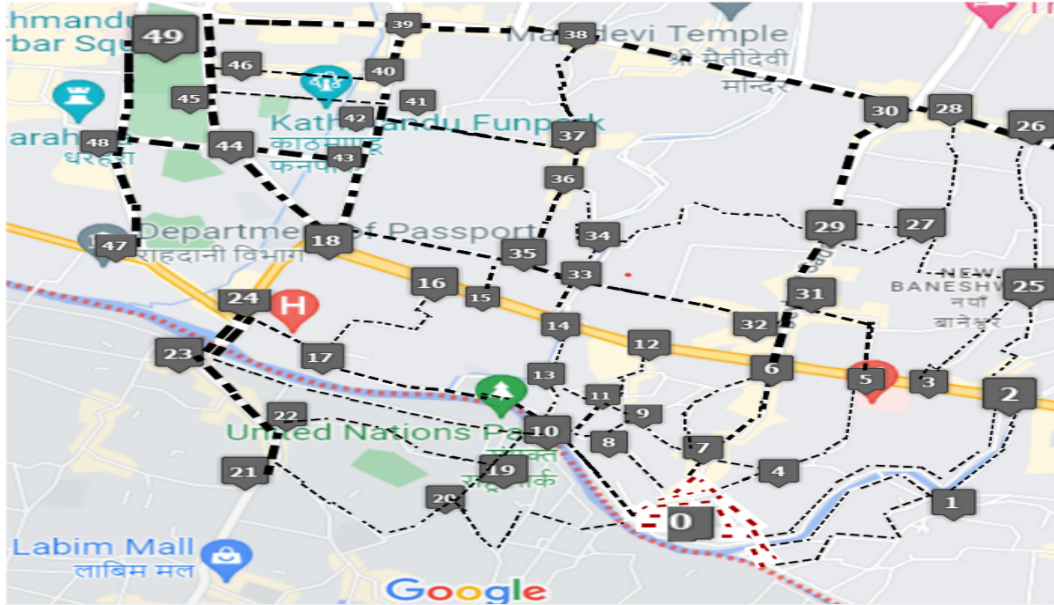


Figure 2.19: Evacuation zone with 50 nodes. Nodes 0 (red dashed lines) and 49 (entry point of green rectangular area) represent the source (emergency area) and sink (safe shelter), respectively.

26 to 49 and 6 to 30 are of 4 lanes (2 lanes in each direction). Thin roads are of 2 lanes. Some roads are one-way, for example, 48 to 49 and 49 to 44 with capacity 6 each. Similarly, some anti-parallel roads are of asymmetric transit times, for example, road from 1 to 0 along corridor of Bagmati river has transit time 1 minute whereas the road next to it from 0 to 1 has transit time 1.5 minutes. The detailed list of arc, capacity and transit time are presented in Appendix, where capacities are considered as per the lanes of road segments and transit times of vehicles (in minutes) are taken with respect to the length and width of the road segments for which average speed of each vehicle is considered as 40 km/hr.

Output: MDCF. To implement MDCF Algorithm 5 to solve the problem, we used programming language of Python 3.7 version on Dell computer with 64-bit operating system, having 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz & 2.42 GHz dual processor and 8 GB RAM. For the given network with 50 nodes (numbered from 0 to 49) and 131 arcs, maximum flow of vehicles sent to the sink in one hour are 377 so that 6032 peoples can be shifted within the time (without contraflow configuration). For anti-parallel path decomposition, we developed and used the codes in python programming language and it executes a extended network with 181 nodes and 393 arcs, where new nodes are numbered by system code from 50 to 180. The maximum flow of vehicles in this network is 756 so that 12096 evacuees can be shifted to the safe shelter. Thus due to asymmetric contraflow with anti-parallel path decomposition, flow of evacuees is increased by 100.53%. The running time of program is 0.12 seconds. While flow transmission with asymmetric contraflow configuration, 23 arcs (1,0), (7,0), (8,0), (10,0), (2,1), (6,7), (7,4), (12,6), (12,11), (14,12), (15,14), (16,15), (18,16), (17,10), (24,17), (25,2), (43,18), (44,43), (45,44), (46,45), (47,24), (48,47) and (49,46) are used in the reverse direction

with anti-parallel path decomposition, which is approximately 18% of the total arcs in the given network. For the comparative study of flow values obtained with and without contraflow in each time interval of 10 minutes, we have illustrated it in Table 2.2 and Figure 2.20.

Table 2.2: Comparison of no. of evacuees reaching to the destination with and without contraflow configuration.

Time (in minutes)	0	10	20	30	40	50	60
Evacuees without contraflow	0	416	1152	2672	3792	4912	6032
Evacuees with contraflow	0	896	3136	5376	7616	9856	12096

Output: QCF. If the population in disaster zone is considered to be 15,000, total number of vehicle trips need to evacuate is $FV = 938$. To solve QCF problem by using Algorithm 6, we have $T_{min} = 5.2$, $T_{max} = 943.2$, because the shortest path have capacity 1 and traversal time 5.2. For the convergence of the flow value, we set tolerance value (error of convergence in $|T_{min} - T_{max}|$ and respective dynamic flow) as $tol = 0.1$. The quickest time of evacuation for 15,000 evacuees executed by python program is 74 minutes (1 hour and 14 minutes) whereas without contraflow is 141 minutes (2 hours and 21 minutes). Due to contraflow configuration with anti-parallel path decomposition, quickest time of evacuation is reduced by 47.5%. The running time of program with $tol = 0.1$ is 3.6 seconds. The comparison of quickest time with and without contraflow using anti-parallel path decomposition is presented in Table 2.3 and Figure 2.21.

Table 2.3: Comparison of quickest time (in minutes) with and without contraflow configuration.

No. of Evacuees	0	2,500	5,000	7,500	10,000	12,500	15,000
Time (min.) without contraflow	0	29	51	74	96	119	141
Time (min.) with contraflow	0	18	29	40	51	62	74

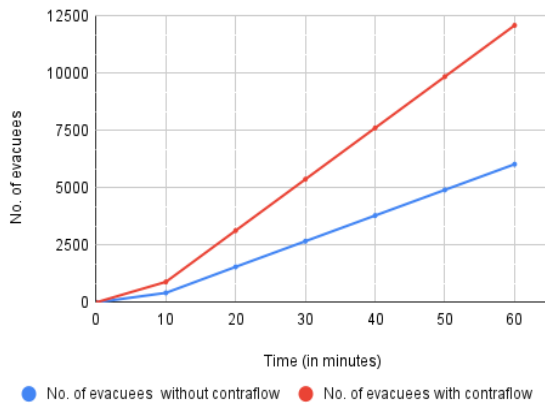


Figure 2.20: Comparison of no. of evacuees with and without contraflow.

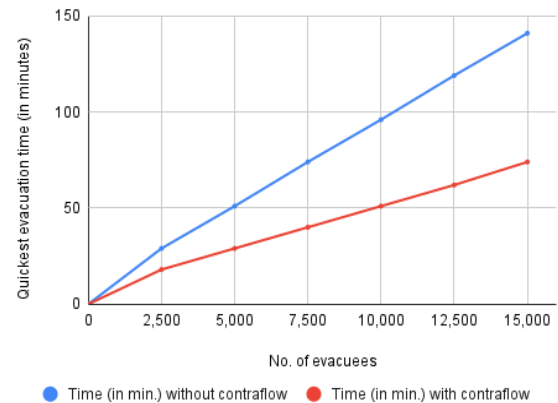


Figure 2.21: Comparison of quickest time with and without contraflow.

Chapter 3

Abstract Network Flow

In this chapter, we describe the flow problems in an abstract network associated with the set of paths and elements, where elements are capacitated and paths are linearly ordered subset of elements. Every path in the abstract network must satisfy the switching property: when two paths cross at an element then there must be a path that is a subset of the first path up to the crossing element and a subset of the second path after the crossing element. Traffic management offices use different tools of traffic diversion like physical presence of traffic police, use of traffic signals, traffic lights, diversions, barricades, etc. to divert the flows on crossing paths to non-crossing sides. In this chapter, we incorporate the concept of intermediate storage in abstract network which is mainly considered in our paper Pyakurel et al. (2022). We also discuss on an abstract network flow with partial switching of the paths from Khanal et al. (2022).

3.1 Abstract Network Flow with Intermediate Storage

Our main aim in this section is to introduce the concept of intermediate storage on an abstract network topology, where nodes are taken as elements. As the solution strategy, we fix the first priority to the sink and the priority of intermediate elements are set according to the maximum of shortest distance from the source. Flows are stored according to their priority order by using lexicographic maximum flow technique. Similarly, we present TRF to obtain the maximum dynamic abstract flow with intermediate storage if the storage capacity of each intermediate element is sufficient (i.e., at least T times the sum of capacities of incoming paths from its left elements).

With flow conservation at each intermediate elements, Hoffman (1974) generalized the max-flow-min-cut theorem of Ford & Fulkerson (1962) for abstract network flow. Due to bottleneck flow on each path, shipment of the excess flow from the source element greater than the minimum cut capacity is not considered. Here, we adopt the intermediate storage of excess flow introduced by Pyakurel & Dempe (2020) for general network in the abstract network.

3.1.1 Notations

Consider an abstract network topology $\Pi = (\mathcal{E}, \mathcal{P})$ with finite set of elements \mathcal{E} and the collection of paths

$$\mathcal{P} = \{P \subseteq \mathcal{E} : P \text{ has a linear order } <_P \text{ of elements in } P\} \subseteq 2^{\mathcal{E}}.$$

Here, the notation \mathcal{P} represents the set of all paths of two type: the source-sink ($s - t$) paths P and the intermediate paths $P_{[s \rightarrow u]}$ from $s \in \mathcal{E}$ to $u \in \mathcal{E} \setminus \{s, t\}$. Except for sink t , each element $u \in \mathcal{E}$ has the non-negative integral movement capacity $\kappa_u : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{Z}^+$ which is used to send the flow from element u to its adjacent elements. Similarly, each element $u \in \mathcal{E}$ has the storage capacity $\nu_u : \mathcal{E} \rightarrow \mathbb{Z}^+$ which is used to hold the flow at u . We denote the order of elements in the path $P \in \mathcal{P}$ by $<_P$ so that $u <_P v$ represents that u is the left of v on P . Similarly, if u is right of v on path P then $u >_P v$ is used. Element $u \in P$ is said to be the leftmost or first (rightmost or last) element of P if there does not exist v in P such that $v <_P u$ ($v >_P u$). Source node s in $s - t$ path P is the leftmost element whereas the sink t is the rightmost. The set of intermediate elements is denoted by $\mathcal{I} = \mathcal{E} \setminus \{s, t\}$.

To have a network $\Pi = (\mathcal{E}, \mathcal{P})$ an abstract network, it must satisfy the switching property: $\forall P, Q \in \mathcal{P}$ and an intermediate element $u \in P \cap Q$, $\exists R \in \mathcal{P}$ such that $R \subseteq P \times_u Q$, where

$$P \times_u Q = \{v \in P : s \leq_P v \leq_P u\} \cup \{v \in Q : u \leq_Q v \leq_Q t\}.$$

In a similar manner, the definition for switched path $R \subseteq Q \times_u P$ can be obtained. For simplicity, we use the notations

$$P_{[s \rightarrow u]} = \{v \in P : s \leq_P v \leq_P u\} \quad \text{and} \quad P_{[u \rightarrow t]} = \{v \in P : u \leq_P v \leq_P t\}$$

to represent the elements on path P from source s up to u and that begin from u up to sink t , respectively. Similarly, the elements on path P that are left of u and right of u can be written as

$$P_{[s \rightarrow u)} = \{v \in P : s \leq_P v <_P u\}, \quad \text{and} \quad P_{(u \rightarrow t]} = \{v \in P : u <_P v \leq_P t\},$$

respectively. If P and Q are two paths both containing u_1 and u_2 , then it is possible to have $u_1 <_P u_2$ but $u_1 >_Q u_2$.

Our assumption is that the terminal elements source and sink have sufficiently large storage capacity, i.e., $\nu_s = \nu_t \leq \infty$ whereas the intermediate elements have finite storage capacity such that $\nu_u \geq \sum_{P \in \mathcal{P}: v <_P u} \kappa_v$ for all $u \in \mathcal{I}$. The source and intermediate elements have finite movement capacities (i.e., $\kappa_u < \infty$, $\forall u \in \mathcal{E} \setminus \{t\}$) and that of the sink is zero (i.e., $\kappa_t = 0$). If the incoming movement capacity of an intermediate element $u \in \mathcal{I}$ is more than the outgoing movement capacity, then the excess flow is used to store at u . Furthermore, the incoming and

outgoing movement capacities of source and sink elements are zero, respectively, except for the contraflow network.

3.1.2 Abstract Maximum Static Flow with Intermediate storage

For the given abstract network $\Pi = (\mathcal{E}, \mathcal{P})$, the abstract maximum static flow problem with intermediate storage is to obtain the maximum flow leaving the source element which is to be shifted to the sink element via $s - t$ paths $P \in \mathcal{P}$ by allowing the maximum storage of excess flow at intermediate elements u via intermediate paths $P_{[s \rightarrow u]} \forall u \in \mathcal{I}$ with storage capacity

$$\nu_u \geq \sum_{P \in \mathcal{P}: v <_P u} \kappa_v.$$

Flow Model. Let $\phi^P : P \rightarrow \mathbb{R}^+$ be the path flow on $s - t$ path P of network $\Pi = (\mathcal{E}, \mathcal{P})$. We can induce a path-flow ϕ^P through the elements lying on it by $\phi_u^P = \sum_{P \in \mathcal{P}: u \in P} \phi^P$. A path flow ϕ^P is feasible if and only if $\phi_u^P \leq \kappa_u$ and $\phi^P \geq 0$ for all $u \in \mathcal{E}$. We say that an element u is saturated with respect to ϕ if $\phi_u^P = \kappa_u$. Denote $\phi_u^{P,out} = \sum_{P \in \Gamma_u^{out}} \phi^P$ and $\phi_u^{P,in} = \sum_{P \in \Gamma_u^{in}} \phi^P$ as the total outflow from u and the total inflow into u , respectively, where Γ_u^{out} and Γ_u^{in} represent the set of outgoing paths from u and incoming paths into u . Let $c_u : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{Z}^+$ be the per unit cost of flow transmission from u to its right element so that $c_P = \sum_{u \in P} c_u$.

Consider the excess flow function $\hat{\phi}_u : \mathcal{I} \rightarrow \mathbb{R}^+$ which stores the flow at element $u \in \mathcal{I}$ and is obtained by the difference of inflow and outflow. The linear programming model of abstract static network flow with intermediate storage can be presented as follows.

$$\max \sum_{P \in \mathcal{P}} \phi^P + \sum_{u \in \mathcal{I}} \hat{\phi}_u \quad (3.1a)$$

such that,

$$\sum_{P \in \mathcal{P}: u \in P} \phi^P \leq \kappa_u, \quad \forall u \in \mathcal{E} \quad (3.1b)$$

$$\phi_u^{P,in} - \phi_u^{P,out} = \hat{\phi}_u, \quad \forall u \in \mathcal{I} \quad (3.1c)$$

$$0 \leq \hat{\phi}_u \leq \nu_u, \quad \forall u \in \mathcal{I} \quad (3.1d)$$

$$\phi^P \geq 0, \quad \forall P \in \mathcal{P} \quad (3.1e)$$

Here, the objective function in equation (3.1a) is to maximize the total flow reaching at sink t as well as the excess flow at intermediate elements. The boundedness of the path flow by the movement capacity of each element is represented in equation (3.1b) and the excess flow is presented in equation (3.1c). The left inequality of equation (3.1d) represents the weak flow conservation whereas the boundedness of the excess flow by the storage capacity is represented in its right inequality. Similarly, equation (3.1e) represents the non-negativity of the flow on

each path. For the feasible solution, the lower bound of storage capacity is considered as $\nu_u \geq \sum_{P \in \mathcal{P}: v < P u} \kappa_v, \forall u \in \mathcal{I}$.

Solution Procedure. As in Section 2.3, the first priority is given to the sink by considering as most appropriate shelter and transship as much flow as possible. To store the excess flow at intermediate elements, the priority ordering is set as farther in distance higher in priority, i.e., $\forall u, v \in \mathcal{I}$ if $d_{P_{[s \rightarrow u]}} > d_{P_{[s \rightarrow v]}}$, then u has higher priority than v and this is denoted by $u \succ v$ where $d_{P_{[s \rightarrow u]}}$ represents the shortest distance (or minimum cost) of element u from s .

For each prioritized element $u \in \mathcal{I}$, we create the dummy element u^* with cost $c_{P_{[u \rightarrow u^*]}} = d_{P_{[u \rightarrow u^*]}} = 0$ and capacities $\kappa_{[u \rightarrow u^*]} = \nu_u = \nu_{u^*}$, where $c_{P_{[u \rightarrow u^*]}}$ and $\kappa_{P_{[u \rightarrow u^*]}}$ are the cost and movement capacity from u to u^* , respectively. The priority order of dummy element u^* is same as of element u and the collection of dummy elements $\{u^*\}$ together with sink t forms a multiple sink D . The modified network $\Pi^* = (\mathcal{E}^*, \mathcal{P}^*)$ is obtained having single source s and multiple sinks $D = \{t\} \cup \{u^*\}$, where $\mathcal{E}^* = \mathcal{E} \cup \{u^*\}$ and $\mathcal{P}^* = \mathcal{P} \cup \{P_{[s \rightarrow u^*]}\}$.

Let u_1, \dots, u_{n-2} be $n - 2$ intermediate elements with priority order $t \succ u_1 \succ u_2 \succ \dots \succ u_{n-2}$ in \mathcal{E} . Then the dummy element u_i^* of each u_i forms the priority order $u_0^* \succ u_1^* \succ \dots \succ u_{n-2}^*$ in D with denotation $t = u_0^*$. A sequence of elements is said to be compatible if the elements respect their ranks. Here, $D = \{u_0^* \succ u_1^* \succ \dots \succ u_{n-2}^*\}$ is the prioritized set of sinks in which more priority is given to the one in left than in right and satisfies the condition

$$P \in \mathcal{P}^*, u_i^* \neq u_j^* \in P : i < j \implies u_i^* \leq_P u_j^*,$$

and so D forms a compatible sequence of sinks.

Now we define the collection of paths in compatible sequence of sinks as follows.

$$\begin{aligned} \mathcal{P}_0^* &= \{P_{[s \rightarrow u_0^*]}\} \\ \mathcal{P}_i^* &= \mathcal{P}_{i-1}^* \cup \{P_{[s \rightarrow u_i^*]}\} \quad \text{for } i = 1, \dots, n-2. \end{aligned}$$

As in Kappmeier (2015), the network topology $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$ for $i = 0, 1, \dots, n-2$ containing the paths starting from s and ending at u_i^* forms the abstract network satisfying the switching property.

Observation 3.1 (Kappmeier (2015)). For an abstract network $\Pi = (\mathcal{E}, \mathcal{P})$ and a compatible sequence of sinks $u_0^*, u_1^*, \dots, u_{n-2}^*$, the abstract path-system $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$ for each $i = 0, 1, \dots, n-2$ is an abstract network.

Lastly, the solution obtained in this network is transformed to the original network by removing dummy elements and dummy paths. Flows to dummy elements are shifted to their corresponding intermediate elements.

Example 3.1. Consider a network Π presented in Figure 3.1 having storage capacity at each element together with movement capacity and cost in between pair of elements. The set of paths in Π is $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ where, $P_1 = (s, u, x, y, t)$, $P_2 = (s, w, x, v, t)$, $P_3 = (s, u, v, t)$, $P_4 = (s, w, y, t)$, $P_5 = (s, u, x, v, t)$ and $P_6 = (s, w, x, y, t)$. Let P_1, P_2, P_3 and P_4 be four paths with positive flow. Here, x is common element in paths P_1 and P_2 whose switched paths are $P_5 = P_1 \times_x P_2$ and $P_6 = P_2 \times_x P_1$. The shortest distance of each intermediate element is $d_{P[s \rightarrow y]} = 4$, $d_{P[s \rightarrow v]} = 3$, $d_{P[s \rightarrow x]} = 2$, $d_{P[s \rightarrow w]} = 1$ and $d_{P[s \rightarrow u]} = 0$. Thus the priority order of elements is $t \succ y \succ v \succ x \succ w \succ u$. We create the dummy element of each intermediate element together with the dummy path having movement capacity equal to the storage capacity of original element and taking cost on dummy path as zero. Also, the storage capacity of dummy element is taken as the storage capacity of original element. The set of dummy elements $\{y^*, v^*, x^*, w^*, u^*\}$ together with sink element forms a compatible sequence denoted by $D = \{t, y^*, v^*, x^*, w^*, u^*\}$ (See in Figure 3.2). \square

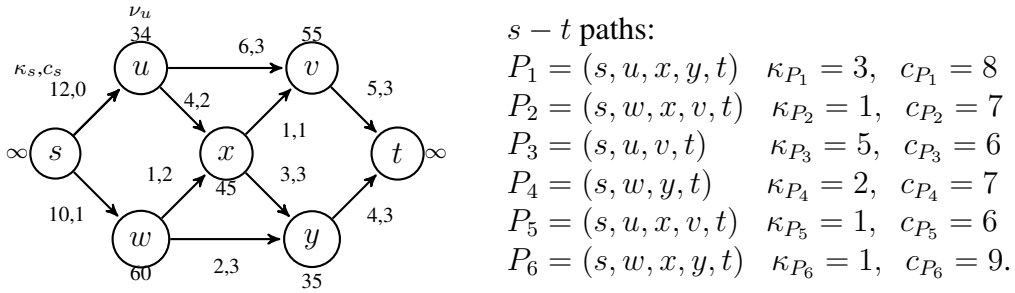


Figure 3.1: Network with movement capacity, cost between elements and storage capacity at elements.

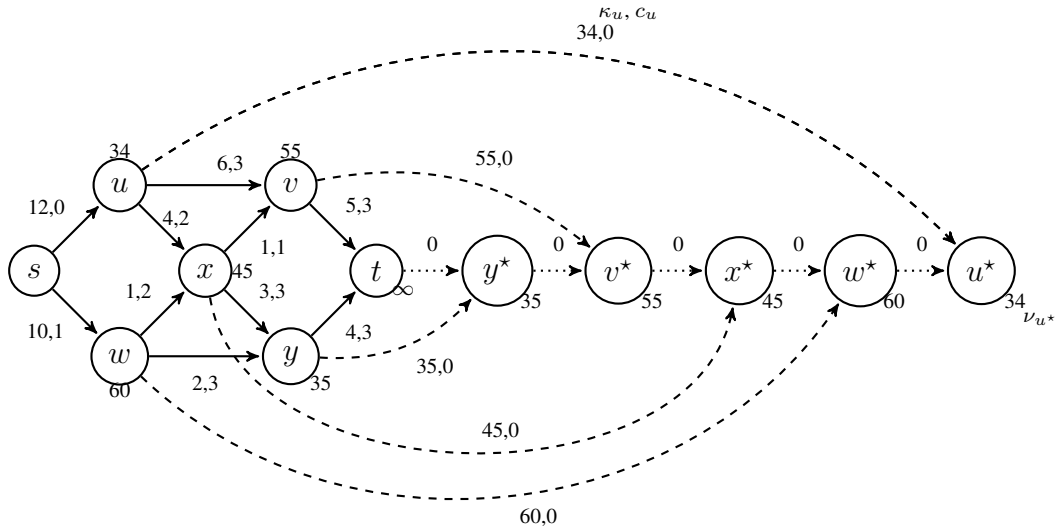


Figure 3.2: Reconfigured network with a compatible sequence of sinks (dummy elements) after priority ordering on Figure 3.1.

Using the lexicographically maximum flow algorithm of Kappmeier (2015) (cf. Page 167, Algorithm 6.2), we present Algorithm 8 to the abstract maximum static flow problem with intermediate storage on single source multi-sink abstract network Π^* as follows.

Algorithm 8: Abstract maximum static flow algorithm with intermediate storage

Input : Given abstract static network $\Pi = (\mathcal{E}, \mathcal{P})$.

Output: Abstract maximum static flow with intermediate storage on Π .

1. For each $u \in \mathcal{I}$ with $\nu_u \geq \sum_{P \in \mathcal{P}: v < P u} \kappa_v$, compute the shortest distance $d_{P_{[s \rightarrow u]}}$ by using Dijkstra's algorithm.
 2. Fix the priority order as $t \succ u_1 \succ \dots \succ u_{n-2}$ with first priority to the sink t and priority for intermediate elements as farther in distance higher in priority order.
 3. Construct the modified network $\Pi^* = (\mathcal{E}^*, \mathcal{P}^*)$ with single source s and compatible sequence of multiple sinks with dummy elements $D = \{t = u_0^*, u_1^*, \dots, u_{n-2}^*\}$, where $\mathcal{E}^* = \mathcal{E} \cup D$ and $\mathcal{P}^* = \mathcal{P} \cup \{P_{[s \rightarrow u_i^*]}\}$.
 4. Compute the lexicographic abstract maximum static flow with priority ordering in Step (2) according to Kappmeier (2015).
 5. Transform the solution to the original network Π by removing dummy elements and dummy paths.
-

Lexicographically Maximum Flow. For any two flows ϕ and ψ , we say that ϕ is lexicographically greater than ψ , and denote $\phi \geq^L \psi$, if either $(\phi_{u^*}^{in})_l > (\psi_{u^*}^{in})_l$ and $(\phi_{u^*}^{in})_{j-1} = (\psi_{u^*}^{in})_{j-1}$ holds for some $l \in \{0, 1, \dots, n-2\}$ and $j = 1, \dots, l$, or $(\phi_{u^*}^{in})_j = (\psi_{u^*}^{in})_j$ holds for all $j = 0, 1, \dots, n-2$. The maximum flow $\bar{\phi}$ with lexicographic order \geq^L among all feasible abstract flows ϕ is a lexicographic abstract maximum flow and is denoted by $\bar{\phi} \geq^L \phi$ for all ϕ .

Theorem 3.1. For each $i = 0, 1, \dots, n-2$, there exists an abstract flow ϕ_i in $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$ which is a lexicographically maximum flow.

Proof. When $i = 0$, $\Pi_0^* = \Pi$ is a single source single sink abstract network. As in McCormick (1996), it provides an abstract maximum flow by taking the initial flow as zero flow and using an augmenting structure. The mathematical induction method is used for further proof. For some $i < n-2$, assume that flow ϕ_i is lexicographic abstract maximum flow which is obtained by taking initial flow ϕ_{i-1} and using augmenting structure of McCormick's algorithm. We aim to show that ϕ_{i+1} is also lexicographic abstract maximum flow. As the sequence $u_0^*, u_1^*, \dots, u_{n-2}^*$ of sinks is compatible, augmenting structure of McCormick assures that the inflow to the sink element is not reduced and so ϕ_{i+1} is maximum. If possible, let us assume that ϕ_{i+1} is not lexicographic abstract maximum flow in the abstract network $\Pi_{i+1}^* = (\mathcal{E}^*, \mathcal{P}_{i+1}^*)$. Then there exists a flow ϕ' which sends more flow to the sink u_r^* for some $r \in \{0, 1, \dots, i\}$. For each $P \in \mathcal{P}_i^*$, define the restricted flow $\tilde{\phi}$ by $\tilde{\phi}^P = \phi'^P$. For $\tilde{\phi}$ and ϕ' , incoming flow at sink u_r^* is the same and $\tilde{\phi}$ is a feasible abstract flow in $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$ which sends more flow to sink u_r^* than ϕ_i . This contradicts to ϕ_i being a lexicographically maximum. \square

Theorem 3.2. *The static flow obtained from Algorithm 8 is an abstract maximum static flow with intermediate storage in $\Pi = (\mathcal{E}, \mathcal{P})$.*

Proof. Steps 1 and 2 of Algorithm 8 are as same in Section 2.3. For the network with abstract path-system $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$, the set inclusion holds i.e., $\Pi_i^* \subseteq \Pi_{i+1}^*$, for $i = 0, 1, \dots, n - 3$. Theorem 3.1 assures the existence of a lexicographic abstract maximum flow in Π_i^* . Lastly, we transform the flow of dummy elements in Π^* to their respective intermediate elements to obtain the solution with intermediate storage in Π and transformation of the network to original network is made by removing dummy elements and dummy paths. Thus the solution obtained is an abstract maximum static flow with intermediate storage in $\Pi = (\mathcal{E}, \mathcal{P})$. \square

Corollary 3.3. *Abstract maximum static flow problem with intermediate storage can be solved in polynomial time by using Algorithm 8.*

Proof. The shortest distance of each element can be obtained in $O(|\mathcal{E}|^2)$ times and their priority ordering with respect to the distance can be calculated in linear time. After fixing the priority order of intermediate elements, the problem is transformed to a single source and multi-sink problem and by using Kappmeier (2015), Step 4 can be obtained in polynomial time. Similarly, a transformation of the solution to the original network can be obtained in linear time. So, the polynomial time solution of Algorithm 8 to solve an abstract maximum static flow problem with intermediate storage is at hand. \square

Example 3.2. This example is continuation of Example 3.1 which proceeds to find the static solution with intermediate storage before and after switching of the paths. The set of dummy elements $D^* = \{y^*, v^*, x^*, w^*, u^*\}$ and a compatible sequence $D = D^* \cup \{t\}$ are obtained in Example 3.1. Here, we obtain the solution of maximum static flow by using lexicographic approach and restore the flows at sink and dummy elements with priority order. Finally, dummy elements and dummy paths are removed to get a maximum static flow with intermediate storage.

The flow with intermediate storage in general network (without switching of paths) is presented in Figure 3.3, where possible paths of flow transmission are P_1, P_2, P_3, P_4, P_5 and P_6 . The numbers in between the elements represent the movement capacity, flow and cost of the path segment. Similarly, in reconfigured abstract network flow is transmitted through four paths P_3, P_4, P_5 and P_6 after switching of two crossing paths P_1 and P_2 (see in Figure 3.4 and Table 3.1). Being minimum cost path, flow of 5 units is first send to the sink via path $P_3 = (s, u, v, t)$, where compatible sequence of sink and intermediate elements is $\{t, v^*, u^*\}$. Now, possible maximum excess flow is send to v^* through two possible paths (s, u, v, t, v^*) and (s, u, v, v^*) . As the first path (s, u, v, t, v^*) is already saturated, only 1 unit of excess flow can be reached to v^* . Similarly, the 4 units flow at u is diverged towards more prioritized element x , 2 units of excess flow is stored at u^* . This process is simultaneously used for all possible paths. At last, the flow at each dummy element is transformed to the respective element.

It is to be noted in Figure 3.4 that the 5 units of flow reaching x through two paths (4 from u and 1 from w) are not been merged at x but diverged to non-crossing sides by some traffic signal mechanism; and excess flows are stored at appropriate shelter. Otherwise, it becomes a general network flow. Out of the 4 units of flow reaching x from u , 1 unit is switched to v due to the switching property, and the remaining 3 units are stored as excess flow. On the other hand, the 1 unit of flow reaching x from w is switched to y with no excess flow. Thus, two flows from different paths are not crossing at x but diverging from the intersection.

The total amount of flow out from the source element before switching the paths (i.e. general network) without intermediate storage is 9 units whereas with intermediate storage is 22 units (Figure 3.5(a)). Figure 3.5(b) shows the flow at sink and excess flow at intermediate elements after switching the paths. Total amount of flow out from the source element without intermediate storage is 8 units whereas with intermediate storage is 22 units, which emphasize the importance of intermediate storage in the abstract network. Table 3.1 represents the flow pattern in each path with intermediate storage. \square

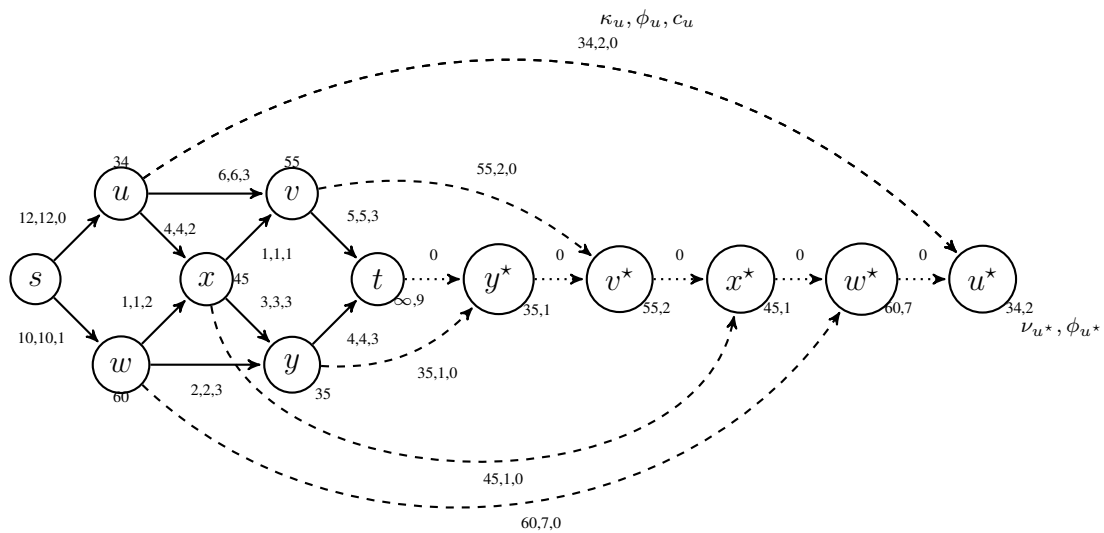


Figure 3.3: Storage of flow at dummy elements and sink before switching the paths (general).

Table 3.1: General and abstract path flows with intermediate storage

Intermediate storage: General						Intermediate storage: Abstract							
Path	u	w	x	v	y	t	Path	u	w	x	v	y	t
P_3	2	\times	\times	1	\times	5	P_3	2	\times	\times	1	\times	5
P_5	0	\times	0	1	\times	0	P_5	0	\times	3	1	\times	0
P_2	\times	7	1	0	\times	0	P_4	\times	7	\times	\times	0	2
P_4	\times	0	\times	\times	0	2	P_6	\times	0	0	\times	0	1
P_1	0	\times	0	\times	1	2							
P_6	\times	0	0	\times	0	0							
Total flow	2	7	1	2	1	9	Total flow	2	7	3	2	0	8

\times = element not used

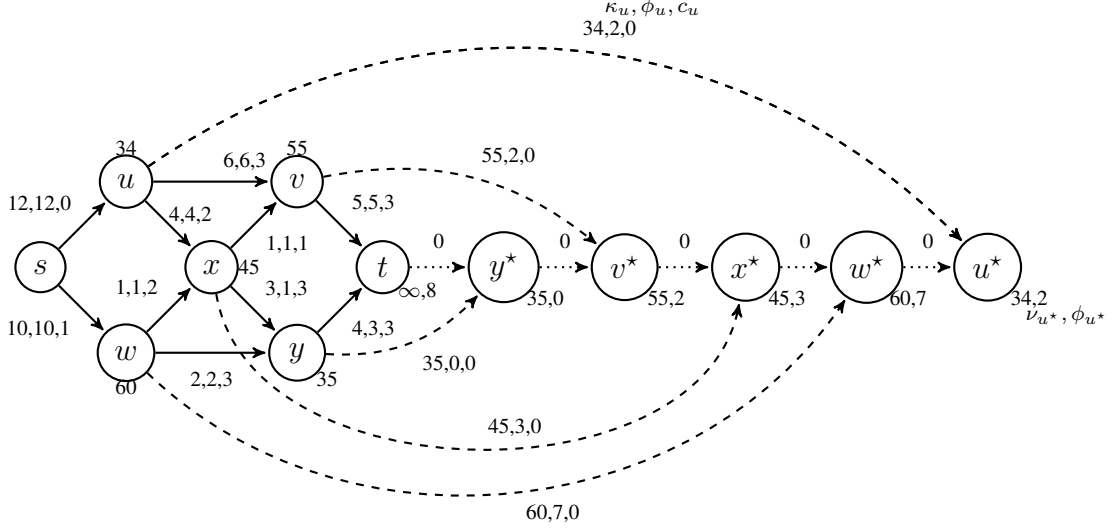


Figure 3.4: Storage of flow at dummy elements and sink after switching the paths (abstract).

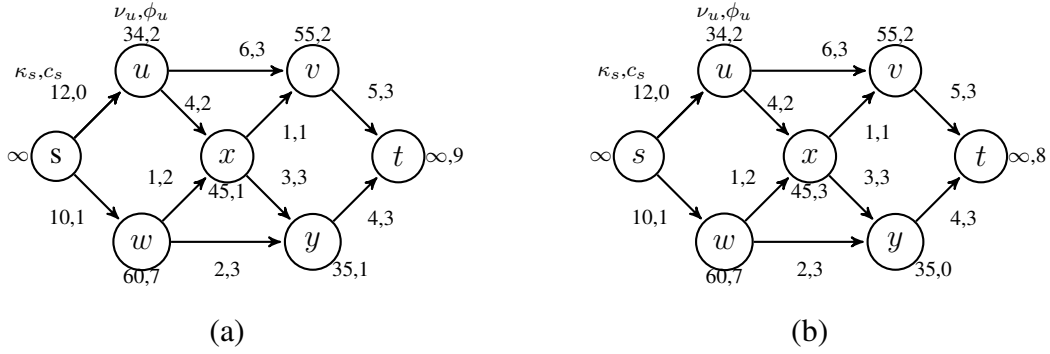


Figure 3.5: Solution in (a) before switching of paths and (b) after switching of paths.

3.1.3 Lexicographic Abstract Maximum Static Flow with Intermediate Storage

In this subsection, we consider a network with single source s but multiple sink $D = \{t_1, \dots, t_l\}$ and solve the lexicographic abstract maximum static flow problem with intermediate storage. Let $\Pi = (\mathcal{E}, \mathcal{P})$ be an abstract $s - D$ network with $\mathcal{E} = \{s\} \cup \mathcal{I} \cup D$, where D be a set of multiple sinks and other components have their usual meaning as defined previously.

Solution Procedure. Due to multiple sinks in given network, we first fix the priority of sinks with a compatible sequence. Secondly, the priority order of intermediate elements are set as in previous subsection and finally, we merge them with a priority of sinks followed by priority of intermediate elements. Three stages for fixing the priority order are as follows.

Stage 1: For a given set of sinks $D = \{t_1, t_2, \dots, t_l\}$ with $d_{P[s \rightarrow t_1]} > d_{P[s \rightarrow t_2]} > \dots > d_{P[s \rightarrow t_l]}$, the priority ordering of sinks is fixed as $t_1 \succ t_2 \succ \dots \succ t_l$ which respect their rankings. In case of equal distance, priority can be set arbitrarily. Similarly, if the priority order is pre-defined, it

can be adopted with the given order.

Stage 2: for the set of intermediate elements $\{u_1, u_2, \dots, u_k\}$ in \mathcal{I} with $k = n - l - 1$, the priority order is set as in previous subsection with farther in distance higher in priority order, i.e., $d_{P_{[s \rightarrow u_i]}} > d_{P_{[s \rightarrow u_j]}} \implies u_i \succ u_j, \forall i, j \leq k$ which respect their rankings. Equality in distance indicates the mutability in order.

Stage 3: In this stage, we first merge two sets in Stage 1 and Stage 2 with priority order $t_1 \succ \dots \succ t_l \succ u_1 \succ \dots \succ u_k$. The set of dummy elements is created as $\{u_{l+1}^*, \dots, u_{l+k}^*\}$ corresponding to $\{u_1, \dots, u_k\}$ with the same priority order of Stage 2. Let us construct a super sink $D' = \{t_1, \dots, t_l, u_{l+1}^*, \dots, u_{l+k}^*\}$ with a compatible sequence of elements having priority order as $t_1 \succ \dots \succ t_l \succ u_{l+1}^* \succ \dots \succ u_{l+k}^*$.

As in previous subsection, we define the collection of paths as

$$\begin{aligned} \mathcal{P}_0^* &= \emptyset \\ \mathcal{P}_i^* &= \mathcal{P}_{i-1}^* \cup \{P_{[s \rightarrow t_i]}\} \quad \text{for } i = 1, \dots, l \\ \mathcal{P}_i^* &= \mathcal{P}_{i-1}^* \cup \{P_{[s \rightarrow u_i^*]}\} \quad \text{for } i = l + 1, \dots, l + k. \end{aligned}$$

Each abstract path-system $\Pi_i^* = (\mathcal{E}^*, \mathcal{P}_i^*)$ for $i = 0, 1, \dots, l + k$ forms the abstract network satisfying the switching property, where $\mathcal{E}^* = \mathcal{E} \cup \{u_{l+1}^*, \dots, u_{l+k}^*\}$. The algorithmic framework to solve the lexicographic abstract maximum static flow problem with intermediate storage is as follows.

Algorithm 9: Lexicographic abstract maximum static flow algorithm with intermediate storage

Input : Given abstract static network $\mathcal{N} = (E, \mathcal{P})$.

Output: Lexicographic abstract maximum static flow with intermediate storage.

1. Fix the priority order $t_1 \succ \dots \succ t_l \succ u_{l+1}^* \succ \dots \succ u_{l+k}^*$ as described in three stages.
 2. Construct the modified network $\Pi^* = (\mathcal{E}^*, \mathcal{P}^*)$ with single source s and compatible sequence of super sinks $D' = \{t_1, \dots, t_l, u_{l+1}^*, \dots, u_{l+k}^*\}$.
 3. Compute the lexicographic abstract maximum static flow with priority ordering in Step 1 according to Kappmeier (2015) with similar procedure of Algorithm 8.
 4. Transform the solution to the original network Π by removing dummy elements and dummy paths.
-

The proof of existence of lexicographic abstract maximum flow in Π_i^* on super sink D' of sinks together with dummy elements and the polynomial time complexity of Algorithm 9 can be proved as in Subsection 3.1.2.

Theorem 3.4. *Algorithm 9 computes the lexicographic abstract maximum static flow with intermediate storage in $\Pi = (\mathcal{E}, \mathcal{P})$ in polynomial time complexity.*

3.1.4 Abstract Maximum Dynamic Flow with Intermediate Storage

Consider an abstract dynamic network topology $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$ with temporal dimensions $\tau : \mathcal{E} \times \mathcal{E} \rightarrow \mathbb{Z}^+$ and $T \in \mathcal{T}$ as a non-negative transit time of element $u \in \mathcal{E}$ that is necessary to transship flow from u to its right element and time horizon in discrete time setting, respectively. If u and v are two consecutive elements on path P with $u <_P v$, then flow traveling through u at time θ reaches v at time $\theta + \tau_u$. For each $s-t$ path $P \in \mathcal{P}$, the traversal time of the flow from s to t is represented as $\tau_P = \sum_{u \in P_{[s \rightarrow t]}} \tau_u$ and the traversal time of flow from s to intermediate element v through the path $P_{[s \rightarrow v]}$ is represented as $\tau_{P_{[s \rightarrow v]}} = \sum_{u \in P_{[s \rightarrow v]}} \tau_u$.

Mathematical Model. Consider an abstract dynamic path flow function $\Phi^P(\theta) : P \times T \rightarrow \mathbb{R}^+$ that sends the flow from s to t as at discrete time $\theta \in \mathcal{T}$ and an excess flow function $\hat{\Phi}_u(\theta) : \mathcal{I} \times T \rightarrow \mathbb{R}^+$ that stores the flow at intermediate element $u \in \mathcal{I}$ within time $\theta \in \mathcal{T}$. We denote the total outflow from u and inflow into u by $\Phi_u^{P,out} = \sum_{P \in \Gamma_u^{out}} \Phi^P$ and $\Phi_u^{P,in} = \sum_{P \in \Gamma_u^{in}} \Phi^P$, respectively, whose difference gives the excess flow. The linear programming formulation of abstract dynamic flow with intermediate storage is as follows.

$$\max \sum_{P \in \mathcal{P}} \sum_{\theta = \tau_P}^T \Phi^P(\theta) + \sum_{u \in \mathcal{I}} \sum_{\theta = \tau_{P_{[s \rightarrow u]}}}^T \hat{\Phi}_u(\theta) \quad (3.2a)$$

such that,

$$\sum_{P \in \mathcal{P}: u \in P} \Phi^P(\theta) \leq \kappa_u, \quad \forall u \in \mathcal{E}, \theta \in \mathcal{T} \quad (3.2b)$$

$$\Phi_u^{P,in}(\theta) - \Phi_u^{P,out}(\theta) = \hat{\Phi}_u(\theta), \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (3.2c)$$

$$0 \leq \hat{\Phi}_u(\theta) \leq \nu_u, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (3.2d)$$

$$\Phi^P \geq 0, \quad \forall P \in \mathcal{P} \quad (3.2e)$$

Our objective in equation (3.2a) is to maximize the sum of two flows, the flow reaching at sink t and the excess flow stored at intermediate elements $u \in \mathcal{I}$, within time horizon T . The capacity constraints and excess flow are represented by equation (3.2b) and equation (3.2c), respectively. Similarly, weak flow conservation at each time step θ and the boundedness of excess flow by storage capacity are presented in the left and right inequalities of equation (3.2d), respectively. Equation (3.2e) represents the non-negativity of the flow on each path. For the existence of solution, the lower (i.e., necessary) and upper (i.e., sufficient) bounds of the storage capacity for each intermediate element u is taken as $\sum_{P \in \mathcal{P}: v <_P u} \kappa_v \leq \nu_u \leq T \sum_{P \in \mathcal{P}: v <_P u} \kappa_v$.

Solution Procedure. To start the solution procedure, we first have to describe the formation of temporal paths in time expanded form. As similar to general network, elements in the time expanded network \mathcal{E}^T are obtained by creating $T + 1$ copies of the elements for each time step

$\theta \in \mathcal{T}$ and defined as

$$\mathcal{E}^T = \{u(\theta) : u \in \mathcal{E}, \theta \in \mathcal{T}\},$$

where $\{u(0)\}$ represents the set of elements in the given network. Any flow starting from source element s at the time θ reaches to u along with a path P at time $\theta + \sum_{v \in P_{[s \rightarrow u]}} \tau_v$. For each path $P \in \mathcal{P}$ and $\theta \in \{0, 1, \dots, T\}$, the temporal path $P(\theta)$ is the copy of elements of P in which flow starts on it at the time θ and travels through path P . That is,

$$P_{[s \rightarrow u]}(\theta) = \left\{ u(\beta) \in \mathcal{E}^T : u \in P, \beta = \theta + \sum_{v \in P_{[s \rightarrow u]}} \tau_v \right\}.$$

Replacing the arbitrary element u by sink element t provides the source-sink temporal path and is denoted simply by $P(\theta)$. The order of elements in temporal path $P(\theta)$ is same as in P . We denote the set of all temporal paths that reach to the intermediate element u and sink element t within time horizon T by $\mathcal{P}_{[s \rightarrow u]}^T(\theta)$ and $\mathcal{P}^T(\theta)$, respectively, which are defined as follows.

$$\begin{aligned} \mathcal{P}_{[s \rightarrow u]}^T(\theta) &= \left\{ P_{[s \rightarrow u]}(\theta) : P_{[s \rightarrow u]} \subset P \in \mathcal{P}, \theta \in \mathcal{T}, \theta + \sum_{v \in P_{[s \rightarrow u]}} \tau_v \leq T \right\} \\ \mathcal{P}^T(\theta) &= \left\{ P(\theta) : P \in \mathcal{P}, \theta \in \mathcal{T}, \theta + \sum_{u \in P} \tau_u \leq T \right\}, \end{aligned}$$

The abstract path system $(\mathcal{E}^T, \mathcal{P}^T(\theta))$ may not be an abstract network because it may not satisfy the switching property (Kappmeier (2015)). To handle this problem, paths with delay in elements are essential. We define the delay function $\delta : P \rightarrow \{0, 1, \dots, T\}$ for each element in path P . Every flow traveling from s along with path P with delay pattern δ reaches to $u \in P$ at time $\sum_{v \in P_{[s \rightarrow u]}} (\tau_v + \delta_v) + \delta_u$. The temporal path with delay pattern can be represented as

$$P_{[s \rightarrow u]}^\delta = \left\{ u(\beta) \in \mathcal{E}^T : u \in P, \beta = \sum_{v \in P_{[s \rightarrow u]}} (\tau_v + \delta_v) + \delta_u \right\}.$$

The order of elements in P^δ is the same as in P . Now, we represent the set of temporal paths with delay pattern δ arriving at the intermediate element u and destination sink t within time T by $\mathcal{P}_{[s \rightarrow u]}^{\delta, T}$ and $\mathcal{P}^{\delta, T}$, respectively, which can be defined as follows.

$$\begin{aligned} \mathcal{P}_{[s \rightarrow u]}^{\delta, T} &= \left\{ P_{[s \rightarrow u]}^\delta : P_{[s \rightarrow u]} \subset P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{v \in P_{[s \rightarrow u]}} (\tau_v + \delta_v) \leq T \right\} \\ \mathcal{P}^{\delta, T} &= \left\{ P^\delta : P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{u \in P} (\tau_u + \delta_u) \leq T \right\}. \end{aligned}$$

Example 3.3. Consider a network with transit times between the elements presented in Figure 3.6 in which a set of $s-t$ paths is $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$, where $P_1 = s-u-t$, $P_2 = s-v-t$, $P_3 = s-u-v-t$ and $P_4 = s-v-u-t$. Let $T = 4$ be given time horizon. Then, the temporal $s-t$ paths $P(\theta)$ with starting time θ from s are as follows:

$$\begin{aligned} P_1(0) &= s(0) - u(2) - t(3), \\ P_1(1) &= s(1) - u(3) - t(4), \\ P_2(0) &= s(0) - v(1) - t(2), \\ P_2(1) &= s(1) - v(2) - t(3), \\ P_2(2) &= s(2) - v(3) - t(4), \\ P_3(0) &= s(0) - u(2) - v(3) - t(4) \\ P_4(0) &= s(0) - v(1) - u(2) - t(3) \\ P_4(1) &= s(1) - v(2) - u(3) - t(4) \end{aligned}$$

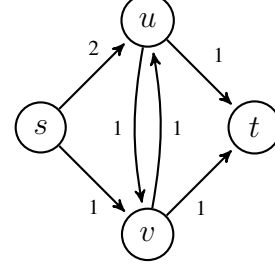


Figure 3.6: Dynamic network with transit time.

Here, two temporal paths $P_3(0)$ and $P_4(0)$ are crossing at u at $\theta = 2$. By applying the switching property, paths are switched along $s(0) - u(2) - t(3)$ and $s(0) - v(1) - u(2) - v(3) - t(4)$. The second switched path does not belong to the path set which also forms a cycle $v(1) - u(2) - v(3)$. The cycle can be removed by taking path as $s(0) - v(1) - t(4)$ but still it is not an abstract path because the transit time from v to t is not 3 units but 1 unit. Thus we have to assign the waiting time $\delta_v = 2$ at v to form an abstract path.

Lemma 3.5 (Kappmeier (2015)). (i) For $u \in P \cap Q$ and $R = P \times_u Q$, an abstract network is said to preserve the order of path if $v, w \in R \cap P_{[s \rightarrow u]}$ with $v <_P w$ implies $v <_R w$ and if $v, w \in R \setminus P_{[s \rightarrow u]}$ with $v <_Q w$ implies $v <_R w$.

(ii) If an abstract network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$ preserves the order on paths, then the network $\Pi^{\delta, T} = (\mathcal{E}^T, \mathcal{P}^{\delta, T}, \tau, T)$ with path system $\mathcal{P}^{\delta, T}$ is an abstract network.

As in static case, the compatible set of dummy elements $D = \{t = u_0^*, u_1^*, \dots, u_{n-2}^*\}$ with same priority ordering $u_0^* \succ u_1^* \succ \dots \succ u_{n-2}^*$ is obtained and the problem is transformed to a single source multi-sink $s - D$ flow problem in abstract network. We use the lexicographic maximum dynamic flow solution procedure of Kappmeier (2015) in the reconfigured network $\Pi_i^{\delta, T} = (\mathcal{E}_i, \mathcal{P}_i^{\delta, T})$ with $\mathcal{E}_i = \mathcal{E} \cup D_i$, $D_i = \{u_0^*, u_1^*, \dots, u_i^*\}$ and $\mathcal{P}_i^{\delta, T} = \mathcal{P}^{\delta, T} \cup \mathcal{P}_{[s \rightarrow u_i^*]}^{\delta, T}$ for all $i = 0, 1, \dots, n-2$ and for each time step $\theta \in \mathcal{T}$. Here, $D_0 \subseteq \dots \subseteq D_{n-2}$ and similar set inclusion holds in $\mathcal{P}_i^{\delta, T}$. The algorithmic framework to solve abstract maximum dynamic flow with intermediate storage is presented in Algorithm 10.

Theorem 3.6. Algorithm 10 provides an optimal solution to the abstract maximum dynamic flow problem with intermediate storage.

Proof. While prioritizing the intermediate elements, constructing the modified network with dummy elements, constructing the reconfigured network with $D_0 \subseteq \dots \subseteq D_{n-2}$ and $\Pi_i^{\delta, T} \subseteq \Pi_{i+1}^{\delta, T}$ for $i = 0, \dots, n-3$, the movement capacity and the flow conservation constraints are not violated. Using time expanded temporal paths with time horizon T , flow at sink $t = u_0^*$ is

Algorithm 10: Abstract maximum dynamic flow algorithm with intermediate storage

Input : Given abstract dynamic network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$.

Output: Abstract maximum dynamic flow with intermediate storage on Π .

1. For each $u \in \mathcal{I}$ with $\nu_u \geq \sum_{P \in \mathcal{P}: v < P u} \kappa_v$, compute the shortest distance $d_{P_{[s \rightarrow u]}}$ by using Dijkstra's algorithm.
 2. Fix the priority order $t = u_0 \succ u_1 \succ \dots \succ u_{n-2}$ with first priority to the sink $t = u_0$ and priority of intermediate elements with farther in distance from source higher in priority.
 3. Construct the modified network $\Pi^* = (\mathcal{E}^*, \mathcal{P}^*)$ with single source s and compatible sequence of multiple sinks with dummy elements $D = \{u_0^*, u_1^*, \dots, u_{n-2}^*\}$, where $\mathcal{E}^* = \mathcal{E} \cup D$ and $\mathcal{P}^* = \mathcal{P} \cup \{P_{[s \rightarrow u_{n-2}^*]}\}$.
 4. Set $D_i = \{u_0^*, u_1^*, \dots, u_i^*\}$ for all $i = 0, 1, \dots, n-2$ so that $D_0 \subseteq \dots \subseteq D_{n-2}$.
 5. Construct the reconfigured network $\Pi_i^{\delta, T} = (\mathcal{E}_i, \mathcal{P}_i^{\delta, T})$ with $\mathcal{E}_i = \mathcal{E} \cup D_i$ and $\mathcal{P}_i^{\delta, T} = \mathcal{P}^{\delta, T} \cup \mathcal{P}_{[s \rightarrow u_i^*]}^{\delta, T}$ for all $i = 0, 1, \dots, n-2$.
 6. For $\theta = 0, 1, \dots, T$:
 - For $i = 0, 1, \dots, n-2$:
 - Compute the lexicographic abstract maximum static flow with priority ordering of Step 2 in $\Pi_i^{\delta, T} = (\mathcal{E}_i, \mathcal{P}_i^{\delta, T})$ using Kappmeier (2015).
 7. Transform the solution to the original network Π by removing dummy elements and dummy paths.
-

obtained. At the same time, the excess flows on prioritized intermediate elements are stored with respect to their priority order. The next priority is given to element u_1^* in which flow through path $\mathcal{P}_{[s \rightarrow u_1^*]}^{\delta, T}$ is shifted to u_1^* satisfying the storage capacity constraints within time T together with storing excess flow on the rest of the prioritized intermediate elements along the paths. The process is continued in successive order of prioritized elements u_i^* as long as all elements have storage capacity and sufficient time to reach the flow. In each iteration of Step 6, feasible flow from s to u_i^* is executed within time horizon T in lexicographic order. Thus the flow obtained from Algorithm 10 is feasible. The optimality of algorithm is assured by the optimality of Step 6. Because of the lexicographic abstract maximum flow being optimal, Algorithm 10 provides the optimal abstract maximum dynamic flow with intermediate storage. \square

Lemma 3.7. *Algorithm 10 computes the abstract maximum dynamic flow with intermediate storage in polynomial time complexity.*

Proof. The shortest distance in Step 1 of Algorithm 10 can be computed in $O(|\mathcal{E}|^2)$ time where as providing the priority order of elements, arranging them in a compatible sequence and reformation of a network in Steps 2–5 and Step 7 can be obtained in linear time. As in Kappmeier et al. (2014), the time complexity of abstract maximum dynamic flow at sink t is $\mu(|\mathcal{E}|, \log(U), \log(T)) \cdot \mathcal{O}(P)$, where μ is a polynomial, $U = \max_{u \in \mathcal{E}} \{\kappa_u\}$, and $\mathcal{O}(P)$ denotes the time needed for a call of the oracle \mathcal{O} for the abstract network path $P \in \mathcal{P}_i^{\delta, T}$. For $Q \subseteq \mathcal{E}$, an oracle \mathcal{O} returns path P with order $<_P$ for some $P \in \mathcal{P}_i^{\delta, T}$ such that $P \subseteq Q$ or verifies that there is no path contained in Q . Since the maximum flow is to be calculated at $(n-1)$ dummy

elements, Algorithm 10 solves an abstract maximum dynamic flow problem with intermediate storage within the time complexity of $(n - 1)[\mu(|\mathcal{E}|, \log(U), \log(T)) \cdot \mathcal{O}(P)]$. \square

Example 3.4. To find an abstract maximum dynamic flow with intermediate storage on the abstract network presented in Example 3.1, we consider the cost c as the transit time τ and time horizon $T = 10$. Using Algorithm 10, we sent the flow to the sink as the first priority for each time step θ and successively store the excess flow at intermediate elements along the path with respective priority order. The detail of which is illustrated in Table 3.2.

The total amount of flow reached at sink t in time $T = 10$ is 35 units whereas intermediate elements y , v and x store 9, 31 and 31 units, respectively. Due to insufficient storage capacity, flow at w through path $P_{[s \rightarrow w]}$ can store only 2 units and only 4 units of flow through $P_{[s \rightarrow u]}$ at time $\theta = 8$. Total amount of flow sent from s within time $T = 10$ is 200 units which is stored at different elements as follows: $\Phi_t = 35$, $\hat{\Phi}_y = 9$, $\hat{\Phi}_v = 31$, $\hat{\Phi}_x = 31$, $\hat{\Phi}_w = 60$ and $\hat{\Phi}_u = 34$. The detailed information (regarding paths and flow values) is given in Table 3.2.

Abstract s - t paths after switching:

$$P_5 = (s, u, x, v, t), \tau_{P_5} = 6, \Phi_{P_5} = 1$$

$$P_3 = (s, u, v, t), \tau_{P_3} = 6, \Phi_{P_3} = 4$$

$$P_4 = (s, w, y, t), \tau_{P_4} = 7, \Phi_{P_4} = 2$$

$$P_6 = (s, w, x, y, t), \tau_{P_6} = 9, \Phi_{P_6} = 1$$

Abstract intermediate paths after switching:

$$P_{[s \rightarrow v]}, P_{[s \rightarrow x]}, P_{[s \rightarrow u]}$$

$$P_{[s \rightarrow v]}, P_{[s \rightarrow u]}$$

$$P_{[s \rightarrow y]}, P_{[s \rightarrow w]}$$

$$P_{[s \rightarrow y]}, P_{[s \rightarrow x]}, P_{[s \rightarrow w]}$$

Here, the waiting pattern of each element is 0 because no two successive common elements appeared in crossing of paths. \square

Observation 3.2. As the sink element has sufficient storage capacity, the abstract maximum dynamic flow at the sink is obtained by using TRF along paths $P \in \mathcal{P}^{\delta, T}$. To calculate the flow at intermediate elements, flows in intermediate paths $P_{[s \rightarrow u_i^*]}^{\delta}$ may not be temporally repeated because flow value may change over time due to insufficient storage capacity (see Table 3.2). Thus the lexicographic maximum flow is essential.

Table 3.2: Abstract flow with intermediate storage in each time θ

Path	Start time at s	u	w	x	v	y	t	Reaching time at last element
P_5	$\theta = 0$	2	×	3	0	×	1	$\theta = 6$
P_5	$\theta = 1$	2	×	3	0	×	1	$\theta = 7$
P_5	$\theta = 2$	2	×	3	0	×	1	$\theta = 8$
P_5	$\theta = 3$	2	×	3	0	×	1	$\theta = 9$
P_5	$\theta = 4$	2	×	3	0	×	1	$\theta = 10$
$P_{[s \rightarrow v]}$	$\theta = 5$	2	×	3	1	×	×	$\theta = 8$
$P_{[s \rightarrow v]}$	$\theta = 6$	2	×	3	1	×	×	$\theta = 9$
$P_{[s \rightarrow v]}$	$\theta = 7$	2	×	3	1	×	×	$\theta = 10$
$P_{[s \rightarrow x]}$	$\theta = 8$	2	×	4	×	×	×	$\theta = 10$
$P_{[s \rightarrow u]}$	$\theta = 9$	6	×	×	×	×	×	$\theta = 9$
$P_{[s \rightarrow u]}$	$\theta = 10$	6	×	×	×	×	×	$\theta = 10$
P_3	$\theta = 0$	0	×	×	2	×	4	$\theta = 6$
P_3	$\theta = 1$	0	×	×	2	×	4	$\theta = 7$
P_3	$\theta = 2$	0	×	×	2	×	4	$\theta = 8$
P_3	$\theta = 3$	0	×	×	2	×	4	$\theta = 9$
P_3	$\theta = 4$	0	×	×	2	×	4	$\theta = 10$
$P_{[s \rightarrow v]}$	$\theta = 5$	0	×	×	6	×	×	$\theta = 8$
$P_{[s \rightarrow v]}$	$\theta = 6$	0	×	×	6	×	×	$\theta = 9$
$P_{[s \rightarrow v]}$	$\theta = 7$	0	×	×	6	×	×	$\theta = 10$
$P_{[s \rightarrow u]}$	$\theta = 8$	4	×	×	×	×	×	$\theta = 8$ storage full
P_4	$\theta = 0$	×	7	×	×	0	2	$\theta = 7$
P_4	$\theta = 1$	×	7	×	×	0	2	$\theta = 8$
P_4	$\theta = 2$	×	7	×	×	0	2	$\theta = 9$
P_4	$\theta = 3$	×	7	×	×	0	2	$\theta = 10$
$P_{[s \rightarrow y]}$	$\theta = 4$	×	7	×	×	2	×	$\theta = 8$
$P_{[s \rightarrow y]}$	$\theta = 5$	×	7	×	×	2	×	$\theta = 9$
$P_{[s \rightarrow y]}$	$\theta = 6$	×	7	×	×	2	×	$\theta = 10$
$P_{[s \rightarrow w]}$	$\theta = 7$	×	9	×	×	×	×	$\theta = 8$
$P_{[s \rightarrow w]}$	$\theta = 8$	×	2	×	×	×	×	$\theta = 9$ storage full
P_6	$\theta = 0$	×	0	0	×	0	1	$\theta = 9$
P_6	$\theta = 1$	×	0	0	×	0	1	$\theta = 10$
$P_{[s \rightarrow y]}$	$\theta = 2$	×	0	0	×	1	×	$\theta = 8$
$P_{[s \rightarrow y]}$	$\theta = 3$	×	0	0	×	1	×	$\theta = 9$
$P_{[s \rightarrow y]}$	$\theta = 4$	×	0	0	×	1	×	$\theta = 10$
$P_{[s \rightarrow x]}$	$\theta = 5$	×	0	1	×	×	×	$\theta = 8$
$P_{[s \rightarrow x]}$	$\theta = 6$	×	0	1	×	×	×	$\theta = 9$
$P_{[s \rightarrow x]}$	$\theta = 7$	×	0	1	×	×	×	$\theta = 10$
$P_{[s \rightarrow w]}$	$\theta = 8$	×	0	×	×	×	×	$\theta = 9$ storage full
Total flow stored		34	60	31	31	9	35	Total=200

× = element not used along the path

3.1.5 Abstract TRF with Intermediate Storage

To find the temporally repeated maximum dynamic flow with intermediate storage, the storage capacity of each prioritized intermediate element must be sufficient (at least upper bound, i.e., $\nu_u \geq T \sum_{P \in \mathcal{P}: v < P u} \kappa_v \quad \forall u \in \mathcal{I}$). The flow value can be obtained by using TRF on sink and intermediate elements through paths with waiting pattern δ as follows:

For $u_0 = t$ and $P \in \mathcal{P}^{\delta, T}$ with $\phi_P = \min\{\kappa_u : u \in P\}$,

$$|\Phi|_t = \sum_P (T - \tau_P + 1) \cdot \phi_P$$

For intermediate element $u_i \in \mathcal{I}$ and $P_{[s \rightarrow u_i]} \in \mathcal{P}_{[s \rightarrow u_i]}^{\delta, T} \subseteq \mathcal{P}^{\delta, T}$ with $u_i <_P u_j$,

$$|\hat{\Phi}|_{u_i} = \sum_{P_{[s \rightarrow u_i]}} [(T - \tau_{P_{[s \rightarrow u_j]}} + 1) \cdot \hat{\phi}_{u_i} + (\tau_{P_{[s \rightarrow u_j]}} - \tau_{P_{[s \rightarrow u_i]}}) \cdot \phi_{P_{[s \rightarrow u_i]}}]$$

where, $|\hat{\Phi}|_{u_i}$ is the net flow at element u_i within time T , $\phi_{P_{[s \rightarrow u_i]}} = \min\{\kappa_u : u \in P_{[s \rightarrow u_i]}\}$ and $\tau_{P_{[s \rightarrow u_i]}} = \sum_{u \in P_{[s \rightarrow u_i]}} (\tau_u + \delta_u)$. Similarly, $\hat{\phi}_{u_i}$ is the excess flow at u_i used on path flow balancing.

Before using the formula, we first find the abstract static path flow and excess flow at each element as described in Subsection 3.1.2. We first choose one of the $s - t$ path and balance the flow such that inflow at an element is equal to the sum of out flow and excess flow. The dynamic flow is obtained at each elements of the path by using above formula. The network is now updated by reducing the movement capacity and excess flow of elements used by this path. In updated network, next path is obtained, then dynamic flow is calculated and again the network is updated. This process continues until there exists any $s - t$ or $s - u_i$ path with priority order of elements having positive flow on path (cf. Section 2.3).

Example 3.5. By using the TRF in Example 3.4, total amount of flow that can be pushed from the source within time $T = 10$ is 232 units, where amount of flow stored at sink and intermediate elements are as follows: $\Phi_t = 35$, $\hat{\Phi}_y = 9$, $\hat{\Phi}_v = 31$, $\hat{\Phi}_x = 31$, $\hat{\Phi}_w = 78$ and $\hat{\Phi}_u = 48$. \square

Observation 3.3. For a given abstract static [dynamic] network $\Pi = (\mathcal{E}, \overleftrightarrow{\mathcal{P}})$ [$\Pi = (\mathcal{E}, \overleftrightarrow{\mathcal{P}}, \tau, T)$], the abstract maximum static [dynamic] contraflow problem with intermediate storage can be solved by sending the flow via $s - t$ paths $\overrightarrow{\mathcal{P}} \cup \overleftarrow{\mathcal{P}}$ at sink and allowing the storage of excess flow at intermediate elements u via paths $\overrightarrow{\mathcal{P}}_{[s \rightarrow u]} \cup \overleftarrow{\mathcal{P}}_{[u \rightarrow s]}$, $\forall u \in \mathcal{I}$ with storage capacity $\nu_u \geq \sum_{P \in \mathcal{P}: v < P u} \kappa_v$; $P' = \overrightarrow{\mathcal{P}} \cup \overleftarrow{\mathcal{P}}$ [within a given time horizon T] by reversing the direction of paths $\overleftarrow{\mathcal{P}}$ and $\overleftarrow{\mathcal{P}}_{[u \rightarrow s]}$ at time zero. The solution procedure can be found in detail in our paper Pyakurel et al. (2022).

3.2 Abstract Network Flow with Partial Switching

In section 3.1, we consider the abstract network in which flow on paths are switched completely. As abstract flow is very beneficial to reduce the congestion by crossing elimination, it may reduce the flow value while switching the direction of flow. In this subsection, we aim to introduce the concept of partial switching property, where the existence of residual path together with complete switched paths eliminate the crossing effect and increases the flow value.

As a motivational example of the partial switching, we consider a road network presented in Figure 3.7. Figure 3.7(i) represents the general network in which green and red paths from a are crossing with blue path from b at o . In Figure 3.7(ii), paths from a and b are switched to c and d , respectively, by using traffic signals, barricades, diversion, traffic lights or presence of traffic police, etc (see barrier line in Figure 3.7(ii)). Similarly, Figure 3.7(iii) represent the situation of partial switching in which red flow of original network remains same together with other switched paths. Network with complete switching sends 4 units of flows towards the destination. In general network flow pattern and partial switching pattern, 5 units of flows are heading towards the destination but major defect of general network flow is that how long the flows have to wait at crossing point. Thus, partial switching improves the flow by using complete switched paths together with an additional residual path (red path Figure 3.7(iii)), and reduces the congestion effect at crossings.

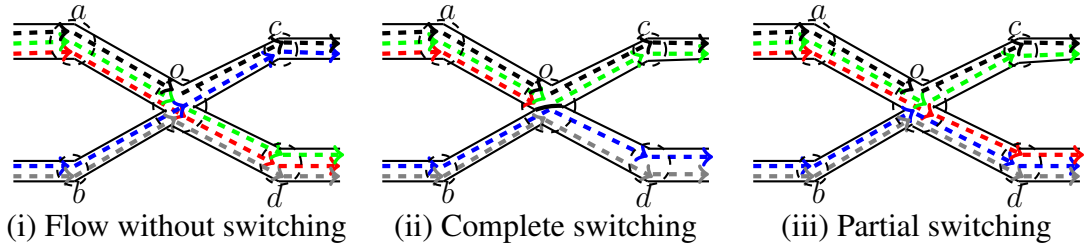


Figure 3.7: Flow pattern in three cases: without switching, with complete switching and with partial switching.

3.2.1 Abstract Static Flow with Partial Switching

The effect of complete switching property on an abstract network is that the abstract flow may not be equal to the general network flow because of the change in capacity and transit time of $s-t$ paths after switching. To recover the flow decrease by complete switching, the partial switching property is defined as follows: $\forall P, Q \in \mathcal{P}$ and intermediate element $u \in P \cap Q$, $\exists R_i \in \mathcal{P}$, $i = 1, 2, 3$ such that

$$R_1 \subseteq P \times_u Q = \{v \in P : s \leq_P v \leq_P u\} \cup \{v \in Q : u \leq_Q v \leq_Q t\}.$$

$$R_2 \subseteq Q \times_u P = \{v \in Q : s \leq_Q v \leq_Q u\} \cup \{v \in P : u \leq_P v \leq_P t\}.$$

and if the saturated path segments of R_1 and R_2 lie in the different sides of crossing element u , then $\exists R_3$, a residual path in residual network Π^r with residual capacity $\tilde{\kappa}_u$, such that

$$R_3 \subseteq P : \tilde{\kappa}_P > 0 \quad \text{or} \quad R_3 \subseteq Q : \tilde{\kappa}_Q > 0,$$

where, $\tilde{\kappa}_P = \min\{\tilde{\kappa}_u : u \in P\}$. Here, R_1 , R_2 and R_3 represent three modes of $s - t$ paths $P \times_u Q$, $Q \times_u P$ and residual path, respectively. Clearly, $\cup_{i=1}^3 R_i = \mathcal{R} \subseteq \mathcal{P}$ represent the collection of switched $s-t$ paths of three modes. If P be a $s - t$ path without crossing effect, then $P = P \times_u P = R_i, \forall u \in P$, and $i = 1, 2, 3$. Except in the two-way contraflow network, no path segments out going from the sink element and incoming to the source element exist. Hereafter, we denote $P \in \mathcal{R} \subseteq \mathcal{P}$ as a member of paths obtained by partial switching.

Flow Model. The linear programming flow model for abstract static network with partial switching can be presented as follows.

$$\max \sum_{P \in \mathcal{R}} \phi^P \tag{3.3a}$$

such that,

$$\sum_{P \in \mathcal{R}: u \in P} \phi^P \leq \kappa_u, \quad \forall u \in \mathcal{E} \tag{3.3b}$$

$$\phi_u^{P,in} - \phi_u^{P,out} = 0, \quad \forall u \in \mathcal{I}, P \in \mathcal{R} \tag{3.3c}$$

$$\phi^P \geq 0, \quad \forall P \in \mathcal{R} \tag{3.3d}$$

Equation (3.3a) is an objective function that maximizes the total flow reaching to the sink t by using partially switched paths. Equation (3.3b) represents the capacity constraint on each path segment and conservation of flow is represented by equation (3.3c). Similarly, the non-negativity of the flow on each partially switched path is represented by equation (3.3d).

Solution Procedure. To solve the abstract maximum static flow problem with partial switching of paths, we first compute the abstract paths with complete switching and obtain the maximum static flow by using McCormick (1996). To improve the flow value, we construct a residual network and find residual paths satisfying partial switching property. The algorithmic framework to solve the problem is presented hereafter in Algorithm 11.

Theorem 3.8. *Algorithm 11 provides an optimal solution to an abstract maximum static flow problem with partial switching in polynomial time.*

Proof. Due to satisfiability of flow conservation and capacity constraints, all steps of Algorithm 11 are feasible. As Step 2 provides optimal solution and Steps 3 and 4 compute optimal solution in residual paths, so Algorithm 11 provides an optimal abstract maximum static flow with partial switching. As proven in McCormick (1996), Step 2 can be computed in poly-

Algorithm 11: Abstract maximum static flow algorithm with partial switching

Input : Given abstract static network $\Pi = (\mathcal{E}, \mathcal{P})$ with partial switching.

Output: Abstract maximum static flow with partial switching on Π .

1. Initialize $\phi = \phi_0$ if initial flow is given, otherwise set initial flow as zero flow.
 2. Use augmenting structure to compute an abstract maximum static flow with complete switching using McCormick (1996).
 3. Construct a residual network Π^r by finding residual capacity of each path.
 4. Decompose the residual capacity into residual $s - t$ path flow satisfying the partial switching property.
-

mial time and the construction of residual network in Step 3 can be computed in $O(|\mathcal{E}^2|)$ time. Therefore, Algorithm 11 provides an optimal solution in polynomial time. \square

3.2.2 Abstract Dynamic Flow with Partial Switching

For a given abstract dynamic network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$, an abstract maximum dynamic flow problem with partial switching is to find the maximum flow leaving the source element that is to be sent to the sink via $s - t$ paths $P \in \mathcal{R} \subseteq \mathcal{P}$ by allowing the partial switching property within the given time horizon T .

Flow Model. for the dynamic $s - t$ path flow function $\Phi^P(\theta) : P \times T \rightarrow \mathbb{R}^+$ with partial switching, the linear program for abstract dynamic flow with partial switching is as follows.

$$\max \sum_{P \in \mathcal{R}} \sum_{\theta = \tau_P}^T \Phi^P(\theta) \quad (3.4a)$$

such that,

$$\sum_{P \in \mathcal{R}: u \in P} \Phi^P(\theta) \leq \kappa_u, \quad \forall u \in \mathcal{E}, \theta \in \mathcal{T} \quad (3.4b)$$

$$\Phi_u^{P,in}(\theta) - \Phi_u^{P,out}(\theta) \geq 0, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (3.4c)$$

$$\Phi_u^{P,in}(T) - \Phi_u^{P,out}(T) = 0, \quad \forall u \in \mathcal{I} \quad (3.4d)$$

$$\Phi^P \geq 0, \quad \forall P \in \mathcal{R} \subseteq \mathcal{P} \quad (3.4e)$$

Objective function in equation (3.4a) represents the maximization of the amount of flow transmitted from s to t within time horizon T . Equation (3.4b) is the capacity constraint of each path segment at $\theta \in \mathcal{T}$ and the non-negativity of the flow on each path is represented by equation (3.4e). Similarly, weak flow conservation at intermediate elements at time θ is presented in equation (3.4c), whereas the flow conservation at time T is presented in equation (3.4d).

Solution Procedure. To solve the problem, we first find the completely switched paths as well as residual paths P by using partial switching property on static network, where the transit

time is considered as cost. We construct the temporal paths from each partially switched paths P . As these paths may not be abstract, the paths with delay pattern $P \in \mathcal{R}^{\delta,T} \subseteq \mathcal{P}^{\delta,T}$ is essential. The temporally repeated abstract flow is obtained on $P \in \mathcal{R}^{\delta,T}$ which provides the abstract maximum dynamic flow with partial switching. This temporally repeated abstract flow is an abstract dynamic flow obtained from static flow ϕ by repeatedly sending in each paths $P \in \mathcal{R}^{\delta,T}$ as long as possible to reach the destination, i.e., up to the point in time $T - \tau_P$, Kappmeier et al. (2014). Here, we present an algorithm to solve the maximum dynamic flow problem with partial switching.

Algorithm 12: Abstract maximum dynamic flow algorithm with partial switching

Input : Given abstract dynamic network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$.

Output: Abstract maximum dynamic flow with partial switching on Π .

1. Using static flow computation, find the partially switched paths P .
 2. From P , construct temporal paths $P(\theta)$.
 3. Construct abstract paths $P \in \mathcal{R}^{\delta,T}$ with delay pattern δ .
 4. Use temporally repeated flow on $P \in \mathcal{R}^{\delta,T}$ by using Kappmeier et al. (2014).
-

Theorem 3.9. *Algorithm 12 computes the solution of an abstract maximum dynamic flow problem with partial switching optimally.*

Proof. Since temporal paths $P(\theta)$ and the paths with delay pattern $P \in \mathcal{R}^{\delta,T}$ are relaxation of abstract paths P with partial switching and satisfy flow conservation as well as capacity constraints, so Algorithm 12 provides feasible solution. Optimality of algorithm is dominated by Step 4. In each path with delay pattern $P \in \mathcal{R}^{\delta,T}$, the static flow ϕ^P is repeatedly send up to $T - \tau_P$ times so that

$$|\Phi^P| = \sum_{P \in \mathcal{R}^{\delta,T}} (T + 1 - \tau_P) \phi^P.$$

Thus, as in Kappmeier et al. (2014), Algorithm 24 provides an optimal solution to abstract maximum dynamic flow with partial switching. \square

Here, Partially switched paths in Step 1 can be obtained in polynomial time as in static case. As in Lemma 3.7, the maximum abstract dynamic flow with complete switching of paths can be computed in $\mu(|\mathcal{E}|, \log(U), \log(T))\mathcal{O}(P)$ time. Again, residual network for partial switched paths in Step 1 can be obtained in $O(|\mathcal{E}|^2)$ time. So an abstract maximum dynamic flow with partially switching can be computed in $O(|\mathcal{E}|^2)\mu(|\mathcal{E}|, \log(U), \log(T))\mathcal{O}(P)$ time.

Theorem 3.10. *The time complexity of Algorithm 24 is polynomial.*

Example 3.6. Consider a dynamic network presented in Figure 3.8 containing six paths $P_1 = s - a - c - t$, $P_2 = s - a - e - d - t$, $P_3 = s - a - e - c - t$, $P_4 = s - b - d - t$, $P_5 = s - b - e - c - t$ and $P_6 = s - b - e - d - t$. As paths P_2 and P_5 crosses at e , for the flow with complete switching flow must be switched towards the abstract paths P_3 and P_6 together

with non-crossing paths P_1 and P_4 . For the abstract flow with partial switching, we have five paths except P_5 . By taking time horizon $T = 10$, maximum flow value in general network obtained by using TRF is 166 units whereas abstract maximum flow with complete switching is 139 units. Flow value is significantly increased to the general network flow value by partial switching which sends 166 units of flow from s to t . Flow pattern in each path is presented in Table 3.3.

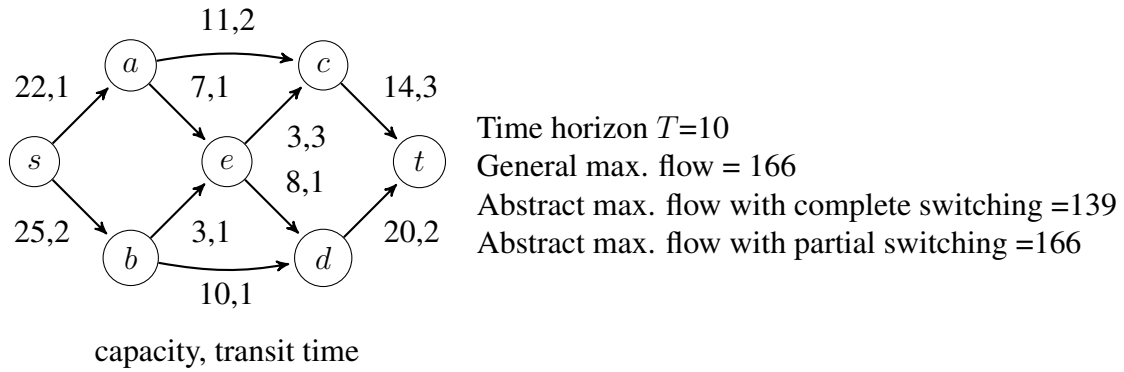


Figure 3.8: Dynamic network with movement capacity and transit time of each element which is used to transship flow to its adjacent element.

Table 3.3: Flow pattern of Figure 3.8 with time horizon $T = 10$.

Path notation	Path	Transit time	General flow	Flow with complete switching	Flow with partial switching
P_1	$s - a - c - t$	6	55	55	55
P_2	$s - a - e - d - t$	5	42	--	30
P_3	$s - a - e - c - t$	8	--	9	6
P_4	$s - b - d - t$	5	60	60	60
P_5	$s - b - e - c - t$	9	4	--	--
P_6	$s - b - e - d - t$	6	5	15	15
	Total		166	139	166

A question arises here is - what is the reason of using the flow pattern with partial switching even if the flow in general network is same? Reason behind is that, in general network, the waiting time of the flow at crossings are not considered in the solution procedure. In real life scenario, when two paths cross at an intersection, turn by turn the flow must wait in one path until the flow passes through another path. Thus possibly half flow can transshipped in paths P_2 and P_5 , and so the flow in general network may decreased.

It is to be remarked that, in Figure 3.8, the crossing effect of paths can be seen for the flow with time horizon $T = 9$ or more. This is because, if we consider time horizon $T = 8$ or less, then $s - t$ flow does not exist in path P_5 so that no crossing effect is seen. The flow value in different time horizon is presented in Table 3.4 hereafter.

Table 3.4: Maximum flow obtained from Figure 3.8 in different time horizon.

Time horizon	General network flow	Flow with complete switching	Flow with partial switching	Crossing effect
$T = 5$	17	17	17	No
$T = 6$	46	46	46	No
$T = 7$	75	75	75	No
$T = 8$	104	104	104	No
$T = 9$	135	112	135	Yes
$T = 10$	166	139	166	Yes

3.2.3 Abstract Quickest Flow with Partial Switching

For an abstract network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$, our concern here is to find the minimum possible time T to transship the given amount of flow $|\Phi|$ from the source element that is to be sent to the sink element via $s - t$ paths $P \in \mathcal{R} \subseteq \mathcal{P}$ by allowing the partial switching property at each crossing element. As in Burkard et al. (1993), the maximum dynamic flow is a non-decreasing function of time T and finding a solution to the quickest flow problem satisfying given amount of flow $|\Phi|$ is equivalent to finding the minimum time T such that $|\Phi(T)| \geq |\Phi|$. By using parametric search, Burkard et al. (1993) presented a polynomial time algorithm for general network topology. Here, we adopt the same technique to introduce the abstract quickest flow problem with partial switching. The text is published in conference paper Khanal et al. (2022).

Solution Procedure. We start the solution procedure by constructing the temporal paths with delay pattern. Flows are sent via temporal paths until the given demand is not fulfilled. For the polynomial time solution, we use binary search method starting with interval $[T_{min}, T_{max}]$ such that $|\Phi(T_{min})| \leq |\Phi(T)| \leq |\Phi(T_{max})|$, which implies that the quickest time $T \in [T_{min}, T_{max}]$. Initially, we start with $T_{min} = \tau_{P^*}$ and $T_{max} = \tau_{P^*} + \left\lceil \frac{|\Phi|}{\phi^{P^*}} \right\rceil$, $\phi^{P^*} > 0$, where $P^* \in \mathcal{R}$ is the shortest path with partial switching and ϕ^{P^*} is the static flow on P^* . In case of the existence of more than one shortest paths with same length, we take a minimum flow shortest path as P^* . In each iteration, the searched interval is halved unless the point of convergence is obtained and the flow values at extreme points of interval are obtained by using polynomial time algorithm of Kappmeier et al. (2014). Applying this technique, the algorithmic framework for the abstract quickest flow with partial switching is presented in Algorithm 13.

Algorithm 13: Abstract quickest flow algorithm with partial switching

Input : Given abstract dynamic network $\Pi = (\mathcal{E}, \mathcal{P}, \tau, T)$ and flow value $|\Phi|$.

Output: Abstract quickest flow with partial switching on Π .

1. Using static flow computation, find the partially switched paths $P \in \mathcal{R}$.
 2. From P , construct temporal paths $P(\theta)$.
 3. Construct abstract paths $P \in \mathcal{R}^\delta$ with delay pattern δ .
 4. Use binary search on $[T_{min}, T_{max}]$ unless $|\Phi(T)|$ converges to $|\Phi|$.
 5. $T =$ quickest time to satisfy demand $|\Phi|$.
-

Theorem 3.11. *Algorithm 13 provides the polynomial time solution to solve the abstract quickest flow problem with partial switching.*

Proof. The time complexity of maximum dynamic flow at each extreme points of the interval can be computed polynomially by using oracle (Kappmeier et al. (2014)). Similarly, due to binary search, time complexity of quickest time on $[T_{min}, T_{max}]$ is also polynomial (Burkard et al. (1993)). So Algorithm 13 solves an abstract quickest flow problem with partial switching in polynomial time, (as similar to Theorem 2.8). \square

Chapter 4

Multi-commodity Network Flow

In day-to-day life, we have realized the shipment of more than one commodities from one place to another. The flow problem regarding the transshipment of more than one different commodities from respective sources (origins) to corresponding sinks (destinations) through a network without violating the capacity constraints on the arcs is known as multi-commodity flow (MCF) problem. Some of the very commonly used examples of multi-commodity flow problems in the literature of mathematical modeling are vehicle routine in transportation, production planning, supply chains for essential goods, message routing in telecommunication, etc., Ahuja et al. (1993); Ali et al, (1980); Assad (1978); Kennington (1978). On the basis of temporal dimension, multi-commodity flow problem can be classified as static multi-commodity flow problem and dynamic multi-commodity flow problem. If we maximize the supply-demand in a fixed time horizon, then the problem becomes a maximum dynamic multi-commodity flow problem. The static multi-commodity flow problem is polynomial time solvable by using the ellipsoid or interior point method, whereas dynamic multi-commodity flow problem is \mathcal{NP} -hard, Hall et al. (2007). By using time expanded network, Kappmeier (2015) provided the solution of maximum dynamic multi-commodity flow problem and multi-source single sink multi-commodity earliest arrival transshipment problem in pseudo-polynomial time complexity.

Some basic characteristics of multi-commodity flow problems that distinguish it from the single commodity flow problems are as follows.

- The multi-commodity flow problems have quite different nature in common arcs that carry more than one commodities, where sum of the flows of different commodities can not exceed its capacity.
- The single commodity models cancel the flows in cycles but the multi-commodity flow may contrary this property for different commodities.
- The single commodity maximum static flow can be computed by very well known max-flow min-cut theorem but this may not be correct for multi-commodity flow problems.

- If capacities on the arc are integer, then single commodity flow problem gives integer solution but this may not always be true for multi-commodity flow.

Notations. Consider a network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, d_i, S, D)$, where S and D are the set of sources and sinks such that $s_i \in S \subset \mathcal{N}$ and $t_i \in D \subset \mathcal{N}$ with respect to the commodities $i \in K = \{1, 2, \dots, h\}$. Here, d_i represents the amount of supply from the source node s_i for each commodity $i \in K$ that is to be sent to the corresponding sink t_i . In case of maximum flow problem, d_i is a variable to be maximized whereas for quickest flow problem, it becomes a constant (given parameter). Rest of the symbols have their usual meaning as defined in Chapter 2. The set of intermediate nodes is denoted by $\mathcal{I} = \mathcal{N} \setminus \{S, D\}$. Similarly, the dynamic multi-commodity network topology with temporal components can be represented as $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, d_i, \tau, S, D, T)$. Hereafter, the flow models for static and dynamic networks are defined as follows.

4.1 Maximum Multi-commodity Flow

Static Multi-commodity Flow Model

For the given network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, d_i, S, D)$, the static multi-commodity flow function ϕ is the sum of non-negative arc flow functions $\phi_a^i : \mathcal{A} \times K \rightarrow \mathbb{R}^+$ for each $i \in K$, satisfying the conditions (4.1a - 4.1c). The linear programming formulation of static multi-commodity flow is as follows.

$$\max \sum_{i \in K} d_i \quad (4.1a)$$

such that,

$$\sum_{a \in \Gamma_u^{out}} \phi_a^i - \sum_{a \in \Gamma_u^{in}} \phi_a^i = \begin{cases} d_i & \text{if } u = s_i \\ -d_i & \text{if } u = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in K \quad (4.1b)$$

$$0 \leq \phi_a = \sum_{i \in K} \phi_a^i \leq \kappa_a \quad \forall a \in \mathcal{A} \quad (4.1c)$$

Objective function in equation (4.1a) is to maximize the total flow out from each source (i.e., supply) which is equal to the total inflow at the sink (i.e., demand). Here, d_i represents the supply/demand of flow (i.e., flow value) so that $d_i = |\phi^i|$. Three conditions in equation (4.1b) represent the supply, demand and conservation of flow at sources, sinks and intermediate nodes, respectively. The constraint in (4.1c) represents the bundle constraint on each arc that is bounded by its capacity. If c represents the per unit cost of static flow ϕ associated with arc a and commodity i with coefficient c_a^i , then

$$c(\phi) = \sum_{i \in K} \sum_{a \in \mathcal{A}} c_a^i \phi_a^i.$$

Dynamic Multi-commodity Flow Model

The multi-commodity flow over time function Φ , defined on a given dynamic network Π with constant transit time τ on each arc a , is the sum of non-negative arc flow functions $\Phi^i : \mathcal{A} \times K \times \mathcal{T} \rightarrow \mathbb{R}^+$, for each $i \in K$, satisfying the constraints (4.2a - 4.2d). The linear programming formulation of dynamic multi-commodity flow is as follows.

$$\max \sum_{i \in K} d_i \quad (4.2a)$$

such that,

$$\sum_{a \in \Gamma_u^{out}} \sum_{\beta=\tau_a}^{\theta} \Phi_a^i(\beta) - \sum_{a \in \Gamma_u^{in}} \sum_{\beta=0}^{\theta} \Phi_a^i(\beta - \tau_a) \leq 0, \quad \forall u \in \mathcal{I}, i \in K, \theta \in \mathcal{T} \quad (4.2b)$$

$$\sum_{a \in \Gamma_u^{out}} \sum_{\theta=0}^T \Phi_a^i(\theta) - \sum_{a \in \Gamma_u^{in}} \sum_{\theta=\tau_a}^T \Phi_a^i(\theta - \tau_a) = \begin{cases} d_i & \text{if } u = s_i \\ -d_i & \text{if } u = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in K \quad (4.2c)$$

$$0 \leq \Phi_a(\theta) = \sum_{i \in K} \Phi_a^i(\theta) \leq \kappa_a, \quad \forall a \in \mathcal{A}, i \in K, \theta \in \mathcal{T} \quad (4.2d)$$

The objective function in equation (4.2a) is to maximize the total flow out from the sources (i.e., supply) within the time horizon T which is equal to the sum of inflow at sinks (i.e., demand). Equation (4.2b) represents the weak flow conservation at intermediate nodes for each time step θ . In any instance of time θ , the bundle constraint in (4.2d) is bounded by arc capacity. Similarly, three steps of equation (4.2c) represent the supply, demand and flow conservation at sources, sinks and intermediate nodes within the time horizon T , respectively. The cost of discrete time dynamic flow Φ associated with arc a and commodity i with cost coefficient c_a^i is

$$c(\Phi) = \sum_{i \in K} \sum_{a \in \mathcal{A}} c_a^i \sum_{\theta=0}^T \Phi_a^i(\theta).$$

If B^i be the budget constraint of commodity i which represents the upper bound of the cost function, then $\sum_{a \in \mathcal{A}} c_a^i \sum_{\theta=0}^T \Phi_a^i(\theta) \leq B^i$, for all $i \in K$.

Solving the multi-commodity flow problem is comparatively more complex than the single-commodity flow problem. The solutions of maximum dynamic multi-commodity flow problem and multi-source single sink multi-commodity earliest arrival transshipment problem can be

found in Kappmeier (2015), where the solutions are obtained by using a time expanded network within pseudo-polynomial time complexity. Similarly, Skutella (2009) presented polynomial time approximation to solve maximum dynamic multi-commodity flow problem using TRF on T-length bound paths.

As sharing of the capacities on bundle arcs is one of the major issues in multi-commodity flow problems, we introduce prioritized multi-commodity flow, multi-commodity flow with proportional and flow-dependent capacity sharing to solve the maximum flow problem. We also incorporate the concept of intermediate storage on solving the maximum flow problem.

4.1.1 Commodity Prioritized Maximum Multi-commodity Flow

Prioritization is the process of deciding the relative importance or urgency of the things or objects. It helps on achieving the goals in an efficient way. As we realize on day to day life, every individual has a list of works to be completed within certain period of time. The optimal use of limited time is only possible by efficient planning of the works with priority. Thus, a vital skill that is used to manage the variety of tasks by deciding the relative importance is the prioritization. Though priority may not be uniform for each individual, it helps to fulfill the objectives, achieve the goals and improve the efficiency of the work. In network flow and optimization, it is highly applicable for large scale disaster management and facility allocation problems. Priority based static multi-commodity flow problem and polynomial time solution strategy can be found in our conference paper Khanal et al. (2020).

Priority Ordering on Evacuation Planning. At the time of post disaster evacuation, each and every evacuees may not have equal priority while shifting them from the danger zone to the safe place. People who are critically injured (or at high risk zone) need very quick treatment (or rescue) to save their lives. Similarly, minor injured, old aged people and pregnant women or with babies, normal evacuees (i.e., comparatively low risk) may be successively lower in priority order in evacuation process. Critically injured people have to be transshipped to the well equipped hospitals as soon as possible, whereas minor injured evacuees can be treated in the health centers. Similarly, old aged people and pregnant women or women with babies have to be sent to the shelters with special care whereas the normal people to some safe shelters. In this evacuation scenario, the collection of evacuees are made according to their priority order (as per the case sensitive) at different collection centers. Our assumption is that the evacuees within the group (collection center) are of homogeneous character and between the groups are of heterogeneous character. Thus, evacuation problem becomes a multi-commodity flow problem with commodity priority where commodities are transshipped from respective sources (collection centers s_i) to corresponding sinks (destinations t_i) without violating the capacity constraint on the arcs.

We define the commodity priority ordering function $\mathbb{P} : K \rightarrow \mathcal{Z}^+$ such that $\mathbb{P}(i) \succ \mathbb{P}(i+1)$, $i =$

$1, \dots, h-1$, where $\mathbb{P}(i)$ represents the priority of i^{th} commodity that are to be transshipped from s_i to t_i . The symbol $\mathbb{P}(i) \succ \mathbb{P}(i+1)$ represents that $\mathbb{P}(i)$ is higher in priority than $\mathbb{P}(i+1)$. At any instance of time θ , if more than one commodities are entering on an arc, then commodity of the first priority $\mathbb{P}(1)$ enters at first and if the flow of first commodity is strictly less than the arc capacity, then commodity of the second priority $\mathbb{P}(2)$ is to be entered and so on with successive priority order.

The priority order of flows in arcs of the network is as follows. If ϕ_a^i and κ_a^i denote the static flow and arc capacity of commodity i on arc a , then

$$\begin{aligned} \sum_{i \in K} \phi_a^i &\leq \kappa_a \\ 0 \leq \phi_a^1 &\leq \kappa_a^1 = \kappa_a \\ 0 \leq \phi_a^2 &\leq \kappa_a^2 = \kappa_a - \phi_a^1 \\ &\vdots \\ 0 \leq \phi_a^h &\leq \kappa_a^h = \kappa_a - \sum_{i=1}^{h-1} \phi_a^i \end{aligned}$$

Similarly, if Φ_a^i and κ_a^i denote the dynamic flow and arc capacity of commodity i on arc a , then at any time step $\theta \in \mathcal{T}$

$$\begin{aligned} \sum_{i \in K} \Phi_a^i(\theta) &\leq \kappa_a \\ 0 \leq \Phi_a^1(\theta) &\leq \kappa_a^1(\theta) = \kappa_a \\ 0 \leq \Phi_a^2(\theta) &\leq \kappa_a^2(\theta) = \kappa_a - \Phi_a^1(\theta) \\ &\vdots \\ 0 \leq \Phi_a^h(\theta) &\leq \kappa_a^h(\theta) = \kappa_a - \sum_{i=1}^{h-1} \Phi_a^i(\theta) \end{aligned}$$

Flow Model. For the given network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, d_i, S, D)$, the prioritized static multi-commodity flow function ϕ is the sum of non-negative arc flow functions $\phi_a^i : \mathcal{A} \times K \rightarrow \mathbb{R}^+$ for each $i \in K$, satisfying the conditions (4.1a - 4.1c) together with the priority ordering constraint in (4.3).

$$0 \leq \phi_a^i \leq \kappa_a^i = \kappa_a - \sum_{r=0}^{i-1} \phi_a^r, \quad \forall i \in K \quad (4.3)$$

Here, constraint in (4.3) represents the allocation of capacity for each commodity with respect to the priority order, where $\phi_a^0 = 0$. The objective is to maximize the amount of flow out from the sources $\sum_{i \in K} d_i$ with respective order of priority.

Similarly, the multi-commodity flow over time function Φ , defined on a given dynamic network Π with constant transit time τ on each arc a , is the sum of non-negative arc flow functions $\Phi^i : \mathcal{A} \times \mathcal{T} \times K \rightarrow \mathbb{R}^+$, for each $i \in K$, satisfying the constraints (4.2a - 4.2d) together with priority ordering constraint (4.4).

$$0 \leq \Phi_a^i(\theta) \leq \kappa_a^i(\theta) = \kappa_a - \sum_{r=0}^{i-1} \Phi_a^r(\theta), \quad \forall i \in K \quad (4.4)$$

Here, allocation of the capacity of bundle arc for each commodity with respect to the priority order is presented in constraint (4.4), where $\Phi_a^0 = 0$ and the goal is to maximize the total flow $\sum_{i \in K} d_i$ out from the sources that reaches to the sinks within the given time horizon T .

Commodity Prioritized Maximum Static Multi-commodity Flow

For a given static multi-commodity network Π with commodity priority order $\mathbb{P}(i) \succ \mathbb{P}(i + 1)$, $i = 1, \dots, h - 1$, our aim is to maximize $\sum_i d_i$ by sending the flow value d_i from s_i to t_i using priority order of commodities and without violating the capacity constraints on the arcs.

To solve this problem, we first prioritize the commodities by $\mathbb{P}(i) \succ \mathbb{P}(i + 1)$, $i = 1, \dots, h - 1$, so that the priority order of sources and sinks are also determined with respect to the commodity. While sending the flow on bundle arcs, priority is given to the commodities as presented in equation (4.3). We use the lexicographic approach to solve the prioritized maximum static flow problem in polynomial time. Here, we present an algorithm to solve the problem.

Algorithm 14: Commodity prioritized maximum static multi-commodity flow algorithm

Input : Given static multi-commodity network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, d_i, S, D)$.

Output: Commodity prioritized maximum static multi-commodity flow on Π .

1. Prioritize the commodities such that $\mathbb{P}(i) \succ \mathbb{P}(i + 1)$, $i = 1, \dots, h - 1$.
 2. Static_flow = 0 (initialization of the static flow).
 3. For $i = 1, \dots, h$,
 - (a) Compute maximum static flow d_i from s_i to t_i .
 - (b) Static_flow = Static_flow + d_i .
 - (c) Update the network with residual capacity $\kappa_a - \phi_a^i$.
 4. Maximum static flow with priority order = Static_flow.
-

Theorem 4.1. *Algorithm 14 provides the lexicographically maximum flow with priority order of commodities.*

Proof. Algorithm starts by fixing the priority order of commodities. Initially, we set the static flow as zero. Maximum flow calculation begins inside the ‘for’ loop with $i = 1$. Since $s_1 - t_1$ network contains single commodity flow from s_1 to t_1 with given arc capacities, it is single source and single sink static flow. So maximum flow algorithm of Ford & Fulkerson (1962)

provides the maximum number of flow units from s_1 to t_1 . Thus, the flow value d_1 is maximal flow and stored it on static flow value as $\text{Static_flow} = d_1$.

Before starting the solution for $s_2 - t_2$ flow, network is updated by reducing the capacity of arcs used by the flow of commodity-1. Thus, the updated network is obtained with residual capacity $\kappa_a - \phi_a^1$. On the updated network, maximum static flow from s_2 to t_2 is calculated using maximum flow algorithm to obtain d_2 and is optimal $s_2 - t_2$ flow. This flow is accumulated to static flow value by $\text{Static_flow} = d_1 + d_2$ and the network is again updated by reducing the arc capacity used by commodity-2. This process is continued by ‘for’ loop until the optimal flow for last prioritized commodity h is obtained. As each d_i is lexicographically optimal, total static flow $\sum_i d_i$ is optimal. Thus, Algorithm 14 provides the lexicographically maximum flow (i.e., maximal flow) with priority order of commodities \square

Theorem 4.2. *Algorithm 14 solves the commodity prioritized maximum static multi-commodity flow problem in polynomial time.*

Proof. At first, we prove the feasibility of the theorem. Step 1 is the prioritization of commodities which can be obtained in constant time, so is feasible. The residual capacity in Step 3(c) is obtained by using priority order as presented in (4.3) in each bundle arcs within $O(m)$ times. As the maximum static flow is polynomial time solvable, Step 3 is also feasible. Next, the optimality of the solution is dominated by the optimality of Step 3(a) In which maximum static flow is computed for h times within the loop and is polynomially solvable ($\because h \leq \frac{n}{2}$). Thus, Theorem 4.1 solves the problem in polynomial time complexity. \square

Example 4.1. Consider a multi-commodity network with three commodities that are to be transshipped from s_i to t_i for $i = 1, 2, 3$ as shown in Figure 4.1. While sending the flow, first priority is given to commodity-1 through two paths: $s_1 - t_1$ with flow value 3 units and $s_1 - v - w - t_1$ with bottleneck capacity 4. Path $s_1 - v - w - t_1$ uses arc (v, w) which is a bundle arc where flow transmission can take place for other commodities as well with priority order. The maximum static flow of $d_1 = 7$ units is obtained for commodity-1. The remaining capacity in arc (v, w) for commodity-2 is $11 - 4 = 7$, so commodity-2 is transshipped through path $s_2 - v - w - t_2$ with flow value $d_2 = 5$ units because of its bottleneck capacity. Similarly, third priority is given to commodity-3 in which we can send 2 units of flow on the path $s_3 - v - w - t_3$ and 4 units of flow is send through path $s_3 - t_3$. Total amount of prioritized flow for commodity-1, commodity-2 and commodity-3 are $d_1 = 7$ units, $d_2 = 5$ units and $d_3 = 6$ units, respectively, and total flow of transmission is 18 units. \square

Commodity Prioritized Maximum Dynamic Multi-commodity Flow

In this subsection, we describe the solution strategy of commodity prioritized maximum dynamic flow in two ways: an approximate solution by TRF with polynomial time complexity and

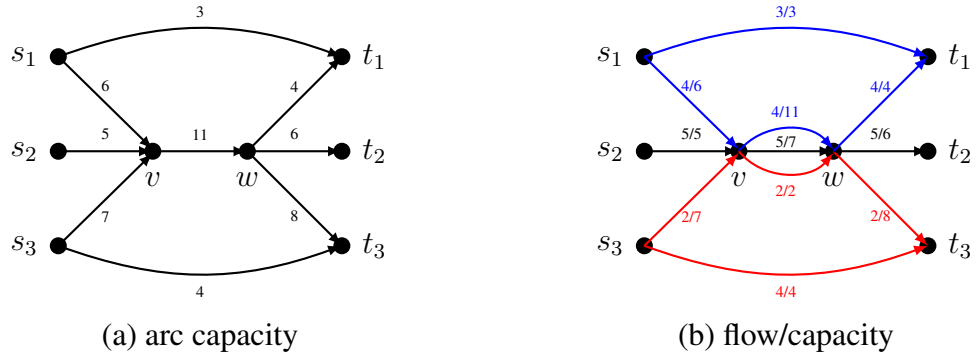


Figure 4.1: (a) Multi-commodity network, (b) Solution with priority order.

another, optimal solution in time expanded network with pseudo-polynomial time complexity.

(i) An Approximate Solution by Temporally Repeated Flow (TRF)

As presented and described in equation (2.9) of Chapter 2, the TRF is a technique to find the maximum dynamic flow for the single commodity flow problem with single source single sink in which constant rate of static flow is repeated along the decomposed paths within the time frame. Here, our concern is to use this technique to find the commodity prioritized maximum dynamic multi-commodity flow problem. For this, we fix the priority order of commodity as in static case. As commodity-1 is in first priority, static $s_1 - t_1$ flow is obtained using static maximum flow ϕ^1 and then the flow is decomposed on paths. On the decomposed paths, the TRF $\Phi^1 = d_1$ is calculated using equation (2.9). The network is then updated by reducing the capacity of arcs used by static flow ϕ_a^1 . As the flow model is dynamic, the static flow ϕ_a^1 can also be written in dynamic flow as $\Phi_a^1(\theta)$ because dynamic flow is the accumulation of static flow in each time step θ . Now, on the updated network, process is repeated for commodity-2 from s_2 to t_2 . The process will be terminated after finding the maximum dynamic flow for commodity- h . In each iteration, the flow of previous iteration fixed and the flow for next iteration is obtained without changing previous solutions, so the solution procedure is lexicographic. The algorithmic framework is as follows.

Algorithm 15: Commodity prioritized maximum dynamic multi-commodity flow algorithm

Input : Given dynamic multi-commodity network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, d_i, \tau, S, D, T)$.

Output: Commodity prioritized maximum dynamic multi-commodity flow on Π .

1. Prioritize the commodities such that $\mathbb{P}(i) \succ \mathbb{P}(i + 1)$, $i = 1, \dots, h - 1$.
 2. Dynamic_flow = 0 (initialization of the dynamic flow).
 3. For $i = 1, \dots, h$,
 - (a) Compute the static $s_i - t_i$ flow ϕ^i and decompose the flow on paths ϕ_p^i .
 - (b) Compute temporally repeated flow $\Phi^i = d_i$ from s_i to t_i .
 - (c) Dynamic_flow = Dynamic_flow + d_i .
 - (d) Update the network with residual capacity $\kappa_a - \phi_a^i$.
 4. Maximum dynamic flow with priority order = Dynamic_flow.
-

Theorem 4.3. *Algorithm 15 computes an approximate solution for the commodity prioritized maximum dynamic flow problem in polynomial time.*

Proof. Due to the commodity priority, the multi-commodity flow problem is reduced to the single commodity one. We first solve maximum dynamic flow on $s_1 - t_1$ sub-network as a single commodity flow problem by using TRF. The arc capacity is reduced in updated network and is fixed for commodity-2. Now, the TRF is possible on $s_2 - t_2$ sub-network as a single commodity flow problem because the capacity is fixed for commodity-2 throughout the time horizon. The process is continued for each commodity with respective priority order. As computational time of TRF in Step 3(b) is constant and other steps can be computed in polynomial time, as in Algorithm 14, so the time complexity of Algorithm 15 is polynomial. \square

Solution provided by Algorithm 15 is an approximate optimal solution. The reason behind is that the capacity on bundle (common) arc is reduced by the higher prioritized commodity from the beginning (i.e., $\theta = 0$) even if the higher prioritized commodity may enter an arc later than the lower prioritized commodity. It may effect the optimal solution of the network.

Example 4.2. Consider a network of Example 4.1 with capacity and transit time on each arc as shown in Figure 4.2. Let $T = 10$ be given time horizon. As in Example 4.1, static flow on two paths $s_1 - t_1$ and $s_1 - v - w - t_1$ are 3 and 4 units, respectively. Using temporally repeated, maximum dynamic flow for commodity-1 is $d_1 = 59$ units.

After reducing the used capacity, the static flow for commodity-2 in path $s_2 - v - w - t_2$ is 5 units so that the dynamic flow for commodity-2 is $d_2 = 35$ units. Again after reduction of used in the capacity on arcs, as in Example 4.1, two paths for commodity-3 with static flow 2 and 4 units exist in which dynamic flow from s_3 to t_3 is $d_3 = 46$ units. Thus, total amount of dynamic flow in time horizon $T = 10$ is 140 units. The solution obtained here is an approximate solution whose optimal solution is obtained by time expanded network in Example 4.3

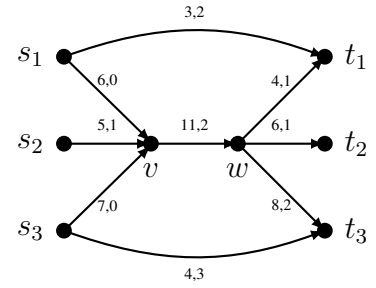


Figure 4.2: Dynamic network with κ_a and τ_a .

(ii) An Optimal Solution Using Time Expanded Network

As described in Section 2.1 of Chapter 2, the time expanded network of given network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, d_i, \tau, S, D, T)$, denoted by $\Pi^T = (\mathcal{N}^T, \mathcal{A}^T, \kappa, \tau, d_i, S(\theta), D(\theta), T)$, is an expanded network over the time T in which the original network is splitted over the time with duplication of each node $u \in \mathcal{N}$ at every time period by $u(0), \dots, u(T)$. The set of arcs in time expanded network contains two types of arcs, movement arcs and holdover arcs, denoted by $\mathcal{A}^T = \mathcal{A}_M \cup \mathcal{A}_H$. As our flow model is commodity prioritized, capacity of each movement arc is shared with respect to the priority of commodity obtained in equation (4.4), especially in the

bundle arcs. Thus, the movement arcs are defined as $\mathcal{A}_M = \{a^i = (u(\theta), v(\theta + \tau_a))^i : a = (u, v) \in \mathcal{A}, \theta = 0, 1, \dots, T - \tau_a, i \in K\}$ and holdover arcs as $\mathcal{A}_H = \{(u(\theta), u(\theta + 1)) : u \in \mathcal{N}, \theta = 0, 1, \dots, T - 1\}$. Movement arcs carry the flow from tail node to the head node beyond the limit of their predefined capacities and transit times whereas holdover arcs hold the flow for unit time step with infinite (sufficiently large) capacity.

Though flow obtained in time expanded network is optimal, major drawback of this method is that the solution provided can not be computed in polynomial time complexity. Its solution depends on time horizon T , so is pseudo-polynomial.

Example 4.3. Consider the network presented in Figure 4.2 whose time expanded network with time horizon $T = 10$ is constructed on Figure 4.3. All $s_i - t_i$ paths obtained by movement arcs are presented in solid arrows and holdover arcs by dashed arrows. At $\theta = 0$, flow of commodity-1 and commodity-3 enter the arc (v, w) , so commodity-1 uses capacity of 4 units and remaining capacity of 7 units is used by commodity-3. But after $\theta = 1$ onward, all three commodities use the arc (v, w) and the capacity is shared as in Example 4.2. Within the time horizon $T = 10$, amount of flow shipped for commodity-1, commodity-2 and commodity-3 are 59, 35 and 51 units, respectively. Thus total amount of flow transshipped is 145 units. \square

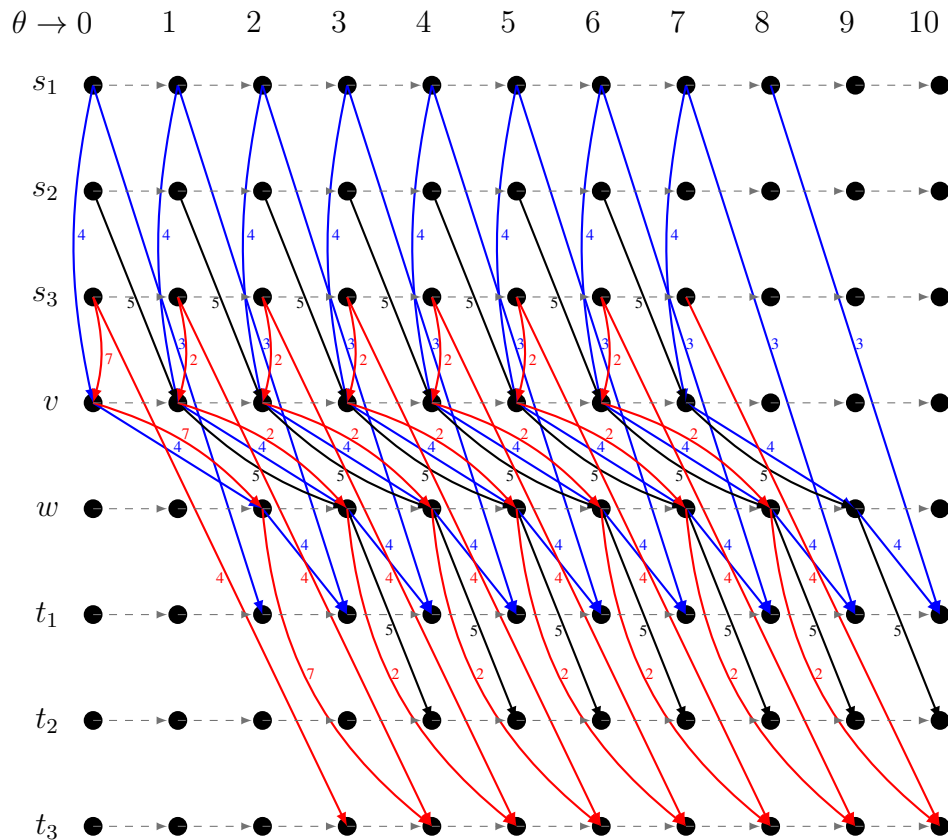


Figure 4.3: The time expanded network of Figure 4.2. Blue, black and red paths are for commodity-1, commodity-2 and commodity-3, respectively.

Commodity-wise Separation of Mix Commodity Terminals

When we go to a supermarkets or a business complex, we can see more than one commodities inside the same roof. The question arises here is, how the multi-commodity flow model applied in such realistic problems with mixed commodity at same destination? To solve the problem, we investigate the partitioning method which separates the mix commodity to single one.

Let $D_r, r = 1, \dots, l$ be destinations which contains commodities $i \in K$. Partition the destination D_r into virtual destinations D_r^i with respect to the commodity $i = 1, \dots, h$ and connect them with virtual arcs $(D_r - D_r^i)$ having infinite capacity and zero cost/transit time. Again, create the super sink t_i and connected with D_r^i by using virtual arc $(D_r^i - t_i), r = 1, \dots, l$ having infinite capacity and zero cost/transit time. The network so formed is an extended network. By this technique, all the destinations of mixed commodity are separated commodity-wise and can be applicable for multi-commodity flow model. The static and dynamic flow problems can be solved in this extended network using priority based flow models, as explained in previous sections. Similar partitions can be used for the sources, if necessary.

Example 4.4. Consider a two-commodity static network having two destinations D_1, D_2 containing both of commodities, commodity-1 and commodity-2, as presented in Figure 4.4, where

s_1 and s_2 are the sources for commodity-1 and commodity-2, respectively. The numbers assigned on each arc are capacity and cost. We partition the destination D_1 into two virtual destinations D_1^1 and D_1^2 for commodity-1 and commodity-2, respectively (as presented in Figure 4.5(a)). Similarly, we partition D_2 into D_2^1 and D_2^2 for commodity-1 and commodity-2, respectively. Join D_1 to D_1^1 and D_1^2 and likewise D_2 to D_2^1 and D_2^2 with $u = \infty, c = 0$. Similarly, join D_1^1 and D_2^1 to t_1 , likewise D_1^2 and D_2^2 to t_2 with $u = \infty, c = 0$, where t_1, t_2 are super sinks for commodity-1 and commodity-2, respectively.

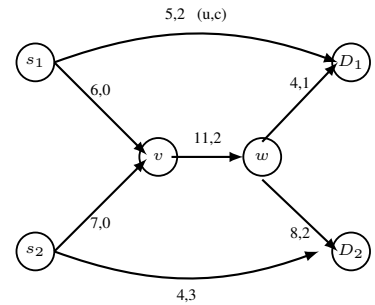


Figure 4.4: Network with mix commodity at D_1 and D_2

Figure 4.5(b) represents the solution of prioritized multi-commodity flow problem from s_i to t_i for $i = 1, 2$, in which 11 and 9 units of flow is transhipped to t_1 and t_2 , respectively. □

4.1.2 Maximum MCF with Proportional and Flow-dependent Capacity Sharing

On solving multi-commodity flow problems, flow of different commodities departing from their sources arrive at the common intermediate node and share the capacity through the common arc. The sharing of the capacity in the common arc (bundle arc) is one of the major issues in the multi-commodity flow problems. If the sharing of capacity on the bundle arc is set in proportion to the bottleneck path capacity from their respective sources to the tail node of the bundle arc,

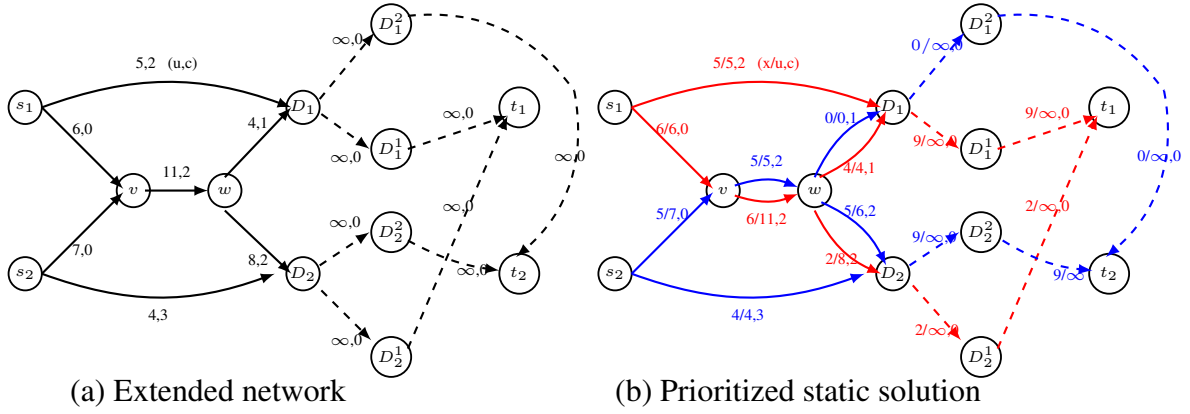


Figure 4.5: Partitioning of destination with mix commodity in Figure 4.4 into commodity-wise separate destination in (a) and its prioritized static solution in (b).

then it is known as proportional capacity sharing. As the shared capacity of the bundle arc for each commodity is fixed, the multi-commodity flow problem is reduced to independent single commodity flow problems. Similarly, flow-dependent capacity sharing refers to the distribution of bundle arc capacity based on the inflow rate of the flow of each commodity. The shared capacity of the bundle arc in this manner might not always be the same because the flow on arc might change over time. However, our assumption is that the nature of flows inside the same commodity group are homogeneous and between the commodity groups are heterogeneous but uniform in the occupancy rate of arc capacity. The text of this subsection is taken from our proceeding paper Khanal et al. (2022a).

(i) Proportional Capacity Sharing

Due to transshipment of more than one commodity on the bundle arcs and the unique source-sink pair for each commodity, multi-commodity flow problem differs from single commodity flow problem. To share the capacity of bundle arc, we present a proportional capacity sharing technique depending on the bottleneck (minimum cut) capacity of paths $P_{[s_i, u]}$ from their respective sources s_i to the tail node u of bundle arc $a = (u, v)$, for each $i \in K$, as follows.

$$\kappa_a^i = \frac{\kappa_e^i}{\sum_{e \in P_{[s_i, u]}; i \in K} \kappa_e^i} \kappa_a \quad (4.5)$$

where, κ_a is the capacity of a bundle arc a to be shared for each commodity $i \in K$, $P_{[s_i, u]}$ is the path from s_i to the tail node u of bundle arc a and e is an arc in $P_{[s_i, u]}$ with bottleneck capacity κ_e which is also the minimum $s_i - u$ cut arc, i.e., $\kappa_e^i = \min\{\kappa_e : e \in P_{[s_i, u]}\}$. The portion of the capacity of arc a allocated for the commodity i is κ_a^i . Clearly, $\sum_{i \in K} \kappa_a^i = \kappa_a$.

The shared capacity for each commodity obtained from equation (4.5) may be in fraction, i.e., $\kappa_a^i = \text{int}(\kappa_a^i) + \text{fra}(\kappa_a^i)$, the sum of integral part $\text{int}(\kappa_a^i)$ and fractional part $\text{fra}(\kappa_a^i)$. The fractional

capacities can be converted into integral capacities as follows.

- First arrange the fractional arc capacities with descending order of fractional part and find their sum $\sum \text{fra}(\kappa_a^i)$. If $\sum \text{fra}(\kappa_a^i) = r$, then first r fractional capacities with greatest fractional part are rounded up using ceiling function $\lceil \cdot \rceil$ and remaining capacities are rounded below by floor function $\lfloor \cdot \rfloor$.
- For the same fractional part in more than one commodities, priority is given to commodity with greatest integral part among them.
- In case of equal integral parts, set the priority with higher demand among them if demand is pre-defined in problem like quickest flow. Otherwise, either can be rounded up.

The static flow model is as presented in equations (4.1a–4.1c) together with one additional constraint of proportional capacity sharing equation (4.5). Similarly, for dynamic flow with proportional capacity sharing, the flow model is as presented in equations (4.2a–4.2d) together with proportional capacity sharing equation (4.5).

Solution Procedure for Static MCF. The solution procedure starts with reduction of the multi-commodity flow problem into h independent single commodity flow problems by sharing the capacity of the bundle arc using equation 4.5. The maximum static flow $\phi^i = d_i$ is obtained for each individual commodity whose sum is the total maximum static flow value $|\phi| = \sum_i d_i$. The algorithmic framework is presented in Algorithm 16.

Algorithm 16: Maximum static MCF algorithm with proportional capacity sharing

Input : Given static multi-commodity flow network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, d_i, S, D)$.

Output: Maximum static MCF on Π with proportional capacity sharing.

1. Construct h independent sub-problems by proportional capacity sharing (4.5) on bundle arcs for all $i \in K$.
 2. Compute the solution $\phi^i = d_i$ to the static maximum flow problem for all i .
 3. Maximum flow $|\phi| = \sum_{i \in K} \phi^i$.
-

Solution Procedure for Dynamic MCF. To solve the maximum dynamic flow with proportional capacity sharing, multi-commodity flow problem is reduced to h independent single commodity flow problems. The static flow is obtained for each commodity and the TRF is used on the commodity-wise decomposed paths \mathcal{P}^i for the dynamic flow. The solution strategy is presented in Algorithm 17.

Theorem 4.4. *Algorithms 16 and 17 provide the approximate solutions to the maximum static and dynamic MCF problems, respectively, with proportional capacity sharing in polynomial time.*

Algorithm 17: Maximum dynamic MCF algorithm with proportional capacity sharing

Input : Given dynamic multi-commodity flow network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$.

Output: Maximum dynamic MCF on Π with proportional capacity sharing.

1. Construct h independent sub-problems by proportional capacity sharing (4.5) on bundle arcs for all $i \in K$.
 2. Compute the maximum static flow ϕ^i for all i using Algorithm 16.
 3. Decompose flow ϕ^i into path flows ϕ_P^i with $P \in \mathcal{P}^i$.
 4. Determine maximum dynamic flow for each $i \in K$ using TRF such that
$$d_i = \Phi^i = \sum_{P \in \mathcal{P}^i} (T + 1 - \tau_P) \phi_P^i.$$
 5. $|\Phi| = \sum_{i \in K} \Phi^i$.
-

(ii) Flow-dependent Capacity Sharing

In proportional capacity sharing technique, the shared capacity of each commodity remains fixed at each time step θ so that temporal repetition of flow is possible. But, the major weakness of this technique is that the capacity of bundle arc allocated for one commodity can not be used by other commodities in absence of that commodity. To manage this issue, we introduce flow-dependent capacity sharing where the share of capacity for each commodity depends on the inflow rate of the flow in predecessor arcs. At any instance of time θ , if a bundle arc $a = (u, v)$ with capacity κ_a holds more than one commodities $i \in K$, then the flow-dependent capacity sharing of κ_a for each commodity $i \in K$ is,

$$\kappa_a^i(\theta) = \frac{\Phi_e^i(\theta - \tau_e)}{\sum_{e \in \alpha(a): i \in K} \Phi_e^i(\theta - \tau_e)} \kappa_a \quad (4.6)$$

where, $\alpha(a)$ is the set of predecessor arcs of bundle arc a so that $e \in \alpha(a) \implies \text{head}(e) = \text{tail}(a)$. Clearly, $\sum_{i \in K} \kappa_a^i(\theta) = \kappa_a$. If the shared capacities are in fraction, we can convert them into integer values as described in proportional capacity sharing.

For a given multi-commodity network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$, our concern here is to transship the maximum amount of flow from s_i to t_i within given time horizon T , where shared capacity for each $i \in K$ on the bundle arc is depending on the inflow of the bundle arc. Since the flow of the commodity on the bundle arc may not be fixed for each time step θ , temporal repetition is not possible. So, we solve the problem by using a 3-dimensional time expanded layer graph as follows.

Time expanded Layer Graph. The multi-commodity time expanded layer graph is a three dimensional graph that contains the copy of nodes from underlying static network for every discrete time steps and for each commodity. It is applicable to solve the variety of flow over time problems by applying the algorithms and techniques developed for the static network flows. For a given network Π with integral transit time on arcs and time horizon T , the T -time ex-

panded layer graph Π^T is obtained by creating $T + 1$ copies of node set N which are labeled as $\mathcal{N}(0), \mathcal{N}(1), \dots, \mathcal{N}(T)$ together with θ^{th} copy of node u labeled as $u(\theta)$, $\theta \in \mathcal{T}$ and the commodities $i \in K$. For every arc $a = (u, v) \in \mathcal{A}$ and $\theta \in \{0, 1, \dots, T - \tau_a\}$, there is an arc $a^i(\theta)$ from $u^i(\theta)$ to $v^i(\theta + \tau_a)$ with the same capacity of arc a for single commodity arc and the sharing capacity for bundle arc a . Similarly, the arc from $u^i(\theta)$ to $u^i(\theta + 1)$ represents the holdover arc with infinite capacity that is used to hold the flow for unit time interval $[\theta, \theta + 1)$ for all $\theta \in \{0, 1, \dots, T - 1\}$. As the size of the graph in time expanded network depends on time horizon T , its time complexity is pseudo-polynomial.

Theorem 4.5. *An approximate solution to the maximum dynamic MCF problem with flow-dependent capacity sharing can be obtained in pseudo-polynomial time.*

Example 4.5. Let us consider a two-commodity network presented in Figure 4.6(a) in which commodity-1 is transshipped from s_1 to t_1 and commodity-2 from s_2 to t_2 . The 3-dimensional layer graph Π^T with time horizon $T = 6$ taking the coordinate axes as \mathcal{N} , T and K with usual meanings is presented in Figure 4.6(b). Each commodity $i \in K$ preforms the layers of graphs in vertical line. As commodity-1 can not reach to bundle arc (u, v) at time $\theta = 0$ and $\theta = 1$, commodity-2 uses full capacity of the bundle arc. But then after, capacity is shared among the commodities for $\theta = 2$ and $\theta = 3$ (presented as red arcs). Again, at time $\theta = 4$, the flow of commodity-1 uses full capacity of bundle arc due to the absence of commodity-2. \square

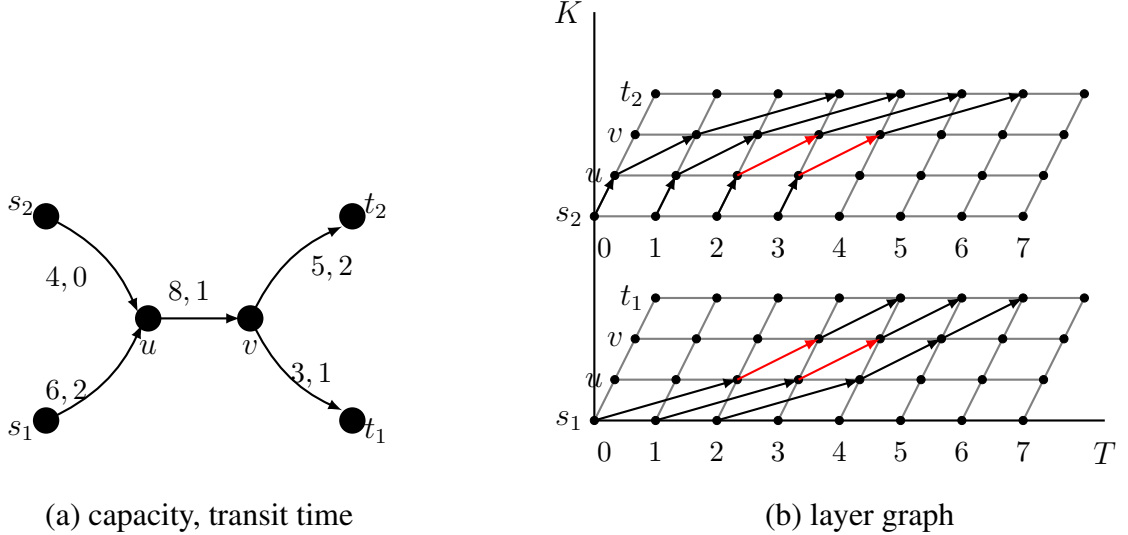


Figure 4.6: (b) represents the time expanded layer graph Π^T of given network (a).

4.1.3 Maximum Multi-commodity Flow with Intermediate Storage

In this subsection, we deal the multi-commodity flow problems with intermediate storage where the storage of the flow at intermediate nodes with some priority order is allowed. We consider the network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, \nu, d_i, S, D)$ or $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \nu, \tau, d_i, S, D, T)$ according as the problem is static or dynamic. Here, an additional component ν represents the storage

capacity of node and for the existence of a feasible solution, the storage capacity of intermediate nodes must be at least the sum of incoming arc capacities. The static and dynamic multi-commodity flow models with intermediate storage and their solution strategies, presented in this subsection, are taken from our paper Khanal et al. (2021).

Maximum Static Multi-Commodity Flow with Intermediate Storage

Flow Model. The mathematical model for the static multi-commodity flow with intermediate storage is as follows.

$$\max \sum_{i \in K} d_i \quad (4.7a)$$

such that,

$$\sum_{a \in \Gamma_{s_i}^{out}} \phi_a^i = \sum_{a \in \Gamma_{t_i}^{in}} \phi_a^i + \sum_{u \in \mathcal{I}} \hat{\phi}_u^i = d_i, \quad \forall i \in K \quad (4.7b)$$

$$\sum_{a \in \Gamma_u^{in}} \phi_a^i - \sum_{a \in \Gamma_u^{out}} \phi_a^i = \hat{\phi}_u^i, \quad \forall u \in \mathcal{I}, i \in K \quad (4.7c)$$

$$0 \leq \phi_a = \sum_{i \in K} \phi_a^i \leq \kappa_a, \quad \forall a \in \mathcal{A} \quad (4.7d)$$

$$0 \leq \hat{\phi}_u = \sum_{i \in K} \hat{\phi}_u^i \leq \nu_u, \quad \forall u \in \mathcal{I} \quad (4.7e)$$

Equation (4.7a) is an objective function which maximizes the total flow $\sum_{i \in K} d_i$. The total flow out from each source for all $i \in K$ is equal to the sum of inflow at the sink and excess flow at intermediate nodes, which is presented in (4.7b). The excess flow of commodity i at each intermediate node is represented in equation (4.7c). The bundle constraint on each arc bounded by its capacity is presented in equation (4.7d) whereas the excess flow at each intermediate node bounded by the storage capacity is presented in equation (4.7e). The storage capacity of intermediate node $u \in \mathcal{I}$ must be at least the sum of incoming arc capacities to u , i.e., $\nu_u \geq \sum_{a \in \Gamma_u^{in}} \kappa_a$.

Solution Procedure. As the solution strategy, the multi-commodity flow problem is first reduced to h independent single commodity flow problems by reallocating the capacity of bundle arcs using the resource directive decomposition method (Ahuja et al. (1993)). It reallocates the capacity of bundle arc for each commodity in such a way that the flow value is optimal. Other approaches of solving multi-commodity flow problems are Lagrangian relaxation, Dantzig-Wolfe decomposition and partitioning method. The maximum flow at each sink $t_i, \forall i \in K$, is obtained by using maximum flow algorithm and the excess flow is stored at intermediate nodes $u \in \mathcal{I}$ with some priority order of nodes. The priority order of nodes can be set as per the justification of the problem instances. We consider the network with single sink for each

commodity $i \in K$ and if there are multiple sinks for the same commodity, then we create a virtual sink to convert the single sink of the commodity and connect it by virtual arcs of infinite capacity. Similar procedure is used for multiple sources of a commodity. Here, we adopt the process of declaration of priority order of nodes as in Section 2.3. We must note that, due to multiple commodities in the bundle arcs, there may be different priority ordering of a node with respect to the commodity. So, the node u with respect to commodity i is denoted by u_i .

The dummy node u_i^* of each prioritized node $u \in \mathcal{I}$ is created commodity-wise (since the node $u \in I$ lying in the bundle arc contains the flow of more than one commodity, so dummy nodes are represented commodity-wise, i.e., u_i^*) with cost $c_{[u, u_i^*]} = 0$ and capacity $\kappa_{[u, u_i^*]} = \nu_u = \nu_{u_i^*}$. Here, the notations $\kappa_{[u, u_i^*]}$ and $c_{[u, u_i^*]}$ represent the arc capacity and cost of dummy arc (u, u_i^*) , respectively. Every dummy node u_i^* with respect to commodity i has the same priority order as u_i has. The modified network $\Pi_i^* = (\mathcal{N}_i^*, \mathcal{A}_i^*, \kappa, c, \nu, d_i, s_i, D_i^*)$ with single source s_i and multiple sink $D_i^* = \{t_i\} \cup \{u_i^*\}$ is formed associated with each commodity i , where D_i^* is the collection of dummy nodes $\{u_i^*\}$ together with the sink t_i . For an instance, if $t_i \succ u_{i,1} \succ \dots \succ u_{i,r}$ be priority order of nodes with respect to commodity i , then $D_i^* = \{t_i = u_{i,0}^*, u_{i,1}^*, \dots, u_{i,r}^*\}$. Hereafter, we present Algorithm 18 to solve the maximum static MCF problem with intermediate storage in single source multi-sink network $\Pi_i^*, \forall i \in K$ as follows.

Algorithm 18: Maximum static MCF algorithm with intermediate storage

Input : Given static network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, c, K, \nu, d_i, S, D)$.

Output: Maximum static multi-commodity flow with intermediate storage in Π .

1. Reconfigure the multi-commodity flow problem into h independent single commodity flow problems by reallocating the capacity of bundle arcs using the resource directive decomposition.
 2. For each $u \in I$, compute commodity-wise shortest distance $d_{[s_i, u]}$ by using Dijkstra's algorithm.
 3. Fix the commodity-wise priority order of nodes $t_i \succ u_{i,1} \succ \dots \succ u_{i,r}$, with first priority to the sink t_i and priority for intermediate nodes as $d_{[s_i, u_{i,k}]} > d_{[s_i, u_{i,k+1}]} \implies u_{i,k} \succ u_{i,k+1}$, for $k = 1, \dots, r-1$.
 4. For each $i \in K$, construct the modified network $\Pi_i^* = (\mathcal{N}_i^*, \mathcal{A}_i^*, \kappa, c, \nu, d_i, s_i, D_i^*)$ with single source s_i and multiple sinks with dummy nodes $D_i^* = \{t_i = u_{i,0}^*, u_{i,1}^*, \dots, u_{i,r}^*\}$.
 5. For $i = 1, \dots, h$:
Compute the lexicographic maximum static flow in Π_i^* with priority order in Step (3).
 6. Transform the solution to the original network Π by removing the dummy nodes and the dummy arcs.
-

Theorem 4.6. *Algorithm 18 solves the maximum static MCF problem with intermediate storage optimally.*

Proof. The proof of the theorem starts with the feasibility of the algorithm. In Step 1, the multi-commodity flow problem is decomposed to single commodity flow problems using resource directive decomposition method and in Steps 2, Dijkstra's algorithm is used to calculate the

shortest distances, so both steps are feasible. Steps 3, 4 and 6 are prioritization of nodes, modification of network and transformation of solution, which does not violate the capacity constraints on the arcs and can be solved in linear time, and so are feasible. Similarly, multi-commodity flow problem is decomposed into h independent single commodity sub-problems and flow with intermediate storage in single source single sink network is feasible (see Pyakurel & Dempe (2020)), so Step 5 provides feasible solution for each commodity $i \in K$.

Next, we prove the optimality of the algorithm which is assured by the optimality of Step 5. Since the lexicographic maximum static flow in prioritized sink D_i^* is obtained optimally after decomposition as the single commodity flow problem. So, the sum of optimal single commodity flows $\sum_{i \in K} d_i$ is optimal multi-commodity flow with intermediate storage. \square

Theorem 4.7. *Maximum static multi-commodity flow problem with intermediate storage can be solved in polynomial time complexity by using Algorithm 18.*

Proof. The time complexity of calculating the shortest distance using Dijkstra's algorithm is $O(n^2)$. The prioritization of nodes and creating dummy nodes can be obtained in linear time. Similarly, decomposition of the multi-commodity flow problem to the single one can be computed in polynomial time. Step 3 can be solved in polynomial time complexity of $O(\delta nm^2)$ for $0 < \delta < n$ by shortest augmenting path algorithm (Pyakurel & Dempe (2020)). Therefore, Algorithm 18 solves the maximum static multi-commodity flow problem with intermediate storage in polynomial time with complexity $O(n^2 + h\delta nm^2)$, where $|K| = h \leq n/2$. \square

Example 4.6. Consider a two-commodity network presented in Figure 4.7(a) where numbers along the arcs are capacity and cost, and numbers aside of the nodes are node capacities. We set the priority order of intermediate nodes with farther-in-distance-higher-in-priority order, where the distance of each intermediate node is obtained by using Dijkstra's algorithm considering the cost as distance. The priority order of nodes for commodity-1 and commodity-2 are $t_1 \succ v \succ u$ and $t_2 \succ u \succ v$, respectively, whose set of prioritized dummy nodes are $D_1^* = \{t_1, v_1^*, u_1^*\}$ and $D_2^* = \{t_2, u_2^*, v_2^*\}$ (see also in Figure 4.7(b)).

While decomposing the flow on the bundle arc (u, v) , flow of 3 and 2 units are assigned for commodity-1 and commodity-2, respectively. By using Algorithm 18, maximum amount of flow leaving the source s_1 is 8 units out of which 2 units of flow through $s_1 - t_1$ path and 2 units along $s_1 - u - v - t_1$ is reached at sink t_1 . As node v is more prioritized than u , flow of 1 and 3 units are reached to v and u , respectively. Similarly, there are three paths $s_2 - t_2$, $s_2 - v - t_2$ and $s_2 - u - v - t_2$ for commodity-2 through which flow of 3, 4 and 1 units are reached at the sink t_2 , respectively. As node u is in higher prioritized than v , the intermediate nodes u holds 2 units of flow along $s_2 - u - v - t_2$ and no flow is hold at node v . At last, the solution is transformed to the original network by removing the dummy nodes and dummy arcs, and flow

of dummy nodes are back to their respective original nodes. The amount of flow stored at sinks and the intermediate nodes are $\phi_{t_1} = 4$, $\phi_{t_2} = 8$, $\phi_u = 5$ and $\phi_v = 1$.

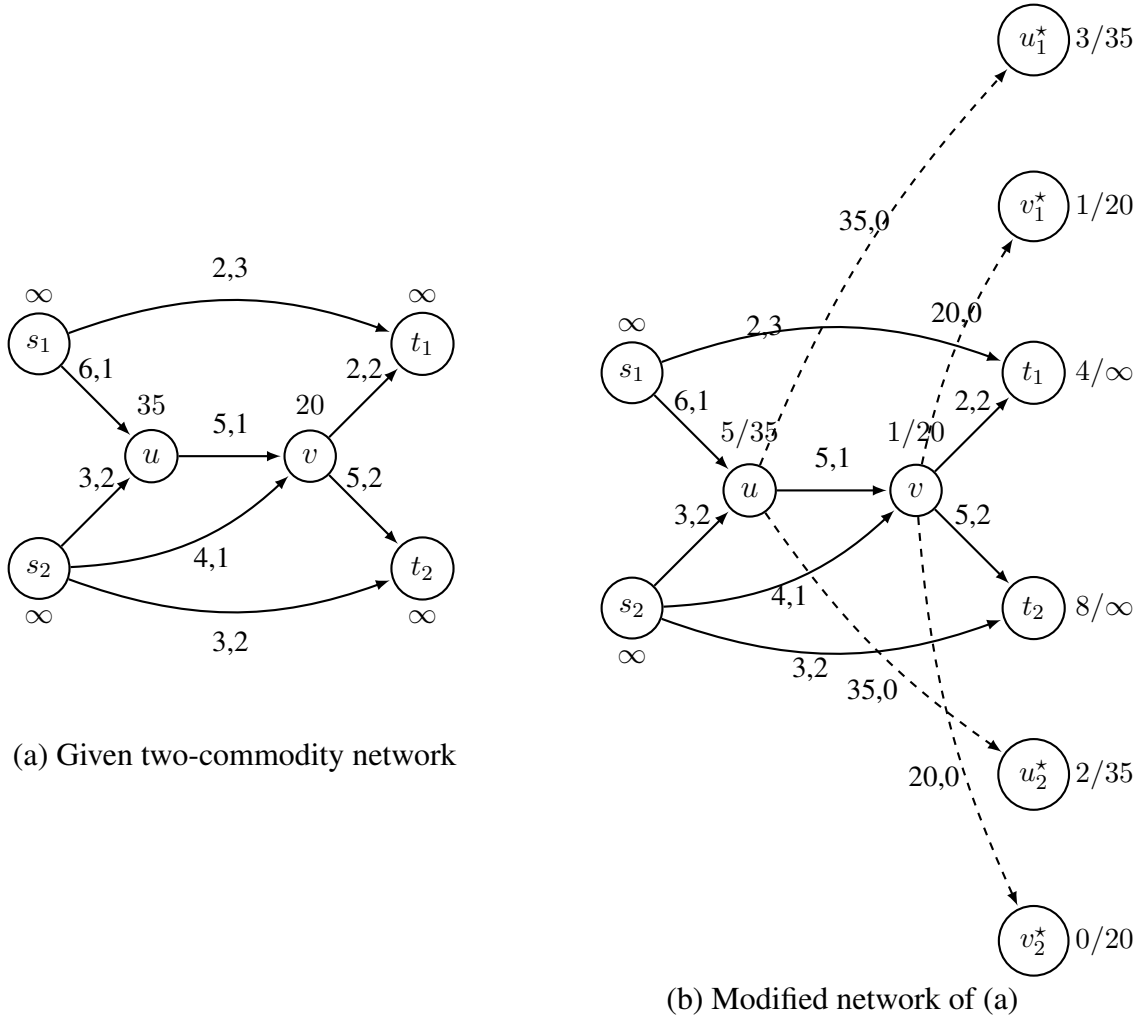


Figure 4.7: Two-commodity network (a) with arc capacity, cost and storage capacity. Figure (b) is its modified network with prioritized dummy nodes.

If the intermediate storage is not permitted, then the total flow of $\sum_{i=1}^2 d_i = 12$ units is transhipped out of which $\phi_{t_1} = 4$ and $\phi_{t_2} = 8$ units. Due to intermediate storage, the total flow transmission is $\sum_{i=1}^2 d_i = 18$ units, which is 50% more than without intermediate storage. \square

Maximum Dynamic Multi-Commodity Flow with Intermediate Storage

Flow Model. The dynamic flow model with intermediate storage for the maximum flow problem is as follows.

$$\max \sum_{i \in K} d_i \tag{4.8a}$$

such that,

$$\sum_{a \in \Gamma_{s_i}^{out}} \sum_{\theta=0}^T \Phi_a^i(\theta) = \sum_{a \in \Gamma_{t_i}^{in}} \sum_{\theta=\tau_a}^T \Phi_a^i(\theta - \tau_a) + \sum_{u \in \mathcal{I}} \hat{\Phi}_u^i(T) = d_i, \quad \forall i \in K \quad (4.8b)$$

$$\sum_{a \in \Gamma_u^{in}} \sum_{\beta=\tau_a}^{\theta} \Phi_a^i(\beta - \tau_a) - \sum_{a \in \Gamma_u^{out}} \sum_{\beta=0}^{\theta} \Phi_a^i(\beta) = \hat{\Phi}_u^i(\theta), \quad \forall u \in \mathcal{I}, i \in K, \theta \in \mathcal{T} \quad (4.8c)$$

$$0 \leq \Phi_a(\theta) = \sum_{i \in K} \Phi_a^i(\theta) \leq \kappa_a, \quad \forall a \in \mathcal{A}, \theta \in \mathcal{T} \quad (4.8d)$$

$$0 \leq \hat{\Phi}_u(\theta) = \sum_{i \in K} \hat{\Phi}_u^i(\theta) \leq \nu_u, \quad \forall u \in \mathcal{I}, \theta \in \mathcal{T} \quad (4.8e)$$

Equation (4.8a) is an objective function that maximizes the total flow $\sum_{i \in K} d_i$. The value of d_i , for each commodity i , is defined in (4.8b) as total flow out from the source s_i within time horizon T which is equal to the sum of inflow at sink t_i and the excess flow at intermediate nodes. Equation (4.8c) represents the excess flow at intermediate nodes for each time step θ . In any instance of time θ , the bundle constraint in (4.8d) is bounded by arc capacity and the boundedness of the excess flow at intermediate nodes by its storage capacity is represented by the constraint in (4.8e). The lower bound (necessary) and upper bound (sufficient) of the storage capacity of intermediate node $u \in \mathcal{I}$ is represented as $\sum_{a \in \Gamma_u^{in}} \kappa_a \leq \nu_u \leq T \sum_{a \in \Gamma_u^{in}} \kappa_a$.

Solution Procedure. As in static MCF problem with intermediate storage, we first reduce the multi-commodity flow problem into h independent sub-problems and fix the priority order of intermediate nodes. Static solution is obtained in the modified single source multi-sink network $\Pi_i^* = (\mathcal{N}_i^*, \mathcal{A}_i^*, \kappa, \nu, \tau, d_i, s_i, D_i^*, T)$ for all $i \in K$, by using Algorithm 18. As in Kappmeier (2015), we use the static multi-commodity flow on time expanded network Π^T to obtain the dynamic solution as follows.

For each i , construct a virtual sink \bar{t}_i with infinite capacity and join it to each dummy node $u_i^* \in D_i^*$ with capacity $\kappa_{[u_i^*, \bar{t}_i]} = \phi_{u_i^*}(\theta)$ and transit time $\tau_{[u_i^*, \bar{t}_i]} = 0$, where $\phi_{u_i^*}(\theta)$ is the static flow of commodity i at dummy node u_i^* at time θ . The choice of $\kappa_{[u_i^*, \bar{t}_i]} = \phi_{u_i^*}(\theta)$ is taken to assure that the flow while sending back from the dummy nodes must be on respective nodes. Now, for each $i \in K$, the super network $\tilde{\Pi}_i^*$ is obtained by adding virtual sink and arcs in Π_i^* so that it becomes a single source single sink network with prioritized intermediate nodes. Let $\Pi^T = (\mathcal{N}^T, \mathcal{A}^T = \mathcal{A}_M \cup \mathcal{A}_H \cup \mathcal{A}_s \cup \mathcal{A}_t, \kappa, K, \nu, \tau, d_i, \hat{S}, \hat{D}, T)$ be the time expanded network of super network $\tilde{\Pi}^*$ which is obtained by including virtual sinks, super source and super sink in Π_i^* . The components in time expanded network Π^T are defined as follows.

$$\begin{aligned} \mathcal{N}^T &= \{u(\theta) : u \in \mathcal{N}, \theta \in \mathcal{T}\} \cup \{u_i^*(\theta)\} \cup \{\bar{t}_i(\theta) : i \in K, \theta \in \mathcal{T}\} \cup \{s'_i, \bar{t}'_i : i \in K\} \cup \{\tilde{s}, \tilde{t}\} \\ \mathcal{A}_M &= \{(u(\theta), v(\theta + \tau_a)) : a = (u, v) \in \mathcal{A}, \theta \in \mathcal{T}\} \cup \{(u(\theta), u_i^*(\theta)) \cup (u_i^*(\theta), \bar{t}_i(\theta)) : u \in \mathcal{I}, \theta \in \mathcal{T}\} \\ \mathcal{A}_H &= \{(u(\theta), u(\theta + 1)) : u \in \mathcal{N}, \theta \in \mathcal{T}\} \cup \{(u_i^*(\theta), u_i^*(\theta + 1)) \cup (\bar{t}_i(\theta), \bar{t}_i(\theta + 1)) : i \in K, \theta \in \mathcal{T}\} \end{aligned}$$

$$\begin{aligned}
\mathcal{A}_s &= \{(\tilde{s}, s'_i) : i \in K\} \cup \{(s'_i, s_i(\theta)) : i \in K, \theta \in \mathcal{T}\} \\
\mathcal{A}_t &= \{(\tilde{t}, \bar{t}_i) : i \in K\} \cup \{(\bar{t}_i, \bar{t}_i(\theta)) : i \in K, \theta \in \mathcal{T}\} \\
\hat{S} &= \{\tilde{s}, s'_i, s_i(\theta) : i \in K, \theta \in \mathcal{T}\} \\
\hat{D} &= \{\tilde{t}, \bar{t}_i, \bar{t}_i(\theta) : i \in K, \theta \in \mathcal{T}\}.
\end{aligned}$$

Kappmeier (2015) has shown that the static multi-commodity flow on the time expanded network is equivalent to the dynamic multi-commodity flow on the original network.

Theorem 4.8 (Kappmeier (2015)). *For a given dynamic network Π with time horizon T , a feasible static $\hat{S} - \hat{D}$ multi-commodity flow ϕ^T in the time expanded network Π^T yields the feasible dynamic multi-commodity flow Φ in Π such that $|\phi^T| = |\Phi|$.*

We now present an algorithm to solve the dynamic MCF problem with intermediate storage by using time expanded network.

Algorithm 19: Maximum dynamic MCF algorithm with intermediate storage

Input : Given dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \nu, \tau, d_i, S, D, T)$.

Output: Maximum dynamic multi-commodity flow with intermediate storage in Π .

1. Reconfigure the multi-commodity flow problem into h independent single commodity flow problems by reallocating the capacity of bundle arcs and fix the priority order by considering the transit times as cost.
 2. Construct a super network $\tilde{\Pi}^*$ by including virtual sinks, super source and super sink in Π_i^* .
 3. Construct $\Pi^T = (\mathcal{N}^T, \mathcal{A}^T = \mathcal{A}_M \cup \mathcal{A}_H \cup \mathcal{A}_s \cup \mathcal{A}_t, \kappa, K, \nu, \tau, d_i, \hat{S}, \hat{D}, T)$, the time expanded network of super network $\tilde{\Pi}^*$.
 4. As in Kappmeier (2015), calculate the maximum static multi-commodity flow with intermediate storage on the time expanded network Π^T .
 5. Transform the solution to the original network Π by removing the dummy nodes, virtual sinks, super source, super sink and dummy arcs.
-

Theorem 4.9. *Algorithm 19 provides an feasible solution to the maximum dynamic multi-commodity flow problem with intermediate storage in pseudo-polynomial time complexity.*

Proof. As the static multi-commodity flow with intermediate storage is optimal from Theorem 4.7, every static flow in time expanded network is feasible. Next, the time complexity of static multi-commodity flow with intermediate storage is polynomial. As the dynamic solution in time expanded network not only depend on input size of m and n but also in time horizon T , Algorithm 19 solves the maximum dynamic multi-commodity flow problem with intermediate storage in pseudo-polynomial time. \square

Example 4.7. Consider the network presented in Figure 4.7 by considering the arc cost as the transit time. Let the time horizon be $T = 5$. The prioritization of intermediate nodes and creating dummy nodes are as similar to Example 4.6. After creating the dummy nodes, we add

the virtual sink \bar{t}_i for each i , so that it reduces to commodity-wise single source single sink $(s_i - \bar{t}_i)$ problem (see Figure 4.8).

As in Kappmeier (2015), We calculate the maximum static multi-commodity flow with intermediate storage in time expanded network, where maximum static flow is calculated from minimum cost flow by considering the transit time as cost. The time expanded network of Figure 4.8 with time horizon $T = 5$ is presented in Figure 4.9. At last, dummy nodes and arcs are removed to obtain the maximum dynamic flow with intermediate storage and flow at dummy nodes are back to their original nodes.

The flow pattern in two paths of commodity-1 and three paths of commodity-2 with intermediate storage is presented in Table 4.1. Total amount of flow leaving from the source s_1 of commodity-1 within time $T = 5$ is 36 units out of which 10 units reach to the sink t_1 by storing 18 and 8 units at u and v , respectively. Similarly, 44 units of flow of commodity-2 leaves the source s_2 in same time horizon out of which 11, 8 and 25 units reaching to u , v and t_2 , respectively. Thus, total of 80 units of flow is transshipped through given network so that 35 units reaches to the sinks by holding 29 units at u and 16 units at v . We can see that if intermediate storage is not allowed then only 35 units of flow is transshipped. Thus due to intermediate storage, 128.57% of the flow is increased.

Observation 4.1. If we consider a two-way dynamic multi-commodity network with contraflow configuration as $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \nu, \tau, d_i, S, D, T)$, then auxiliary network $\bar{\Pi}$ can be constructed with symmetric transit time as described Subsection 2.4.1 or orientation-dependent transit time described respectively in Subsection 2.4.2 for symmetric or asymmetric network. We first fix the priority order of nodes in the given network Π and then construct an auxiliary network $\bar{\Pi}$. Static flow with intermediate storage is obtained in auxiliary network where orientation of arcs are fixed as per the static flow computation. We apply algorithm 19 on an auxiliary network to get pseudo-polynomial time solution (see in Khanal et al. (2021)).

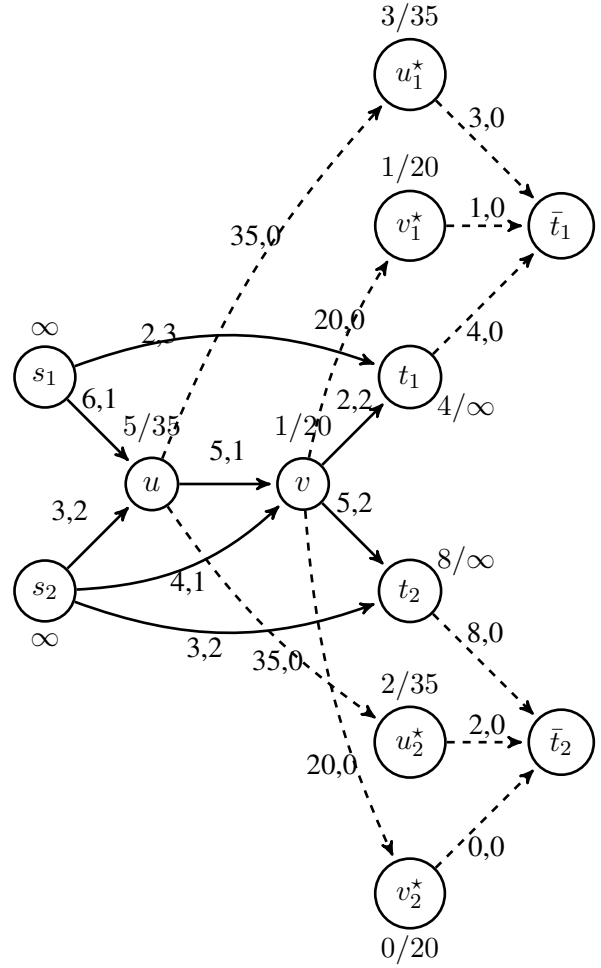


Figure 4.8: Super network $\tilde{\Pi}^*$ with dummy nodes and virtual sinks.

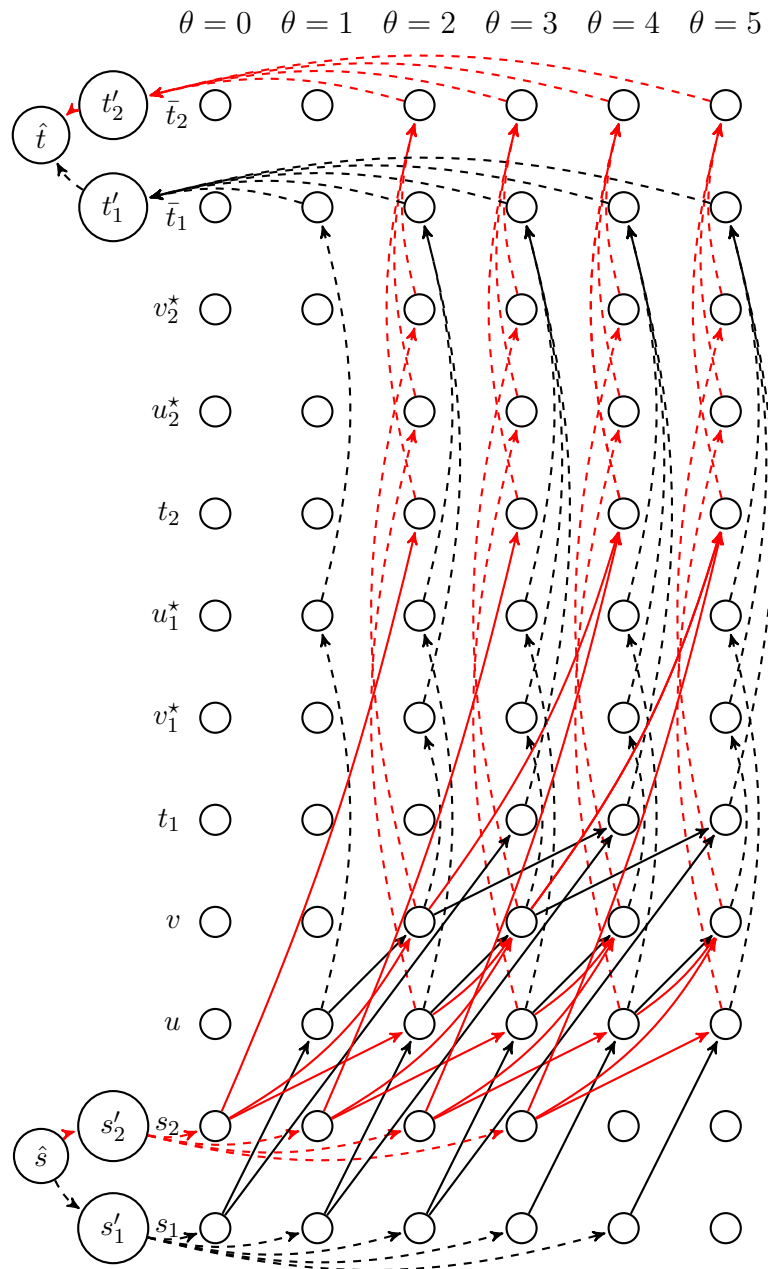


Figure 4.9: Time expanded multi-commodity network flow with intermediate storage, where black and red colors are for commodity-1 and commodity-2, respectively. Dotted arcs represent the dummy arcs.

Table 4.1: Multi-commodity flow with intermediate storage in each time θ

commodity-1				commodity-2					
Start time at s_1	flow at			Reaching time at last node	Start time at s_2	flow at			Reaching time at last node
	u	v	t_1			u	v	t_2	
Path: s_1-t_1				Path: s_2-t_2					
$\theta =0$	×	×	2	$\theta =3$ at t_1	$\theta =0$	×	×	3	$\theta =2$ at t_2
$\theta =1$	×	×	2	$\theta =4$ at t_1	$\theta =1$	×	×	3	$\theta =3$ at t_2
$\theta =2$	×	×	2	$\theta =5$ at t_1	$\theta =2$	×	×	3	$\theta =4$ at t_2
					$\theta =3$	×	×	3	$\theta =5$ at t_2
Path: $s_1-u-v-t_1$				Path: $s_2-u-v-t_2$					
$\theta =0$	3	1	2	$\theta =4$ at t_1	$\theta =0$	2	0	1	$\theta =5$ at t_2
$\theta =1$	3	1	2	$\theta =5$ at t_1	$\theta =1$	3	×	×	$\theta =3$ at u
$\theta =2$	3	3	×	$\theta =4$ at v	$\theta =2$	3	×	×	$\theta =4$ at u
$\theta =3$	3	3	×	$\theta =5$ at v	$\theta =3$	3	×	×	$\theta =5$ at u
$\theta =4$	6	×	×	$\theta =5$ at u	Path: s_2-v-t_2				
					$\theta =0$	×	0	4	$\theta =3$ at t_2
					$\theta =1$	×	0	4	$\theta =4$ at t_2
					$\theta =2$	×	0	4	$\theta =5$ at t_2
					$\theta =3$	×	4	×	$\theta =4$ at v
					$\theta =4$	×	4	×	$\theta =5$ at v
Total	18	8	10	Total $d_1=36$	Total	11	8	25	Total $d_2=44$

Observation 4.2. If the storage capacity of the intermediate nodes is sufficient, then the approximate maximum MCF with intermediate storage can be obtained by using TRF defined in Chapter 2 after sharing the capacity of bundle arc by proportional capacity sharing.

4.2 Quickest Multi-commodity Flow

For the given demand d_i of each commodity $i \in K$, quickest multi-commodity flow problem concerns with minimization of makespan T to satisfy the given demand. In this section, we concern with solving quickest MCF problem by using proportional as well as flow-dependent capacity sharing on bundle arcs. We also solve the quickest multi-commodity contraflow problem with inflow-dependent transit times using length bound and Δ -condensed techniques by reversing the necessary arcs.

4.2.1 Quickest MCF with Proportional and Flow-dependent Capacity Sharing

Using proportional capacity sharing technique presented in equation (4.5), multi-commodity flow problem is reduced to h independent single commodity flow problems where commodity-wise capacity is fixed on each bundle arc. So, polynomial time solution is possible by using scaling algorithm of Lin & Jaillet (2015). Similarly, using flow-dependent capacity sharing

technique presented in equation 4.6, pseudo-polynomial time solution is possible in time expanded layer graph because the flow a commodity may vary over time. Due to sharing of the capacities on bundle arcs using ceiling and floor functions, the solutions obtained are approximate solutions.

(i) Quickest MCF with Proportional Capacity Sharing

As the quickest flow problem is an inverse problem of maximum dynamic flow problem, the quickest multi-commodity flow problem with proportional capacity sharing can be represented by rearranging the temporally repeated flow of discrete time setting as

$$\begin{cases} T_i^* = \min \frac{d_i + \sum_{a \in \mathcal{A}} \tau_a \phi_a^i}{|\phi^i|} - 1, & (|\phi^i| \neq 0) \\ \text{satisfying } \kappa_a^i = \frac{\kappa_e^i}{\sum_{e \in P_{[s_i, u]}; i \in K} \kappa_e^i} \kappa_a & \text{and constraints in (4.1b - 4.1c)} \end{cases} \quad (4.9)$$

where, ϕ_a^i denotes the feasible static flow of commodity i on arc a (see in Lin & Jaillet (2015) for single commodity). Similarly, τ_a and $|\phi^i|$ denote the arc cost (or transit time) and the value of static flow for commodity i , respectively. κ_a^i is the shared capacity of bundle arc a for the commodity i as derived in equation (4.5). The quickest time to satisfy all the demands d_i is $T^* = \max\{T_i^* : i \in K\}$.

Solution by Cost-scaling Approach. The quickest flow problem is to search a smallest possible time by sending maximum amount of flow from the source to the sink iteratively over time as long as the demand is not fulfilled. By using Newton's method and Megiddo's parametric search Megiddo (1979), Burkard et al. (1993) developed a polynomial time algorithm based on the min-cost flow problem. Lin & Jaillet (2015) extended the cost-scaling algorithm of Goldberg & Tarjan (1990) which solves the quickest flow problem within the same time bound as the min-cost flow problem. The solution strategy of quickest contraflow problem using cost-scaling approach can be found in Pyakurel et al. (2018). Based on the cost-scaling algorithm of Lin & Jaillet (2015) and Pyakurel et al. (2018), we present Algorithm 20 to solve the quickest MCF problem with proportional capacity sharing on the bundle arcs as follows.

Solution procedure starts with reduction of the multi-commodity flow problem to h independent single commodity sub-problems by sharing the capacity of bundle arcs proportionally for each commodity $i \in K$. The dual variables corresponding to the flow conservation constraints, node potentials π and the residual network Π^r are introduced to minimize the ratio

$$T_i = \frac{d_i + \sum_{a \in \mathcal{A}} \tau_a \phi_a^i}{|\phi^i|} - 1.$$

The reduced cost $c_a(\pi)$ of arc $a = (u, v)$ in Π^r is obtained by $c_a(\pi) = \pi(v) - \pi(u) + \tau_a$ which we consider, as in Lin & Jaillet (2015), not less than $-\epsilon$, for some $\epsilon > 0$. Initially, we consider

Algorithm 20: The cost-scaling algorithm for the quickest MCF

Input : Given multi-commodity flow network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$.

Output: Quickest time to satisfy the demand d_i on Π

1. Construct h independent sub-problems by proportional capacity sharing (4.5) on bundle arcs for all $i \in K$.
 2. Set the node potential $\pi(u) = 0$ for all $u \in \mathcal{N}$, $\phi_a^i = 0$ for all $a \in \mathcal{A}$ and $\epsilon = \hat{\tau} = \max_{a \in \mathcal{A}} \{\tau_a\}$.
 3. Improve the 2ϵ -optimal flow into the ϵ -optimal flow.
 4. Reduce the gap between T_i and the difference of potential $\pi(s_i) - \pi(t_i)$.
 5. Scale ϵ by $\frac{1}{2}$ and repeat Step 3 and Step 4 if $\epsilon \geq \frac{1}{8n_i}$.
 6. **If** T_i is more than the time (cost) of the shortest simple path in residual network Π^r **then** saturate the static flow ϕ^i by sending the maximum flow from s_i to t_i in the sub-network $\hat{\Pi}$ containing those arcs which are on some shortest path in residual network Π^r .
 7. **end if**
 8. $T = \max\{T_i : i \in K\}$.
-

the node potential $\pi = 0$, arc flow $\phi_a = 0$ and $\epsilon = \hat{\tau} = \max_{a \in \mathcal{A}} \{\tau_a\}$. Then, each 2ϵ -optimal flow is improved to an ϵ -optimal one by assigning

$$\phi_a^i = \begin{cases} 0 & \text{if } c_a(\pi) > 0 \\ \kappa_a^i & \text{if } c_a(\pi) < 0 \end{cases} \quad \forall a \in \mathcal{A}.$$

Any static flow ϕ^i is said to be ϵ -optimal if the reduced cost is not less than $-\epsilon$. Push/Relabel algorithm is used for the active nodes having positive excess. To reduce the gap between T_i and $\pi(t_i) - \pi(s_i)$, extra flow is created at the source node s_i , then the admissible flow is pushed through the arcs in Π^r to the sink node t_i and node potentials are relabeled, if required. Then ϵ is scaled by $\frac{1}{2}$ and the process is continued until $\epsilon < \frac{1}{8n_i}$, where n_i is the number of nodes associated with commodity i . If T_i obtained from above by scaling phases is more than the cost in simple shortest $s_i - t_i$ path in Π^r , the flow is then saturated by sending maximum $s_i - t_i$ flow in sub-network $\hat{\Pi}$, where $\hat{\Pi}$ is formed from the residual network Π^r by taking those arcs which are on some shortest path from s_i to t_i . Lastly, we take $T = \max\{T_i : i \in K\}$ which is the quickest possible time to satisfy all the demands of the original network Π .

Here, we first prove the optimality condition for quickest MCF problem and then the correctness of Algorithm 20. For the notational simplicity, we denote $|\phi^i|$ by y^i as a parameter and $f(y^i) = \sum_{a \in \mathcal{A}} \tau_a \phi_a^i$. With the help of this, the objective function of the problem in equation (4.9) can be rewritten as

$$T_i^* = \min \frac{d_i + f(y^i)}{y^i} - 1 = \min T_i^*(y^i), \quad y^i > 0$$

Theorem 4.10 (Lin & Jaillet (2015)). *The function $T_i^*(y^i)$ has local minimum at flow value y^i if and only if*

$$-1 - c_{t_i - s_i} \leq T_i^*(y^i) \leq c_{s_i - t_i}$$

where, $-c_{t_i-s_i}$ and $c_{s_i-t_i}$ are the costs of shortest t_i-s_i and s_i-t_i paths in the residual network of flow y^i , respectively.

Proof. The function $T_i^*(y^i)$ has local minimum at y^i if for arbitrary small $\epsilon > 0$,

$$T_i^*(y^i) = \frac{d_i + f(y^i)}{y^i} - 1 \leq \frac{d_i + f(y^i + \epsilon)}{y^i + \epsilon} - 1 = T_i^*(y^i + \epsilon) \quad (4.10a)$$

and,

$$T_i^*(y^i) = \frac{d_i + f(y^i)}{y^i} - 1 \leq \frac{d_i + f(y^i - \epsilon)}{y^i - \epsilon} - 1 = T_i^*(y^i - \epsilon). \quad (4.10b)$$

Here, $f(y^i + \epsilon)$ represents the increment of flow value y^i by ϵ in cost function $f(y^i)$, which means sending of extra ϵ amount of flow from s_i to t_i maintaining the min-cost flow. For this, extra flow is sent through the shortest path in residual network Π^r with cost $\epsilon c_{s_i-t_i}$. That is, $f(y^i + \epsilon) = f(y^i) + \epsilon c_{s_i-t_i}$. Using this relation in equation (4.10a), we get

$$\begin{aligned} \frac{d_i + f(y^i)}{y^i} - 1 &\leq \frac{d_i + f(y^i) + \epsilon c_{s_i-t_i}}{y^i + \epsilon} - 1 \\ \Rightarrow c_{s_i-t_i} &\geq \frac{d_i + f(y^i)}{y^i} \\ \therefore T_i^*(y^i) &= \frac{d_i + f(y^i)}{y^i} - 1 < \frac{d_i + f(y^i)}{y^i} \leq c_{s_i-t_i} \end{aligned} \quad (4.11)$$

As similar to the above case, $f(y^i - \epsilon)$ represents the reduction of flow value y^i by ϵ in cost function $f(y^i)$, which means sending ϵ amount of flow back from t_i to s_i with cost $\epsilon(-c_{t_i-s_i})$. That is, $f(y^i - \epsilon) = f(y^i) - \epsilon(-c_{t_i-s_i})$. Using this relation in equation (4.10b), we get

$$\begin{aligned} -c_{t_i-s_i} &\leq \frac{d_i + f(y^i)}{y^i} \\ \Rightarrow -c_{t_i-s_i} - 1 &\leq \frac{d_i + f(y^i)}{y^i} - 1 \\ \therefore T_i^*(y^i) &= \frac{d_i + f(y^i)}{y^i} - 1 \geq -c_{t_i-s_i} - 1 \end{aligned} \quad (4.12)$$

□

Theorem 4.11. For each commodity i , the TRF of a feasible static flow ϕ^i is optimal if and only if ϕ^i is a min-cost flow with flow value $|\phi^i|$ and satisfies

$$-1 - c_{t_i-s_i} \leq \frac{d_i + \sum_{a \in A} \tau_a \phi_a^i}{|\phi^i|} - 1 \leq c_{s_i-t_i}. \quad (4.13)$$

Proof. For quickest MCF problem in (4.9), if we consider the flow value $|\phi^i| = y^i$ as a parameter, then the objective function is the sum of $\frac{d_i}{y^i}$ and $\frac{1}{y^i}$ times the min-cost flow problem, and less by 1. The min-cost flow problem can be written as follows.

$$\begin{cases} f(y^i) = \min \sum_{a \in \mathcal{A}} \tau_a \phi_a^i \\ \text{subject to, } \kappa_a^i = \frac{\kappa_e^i}{\sum_{e \in P_{[s_i, u]}; i \in K} \kappa_e^i} \kappa_a \text{ and the constraints (4.1b - 4.1c)} \end{cases} \quad (4.14)$$

The objective function of the problem in (4.9) is to minimize

$$T_i^*(y^i) = \frac{d_i + f(y^i)}{y^i} - 1, \quad y^i > 0.$$

From Theorem 4.10, the flow value y^i is a local minimum for the function $T_i^*(y^i)$ if and only if $-1 - c_{t_i - s_i} \leq T_i^*(y^i) \leq c_{s_i - t_i}$ holds. Moreover, the problem in (4.14) is linear programming problem with cost minimization and $f(y^i)$ is convex with piecewise linear in y^i . So $T_i^*(y^i)$ is a unimodal function and a local minimum of $T_i^*(y^i)$ is the global minimum (Lin & Jaillet (2015)).

Let \bar{y}^i be an optimal flow value for $T_i^*(y^i)$. Then $-1 - c_{t_i - s_i} \leq T_i^*(\bar{y}^i) \leq c_{s_i - t_i}$. Also, if $\bar{\phi}^i$ is a feasible flow with value \bar{y}^i , then $f(\bar{y}^i) = \min \sum_{a \in \mathcal{A}} \tau_a \bar{\phi}_a^i$ if and only if $\bar{\phi}^i$ is a min-cost flow. As a consequence,

$$T_i^*(\bar{y}^i) = \frac{d_i + \sum_{a \in \mathcal{A}} \tau_a \bar{\phi}_a^i}{\bar{y}^i} - 1$$

if and only if $\bar{\phi}^i$ is a min-cost flow with flow value \bar{y}^i . This concludes that, feasible flow $\bar{\phi}^i$ with value \bar{y}^i is an optimal solution to the problem in (4.9) if and only if $\bar{\phi}^i$ is a min-cost flow and

$$-1 - c_{t_i - s_i} \leq \frac{d_i + \sum_{a \in \mathcal{A}} \tau_a \bar{\phi}_a^i}{\bar{y}^i} - 1 \leq c_{s_i - t_i}.$$

Hence, the TRF of static flow $\bar{\phi}^i$ is an optimal solution to the QMCF with proportional capacity sharing of bundle arcs for all $i \in K$. \square

Theorem 4.12. *Algorithm 20 solves the quickest MCF with proportional capacity sharing in polynomial time.*

Proof. The running time of Step 1 of Algorithm 20 is $O(m|K|)$, where $|K| = h \leq n/2$ denotes the given number of commodities and the running time of Steps 3 and 4 are $O(n^3)$. Due to the cost-scaling phase in Step 5, Steps 3 and 4 are repeated at most $O(\log(n\hat{\tau}))$ times. Similarly, the running time of Step 6 is $O(n^3 \log(n))$. So the time complexity of Algorithm 20 is $O(n^3 m |K| \log(n\hat{\tau}))$. Hence, the quickest MCF problem with proportional capacity sharing can be solved in polynomial time. \square

Example 4.8. Consider a two-commodity network presented in Figure 4.10(a), where commodity-1 is transshipped from s_1 to t_1 with demands $d_1 = 35$ and commodity-2 is transshipped from s_2 to t_2 with demands $d_2 = 25$. In order to solve the quickest flow problem, first we use proportional capacity sharing on the bundle arc (u, v) . It provides the capacity of 2 units each for both commodities (as shown in Figure 4.10(b)). Then we convert this problem into two independent sub-problems.

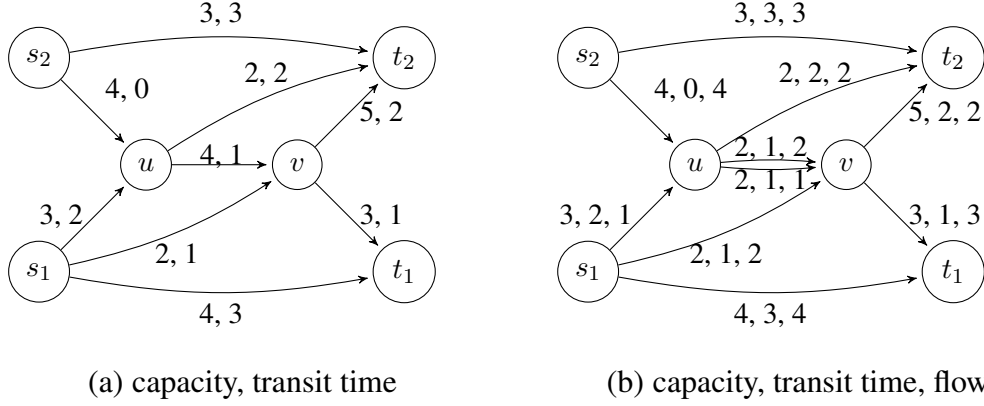


Figure 4.10: (b) represents the network with proportional capacity sharing of (a).

Sub-problem 1: We have to find the quickest time to satisfy the demands $d_1 = 35$ from $s_1 - t_1$. By using Algorithm 20, the quickest time for commodity-1 after proportional capacity sharing is,

$$T_1 = \min \frac{d_1 + \sum_{a \in \mathcal{A}} \tau_a \phi_a^1}{|\phi^1|} - 1 = \frac{55}{7} - 1 = 6.85$$

Sub-problem 2: For given demands $d_2 = 25$, we have to find the quickest time to send flow from $s_2 - t_2$. As above, the quickest time for commodity-2 after proportional capacity sharing is,

$$T_2 = \min \frac{d_2 + \sum_{a \in \mathcal{A}} \tau_a \phi_a^2}{|\phi^2|} - 1 = \frac{44}{7} - 1 = 5.28$$

Thus, the quickest time (in integer) to satisfy both demands is $T = 7$. \square

(ii) Quickest MCF with Flow-dependent Capacity Sharing

The quickest multi-commodity flow problem with flow-dependent capacity sharing can be represented as

$$\begin{cases} \min T \\ \text{satisfying } \kappa_a^i(\theta) = \frac{\Phi_e^i(\theta - \tau_e)}{\sum_{e \in \alpha(a): i \in K} \Phi_e^i(\theta - \tau_e)} \kappa_a \text{ and constraints in (4.1b) - (4.1c)} \end{cases} \quad (4.15)$$

where $\kappa_a^i(\theta)$ is flow-dependent capacity sharing of bundle arc capacity at each time step θ as described in equation (4.6).

Solution Procedure. As a solution strategy, we first construct the temporal paths $P(\theta) \in \mathcal{P}^i(\theta)$, which start from source s_i at time θ and reach to the corresponding sink t_i at time $\theta + \tau_P$ through the arcs $a \in P$. These temporal paths can be visualized in time expanded layer graph. Let $\mathcal{P}(\theta) = \{\cup \mathcal{P}^i(\theta) : i \in K\}$ and $\tau'_P = \min\{\tau_P : P \in \mathcal{P}^i\}$ be set of all temporal paths starting from source at time θ and the length of shortest path in the network, respectively. Denote $\phi^{i,P}(\theta)$ as the amount of time-dependent static flow on temporal paths $P(\theta) \in \mathcal{P}^i(\theta)$ which may vary over time due to flow-dependent capacity sharing on bundle arcs. If two temporal paths $P^i(\theta_1)$ and $Q^j(\theta_2)$ of different commodities $i, j \in K$ meet at $u(\theta)$ with $\theta_1, \theta_2 \leq \theta$, then the capacity along the arc $a = (u(\theta), v(\theta + \tau_a))$ is shared by using flow-dependent capacity sharing. We first construct a time expanded layer graph within the time horizon $T = \tau'_P$ and calculate the flow value of each commodity within this time using flow-dependent capacity sharing. In each iteration, we gradually increase the time horizon by unit time (i.e., $T = \tau'_P + 1$) whenever the demand of each commodity is not satisfied. Flow of the commodity, whose demand is already satisfied, is taken as zero on further construction of time expanded layer graph. The algorithmic framework of solution procedure is as follows.

Algorithm 21: Quickest MCF algorithm with flow-dependent capacity sharing

Input : Given multi-commodity flow network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$.

Output: Quickest time with flow-dependent capacity sharing on Π .

1. Set $T = \tau'_P = \min\{\tau_P : P \in \mathcal{P}^i\}$ and $\Phi^i(T) = 0$.
 2. **while** $\Phi^i < d_i, \forall i \in K$, **do**
 - (a) Send the static flow ϕ^i on time expanded layer graph with sharing the capacity on bundle arc of temporal paths using flow-dependent capacity sharing (4.6).
 - (b) Compute the flow value of each commodity within time horizon T as $\Phi^i(T)$
 - (c) Set $\phi^i = 0$ if $\Phi^i(T) \geq d_i$.
 - (d) Update $\Phi^i(T) = \Phi^i(T) + \phi^i$, and $T = T + 1$.
 3. **end while**
 4. $T =$ Quickest time to satisfy all the demands d_i .
-

Theorem 4.13. *Algorithm 21 provides an approximate solution to the quickest MCF problem with flow-dependent capacity sharing in pseudo-polynomial time.*

Proof. As Step 2(a) shares the capacity on bundle arc using flow-dependent capacity sharing and integer solution is obtained by ceiling $\lceil \cdot \rceil$ and floor $\lfloor \cdot \rfloor$ functions, the solution obtained from Algorithm 21 is an approximate solution. Since the shared capacity in bundle arc depends on the inflow rate of the flow of predecessor arcs and the sharing process is continued on the layer graph until all the demand $d_i, \forall i \in K$ is satisfied, the running time of Algorithm 21 depends on demand d_i . Therefore, the time complexity of the quickest multi-commodity flow over time problem with flow-dependent capacity sharing is pseudo-polynomial. \square

4.2.2 Inflow-dependent Quickest MCF with Partial Contraflow

In general, traversal time from one location to another location depends on the amount of flow (vehicles) on the road, known as flow-dependent transit time. One can realize this fact while traveling on the urban roads with high traffic congestion. Flow-dependent transit times can be classified in two ways: inflow-dependent transit times and load-dependent transit times. Köhler et al. (2002) introduced the quickest single source single sink flow problem with the setting of inflow-dependent transit times in which the current rate of flow is measured when it enters on the arc and moves with constant speed throughout the arc. Similarly, Köhler & Skutella (2005) introduced a model in which, at any moment of time, the actual speed of the flow depends on the current amount of the flow on the arc (i.e., load). Both of these flow problems are related with single commodity flow.

To solve the quickest multi-commodity flow problem with inflow-dependent transit times, Hall et al. (2007a) presented two approximation approaches: one length bound and another condensed time expanded network. In this subsection, we define a symmetric inflow-dependent transit time function on the anti-parallel arcs in the given network and present an inflow-dependent transit time function in auxiliary network. By using partial contraflow, we present two approximation algorithms, one by using T -length bounded function and another fully polynomial time approximation scheme (FPTAS) by using condensed time expanded network. A flow is said to be T -length bounded if it uses the paths of length at most T . Due to \mathcal{NP} -hardness of the problem, we adopt the relaxation technique of bow graph and solve the inflow-dependent quickest multi-commodity flow problem in two-way network using partial contraflow.

The term partial contraflow is the contraflow technique in which only necessary arcs are reversed and rest of the arcs are saved so that it can be used for further purposes. We have to be careful that in multi-commodity network flow, sources of some commodities may lie towards the sink of others so that flow can be bi-directional in some bundle arcs. Our assumption is that flows of the same commodity in opposite directions effectively cancel, while flows of different commodities are accumulated in either directions. The main idea behind the partial contraflow for a two-way multi-commodity network with arcs $a = (u, v)$ and $\overleftarrow{a} = (v, u)$ is as follows.

- Arc $\overleftarrow{a} = (v, u)$ is reversed if and only if either flow along the arc $a = (u, v)$ is greater than its capacity or there is positive flow along the arc $a = (u, v) \notin \mathcal{A}$. If $\phi_a > \kappa_a$, $\phi_{\overleftarrow{a}} < \kappa_{\overleftarrow{a}}$ and $\bar{\kappa}_a > \phi_a + \phi_{\overleftarrow{a}}$, where $\bar{\kappa}_a = \kappa_a + \kappa_{\overleftarrow{a}}$, then the arc \overleftarrow{a} is reversed partially and the unused capacity of arc \overleftarrow{a} is saved.
- If $\phi_a > \kappa_a$, $\phi_{\overleftarrow{a}} = 0$ and $\bar{\kappa}_a = \phi_a$, then we have to reverse the arc \overleftarrow{a} completely. Thus no capacity is saved in \overleftarrow{a} .
- If $\phi_a < \kappa_a$ and $\phi_{\overleftarrow{a}} < \kappa_{\overleftarrow{a}}$, then neither arc a nor arc \overleftarrow{a} is reversed. In this case, the remaining capacity of arc a and \overleftarrow{a} are saved in their respective directions.

Inflow-dependent Transit Times. In the previous models, we deal the problems with constant transit times on the arcs. But in the real road network, transit time is constant only up to some flow value, known as a free flow, and then it depends on the inflow rate of each additional flow. Mathematically, inflow-dependent transit time in general (one-way) network can be presented as

$$\tau_a(\phi) = \begin{cases} \tau_0 & \text{for } 0 < \phi \leq \phi_0 \\ \tau_0 + \tau'(\phi) & \text{for } \phi > \phi_0 \end{cases} \quad \forall a \in \mathcal{A}, \quad (4.16)$$

where τ_0 represents the constant transit time of free flow ϕ_0 and $\tau'(\phi)$ is the inflow-dependent transit time for additional flow. Our assumption is that, if there is no flow then no transit time is considered, i.e., if $\phi = 0$ then $\tau = 0$. Geometrically, inflow-dependent transit time function and its relaxation to piecewise constant function in some arbitrary precision is shown in Figure 4.11.

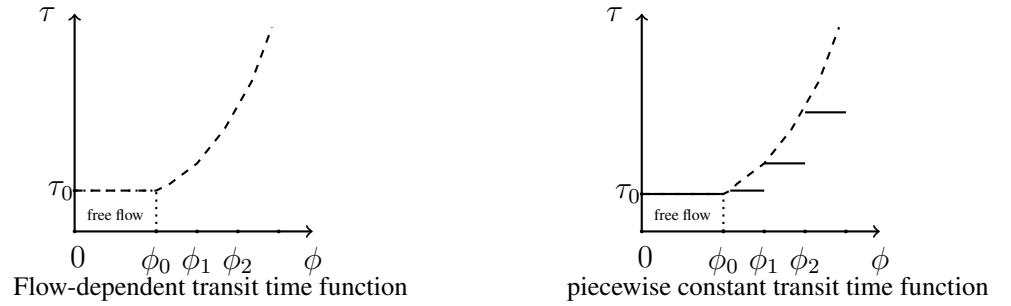


Figure 4.11: Flow-dependent transit time function and transformation to the piecewise constant.

For the purpose of contraflow configuration, we define the symmetric inflow-dependent transit times as follows. Let $\tau_a(\phi)$ and $\tau_{\overleftarrow{a}}(\phi)$ be symmetric inflow-dependent transit times on a two-way network such that

$$\tau_{\overleftarrow{a}}(\phi) = \tau_a(\phi) = \begin{cases} \tau_0 & \text{for } 0 < \phi \leq \phi_0 \\ \tau_0 + \tau'_a(\phi) & \text{for } \phi > \phi_0 \end{cases} \quad \forall a, \overleftarrow{a} \in \mathcal{A}. \quad (4.17)$$

Considering $\tau'_a(\phi) = \tau_0 \alpha \left(\frac{\phi_a}{\kappa_a}\right)^{\beta'}$ and $\tau'_a(\phi) = \tau_0 J \left(\frac{\phi_a}{\kappa_a - \phi_a}\right)$, above transit time function becomes BPR function and Davidson's function, respectively. According to Sheffi (1984), $\alpha = 0.15$, $\beta' = 4$ and $J = 0.1$ are commonly used in usual practice. Pyakurel et al. (2019) studied the effect of lane reversals on the Kathmandu road network using the BPR function and Davidson's function by taking free flow time on the roads. They also compared the effect of quickest time with inflow-dependent transit time and the quickest time with constant transit time by considering three types of constant transit times, the upper bound on the step function, average of the step function values, and the free flow transit times.

After contraflow configuration, the free flow value in auxiliary network increases approximately up to double on the same free flow time τ_0 , and flow-dependent transit time of per additional

flow decrease by half due to increase in the capacity of arc. The transit time function on auxiliary network after lane reversals is

$$\bar{\tau}_a(\phi) = \begin{cases} \tau_0 & \text{for } 0 < \phi \leq 2\phi_0 \\ \tau_0 + \tau'_a(\frac{\phi}{2}) & \text{for } \phi > 2\phi_0 \end{cases} \quad \forall a \in \bar{\mathcal{A}}. \quad (4.18)$$

Moreover, $\bar{\tau}_a = \tau_a$ or $\tau_{\bar{a}}$, if either $\bar{a} \notin \mathcal{A}$ or $a \notin \mathcal{A}$, respectively. To simplify the notation, we use $\tau_a(\phi)$, $a \in \bar{\mathcal{A}}$ instead of $\bar{\tau}_a(\phi)$.

Flow Model. Flow models presented in previous sections are in discrete time and using natural transformation, continuous time dynamic flow can also be computed equivalently. For the dynamic flow with inflow-dependent transit time, flow models are to be considered in continuous time settings because of transit times on the arcs are not constant but flow-dependent. As in Hall et al. (2007a), we generalize the multi-commodity dynamic contraflow model with inflow-dependent transit times as follows. The flow of commodity i entering arc a at time θ at flow rate $\Phi_a^i(\theta)$ arrive at the head of arc a at time $\theta + \tau_a(\Phi_a(\theta))$ for all $a \in \bar{\mathcal{A}}$ and $\theta \in [0, T)$ with $\Phi_a(\theta) > 0$ and $\theta + \tau_a(\Phi_a(\theta)) < T$. For each $u \in \mathcal{N}$, the total inflow and outflow of the commodity $i \in K$ at time $\beta \in [0, T)$ are defined by $\sum_{a \in \Gamma_u^{in}} \int_{\theta \geq 0: \theta + \tau_a(\Phi_a(\theta)) \leq \beta} \Phi_a^i(\theta) d\theta$ and

$\sum_{a \in \Gamma_u^{out}} \int_0^\beta \Phi_a^i(\theta) d\theta$, respectively. Thus the flow model for inflow-dependent quickest flow problem defined on an auxiliary network is as follows.

$$\min T \quad (4.19a)$$

such that,

$$\sum_{a \in \Gamma_u^{in}} \int_{\theta \geq 0: \theta + \tau_a(\Phi_a(\theta)) \leq T} \Phi_a^i(\theta) d\theta - \sum_{a \in \Gamma_u^{out}} \int_0^T \Phi_a^i(\theta) d\theta = \begin{cases} d_i & \text{if } u = s_i \\ -d_i & \text{if } u = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in K \quad (4.19b)$$

$$\sum_{a \in \Gamma_u^{in}} \int_{\theta \geq 0: \theta + \tau_a(\Phi_a(\theta)) \leq \beta} \Phi_a^i(\theta) d\theta - \sum_{a \in \Gamma_u^{out}} \int_0^\beta \Phi_a^i(\theta) d\theta \geq 0 \quad \forall u \in \mathcal{I}, i \in K, \beta \in \mathcal{T} \quad (4.19c)$$

$$0 \leq \Phi_a(\theta) = \sum_{i \in K} \Phi_a^i(\theta) \leq \bar{\kappa}_a, \quad \forall a \in \bar{\mathcal{A}}, i \in K, \theta \in \mathcal{T} \quad (4.19d)$$

Solution Procedure. To solve the quickest multi-commodity flow problem with inflow-dependent transit time, Hall et al. (2007a) presented a strategy based on strong relaxation of inflow-dependent transit time in a bow graph with piecewise constant attribute. They developed an approximation technique based on length bounded static flow, which is modified to inflow-preserving flow to produce provably good solutions. It is then transformed into a feasible multi-commodity flow over time with inflow-dependent transit times and bounded cost. The authors also presented FPTAS, which uses a condensed time expanded network to solve the

problem with inflow-dependent transit time and bounded cost. By adopting these techniques in two-way network, we use relaxation of inflow-dependent transit time for the partial contraflow configuration.

Relaxation of Inflow-dependent Transit Times Using Bow Graph. The bow graph is an extension of the original graph by creating a bunch of parallel arcs to reflect the inflow-dependent transit time. As in Köhler et al. (2002), a strong relaxation of the dynamic flow is essential to replace the inflow-dependent transit time function τ_a of an arc $a \in \bar{\mathcal{A}}$ by piecewise constant, non-decreasing, and left continuous function τ_a^{step} . The transit time function τ is said to be left continuous at ϕ_l if $\sup\{\tau(\phi'_l) : \phi'_l < \phi_l\} = \tau(\phi_l)$. The breakpoints for τ_a^{step} are taken as $\phi_0 < \phi_1 < \dots < \phi_r$ with corresponding transit times $\tau_0 < \tau_1 < \dots < \tau_r$, where τ_0 is the transit time for the free flow $0 < \phi \leq \phi_0$ and τ_k represents the transit time for the flow entering arc a at flow rate $(\phi_{k-1}, \phi_k]$ for all $k = 1, \dots, r$, (see Figure 4.11).

We denote the bow graph by $\bar{\Pi}^b = (\mathcal{N}, \bar{\mathcal{A}}^b)$ having the same set of nodes as in $\bar{\Pi}$ but each arc $a \in \bar{\mathcal{A}}$ is replaced by a set of parallel bow arcs $a_k \in \bar{\mathcal{A}}_a^b$ with corresponding transit times τ_k and capacity ϕ_k , for all $k = 0, 1, \dots, r$. The cost function on each arc remains the same as in original network. The notation $\bar{\mathcal{A}}^b$ represents the set of all bow arcs, and $\bar{\mathcal{A}}_a^b$ represents the set of bow arcs associated with an original arc $a \in \bar{\mathcal{A}}$.

The approximation of non-decreasing left continuous transit time function by the step function can be presented in the following observation.

Observation 4.3 (Köhler et al. (2002)). Let $\delta, \eta > 0$. For every non-negative, non-decreasing, and left continuous transit time function $\tau : [0, \kappa] \rightarrow \mathbb{R}^+$, there exists a step function $\tau^{step} : [0, \kappa] \rightarrow \mathbb{R}^+$ with

- i. $\tau^{step}(\phi) \leq \tau(\phi) \leq (1 + \eta)\tau^{step}(\phi) + \delta$ for every $\phi \in [0, \kappa]$,
- ii. the number of break points of τ^{step} is bounded by $\lceil \log_{1+\eta}(\frac{\tau(\kappa)}{\delta}) \rceil + 1$.

Every dynamic flow function Φ in $\bar{\Pi}$ with inflow-dependent transit times τ_a^{step} and time horizon T corresponds to some dynamic flow function Φ^b in $\bar{\Pi}^b$ with constant transit time and same time horizon T . But the converse may not be true because the flow is scattered in different bow arcs with different transit times. An additional property is essential to address this problem, termed as inflow-preserving flow.

Inflow-preserving flow. A dynamic flow function Φ in $\bar{\Pi}$ is inflow-preserving if, for every original arc $a \in \bar{\mathcal{A}}$ and every point in time θ , function Φ^b sends flow into at most one bow arc $a_k \in \bar{\mathcal{A}}_a^b$ (Hall et al. (2007)).

In bow graph, the capacity of bunch of parallel bow arcs lies between 0 to κ_a and due to inflow-preserving property, only one of the bow arcs is chosen depending on the inflow rate of flow.

Observation 4.4 (Hall et al. (2007a)). Every dynamic inflow-preserving flow Φ^b in $\bar{\Pi}^b$ with

time horizon T corresponds to a dynamic flow function Φ in $\overline{\Pi}$ with inflow-dependent transit time $(\tau_a^{step})_{a \in \overline{\mathcal{A}}}$ and time horizon T , and vice versa.

Example 4.9. Consider an arc $a = (u, v) \in \overline{\mathcal{A}}$ with capacity $\kappa_a = 3$ and transit time function

$$\tau_a(\phi) = \begin{cases} 1 & \text{for } 0 < \phi \leq 1 \\ 1 + 2\phi & \text{for } \phi > 1. \end{cases}$$

Taking $\{0, 1, 2, 3\}$ as break points of κ_a , the lower step function of $\tau_a(\phi)$ becomes

$$\tau_a^{step}(\phi) = \begin{cases} 1 & \text{for } 0 < \phi \leq 1 \\ 3 & \text{for } 1 < \phi \leq 2 \\ 5 & \text{for } 2 < \phi \leq 3. \end{cases}$$

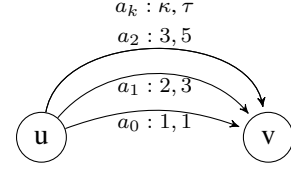


Figure 4.12: Bow graph of arc a with respect to the step function $\tau_a^{step}(\phi)$

In inflow-dependent transit time, flow is splitted into three bow arcs with capacity $\phi_0 = \phi_{a_0} = 1$, $\phi_1 = \phi_{a_1} = 2 - 1 = 1$ and $\phi_2 = \phi_{a_2} = 3 - 2 = 1$. The total flow value at $T = 10$ is $|\Phi| = \sum_{a_k} (T - \tau_{a_k}^{step}) \phi_{a_k} = 9 + 7 + 5 = 21$.

Next, for inflow-preserving flow, flow is sent from a_2 for time units in $(0, 5]$ at flow rate $\phi_2 = 3$ and then from a_1 for time units in $(5, 7]$ at flow rate $\phi_1 = 2$ and at last from a_0 for time units in $(7, 9]$ at flow rate $\phi_0 = 1$ so that total flow within time $T = 10$ is $|\Phi| = 15 + 4 + 2 = 21$. This illustrates the Observation 2. \square

Since inflow-preserving flow over time is a non-convex function, as in Hall et al. (2007a), weakly inflow-preserving flow in $\overline{\Pi}^b$ is used for the convexification of flow.

Weakly Inflow-preserving flow. Let $\mu_{a_k}(\theta) = \frac{\Phi_{a_k}^b(\theta)}{\kappa_{a_k}}$ be the per capacity flow rate on arc $a_k \in \overline{\mathcal{A}}^b$ at time θ . Then the weakly inflow-preserving flow is the flow over time Φ^b in $\overline{\Pi}^b$ with time horizon T satisfying $\sum_{a_k \in \overline{\mathcal{A}}_a^b} \mu_{a_k}(\theta) \leq 1$.

Example 4.10. Consider the arc as explained in Example 4.9. The per capacity flow rate are $\mu_{a_0} = \frac{\phi_0}{1}$, $\mu_{a_1} = \frac{\phi_1}{2}$ and $\mu_{a_2} = \frac{\phi_2}{3}$. Thus, the weakly inflow-preserving flow is

$$|\Phi| = \phi_0(T - 1) + \phi_1(T - 3) + \phi_2(T - 5) = \mu_{a_0}(T - 1) + 2\mu_{a_1}(T - 3) + 3\mu_{a_2}(T - 5).$$

At time $T = 12$,

$$|\Phi| = 11\mu_{a_0} + 18\mu_{a_1} + 21\mu_{a_2} = 11(\mu_{a_0} + \mu_{a_1} + \mu_{a_2}) + 7(\mu_{a_1} + \mu_{a_2}) + 3(\mu_{a_2}) \leq 21.$$

So, to satisfy the same flow value $|\Phi| = 21$ units in Example 4.9, the minimum time required is $T = 12$. \square

Based on these relaxations, we now present the solution procedure by using the length bound approach and the condensed time expanded network as follows.

(i) Length Bound Approximation

The solution procedure starts with the transformation of given network to an auxiliary network. We construct the bow graph according to the lower step function of $\bar{\tau}_a(\phi)$ and use T -length bound approximation to calculate the quickest weakly inflow-preserving flow. A flow is said to be T -length bound if it is obtained by using the paths of length at most T . This flow is converted to inflow-preserving flow by using the δ -resting property which corresponds to the dynamic flow in $\bar{\Pi}$. A dynamic flow Φ is δ -resting if, for every node $u \in \mathcal{N} \setminus S$ and $\delta > 0$, all flow arriving at u is stored there for at least δ time units before it moves on. Partial contraflow is applied on the static flow obtained on T -length bound approximation and the flow is repeated temporally over the time to obtain the dynamic flow unless the demand is not satisfied.

Let us denote the optimal solution to weakly inflow-preserving dynamic multi-commodity flow with time horizon T by Φ^w and the static multi-commodity flow obtained from Φ^w by ϕ^w , where ϕ^w is obtained by averaging over time interval $(0, T]$, i.e., $\phi_{a_k}^{w,i} = \frac{1}{T} \int_0^T \Phi_{a_k}^{w,i}(\theta) d\theta$, for all $a_k \in \bar{\mathcal{A}}^b$ and $i \in K$. Here, flow $\phi_{a_k}^{w,i}$ is T -length bounded, satisfies $\frac{1}{T}$ part of demand of Φ^w and has cost $\frac{1}{T}$ part of the cost of Φ^w . Also, Φ^w being weakly inflow-preserving flow, its per capacity flow $\mu_{a_k} = \frac{\phi_{a_k}^w}{\kappa_{a_k}}$, $a_k \in \bar{\mathcal{A}}^b$ satisfies the condition $\sum_{a_k \in \bar{\mathcal{A}}^b} \mu_{a_k} \leq 1$ for all $a \in \bar{\mathcal{A}}$.

The weakly inflow-preserving static flow ϕ^w in $\bar{\Pi}^b$ can be turned to the dynamic flow within time horizon $2T$ as follows. Send the flow into each T -length bounded $s_i - t_i$ paths P obtained by path decomposition such that $\tau(P) = \sum_{a \in P} \tau_a \leq T$ with $a \in \bar{\mathcal{A}}^b$ for T time units and wait for at most another T times until all flow arrives at the destination. But by Handler & Zang (1980), T -length bounded flow problem is \mathcal{NP} -hard. So relaxation of T -length bound is essential to compute polynomial time approximate solution which gives $(2 + \epsilon)$ -approximation for quickest weakly-inflow preserving flow.

Theorem 4.14 (Hall et al. (2007a)). *For a weakly inflow-preserving dynamic multi-commodity flow Φ^w with time horizon T and cost/budget B , there exists a polynomial time quickest weakly inflow-preserving dynamic multi-commodity flow with time horizon $(2 + \epsilon)T$ and cost B , for every $\epsilon > 0$.*

Now, we present Algorithm'22 to solve the inflow-dependent quickest multi-commodity flow problem with partial reversal of arcs as follows.

Theorem 4.15. *The weakly inflow-preserving dynamic flow Φ^w with time horizon T^w , generated by a static flow ϕ^w , can be turned into dynamic flow Φ with inflow-dependent transit time $(\tau_a)_{a \in \bar{\mathcal{A}}}$ and time horizon T , where T is bounded from above by $(1 + \eta)T^w + 2n\delta$.*

Proof. Since lower step function τ^{step} does not correspond exactly with the original transit time function τ , we define a transit time function $\tau'_{a_k} = (1 + \eta)\tau_{a_k} + \delta$, $\forall a_k \in \bar{\mathcal{A}}^b$ which

Algorithm 22: Length bound approximation algorithm to inflow-dependent quickest MCF with partial contraflow

Input : Given multi-commodity network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$.

Output: The inflow-dependent quickest MCF with partial contraflow.

1. Transform network Π to auxiliary network $\bar{\Pi}$ by adding two-way capacities as $\bar{\kappa}_a = \kappa_a + \kappa_{\bar{a}}$ and inflow-dependent transit time $\bar{\tau}_a(\phi)$, $\forall a \in \bar{\mathcal{A}}$.
 2. Replace the transit time function $\tau_a(\phi)$, $a \in \bar{\mathcal{A}}$ by lower step function $\tau_a^{step}(\phi)$ and construct corresponding bow graph $\bar{\Pi}^b$.
 3. Compute $(2 + \epsilon)$ -approximate quickest weakly inflow-preserving flow by T -length bound as in Hall et al. (2007a).
 4. Convert weakly inflow-preserving flow in $\bar{\Pi}^b$ to inflow-preserving flow by using δ -resting property which corresponds to flow over time in $\bar{\Pi}$.
 5. Decompose the flow along the $s_i - t_i$ paths and cycles and remove the cycles $\forall i$.
 6. Reverse $\bar{a} \in \mathcal{A}$ up to capacity $\phi_a - \kappa_a$ iff $\phi_a > \kappa_a$ and $\phi_{\bar{a}} < \kappa_{\bar{a}}$, replaced κ_a by 0 whenever $a \notin \mathcal{A}$, where $\phi_a = \sum_i \phi_a^i$ and $\phi_{\bar{a}} = \sum_i \phi_{\bar{a}}^i$. Similar for reverse direction.
 7. For each $a \in \mathcal{A}$, if \bar{a} is reversed, $sc(\bar{a}) = \bar{\kappa}_a - \phi_a - \phi_{\bar{a}}$ and $sc(a) = 0$. If neither a nor \bar{a} is reversed, $sc(a) = \kappa_a - \phi_a > 0$, where $sc(a)$ is the saved capacity of a and similarly, $sc(\bar{a}) = \kappa_{\bar{a}} - \phi_{\bar{a}} > 0$.
 8. Transform the solution to the original network.
-

corresponds to the bow graph of the lower step function $\tau'_a{}^{step}(\phi) = (1 + \eta)\tau_a^{step}(\phi) + \delta$ for all $\phi \in (0, \kappa_a]$. Denote the set of $s_i - t_i$ paths obtained by length bounded path decomposition of static flow $\phi^{b,i}$ as \mathcal{P}^i and the set of all such paths by $\mathcal{P} = \cup_{i=1}^k \mathcal{P}^i$. The weakly inflow-preserving flow Φ^w is sent along path P at constant rate ϕ^P . The transit time to reach the destination before the increase in the transit time is $\tau(P) = \sum_{a_k \in P} \tau_{a_k}$ and after increasing is $\tau'(P) = \sum_{a_k \in P} \tau'_{a_k} \leq (1 + \eta)\tau(P) + n\delta$. As $\tau(P) \leq T^w$, so $\tau'(P) \leq (1 + \eta)T^w + n\delta$. Repeat this process by increasing the transit time by factor of δ so that weakly inflow-preserving flow in the bow graph becomes the δ -resting. The flow is now turned to inflow-preserving (Hall et al. (2007a)), and by Observation 4.4, it yields the dynamic flow with inflow-dependent transit times. The time horizon is thus $(1 + \eta)T^w + 2n\delta$. \square

Theorem 4.16. *Algorithm 22 provides the T -length bound approximate solution to the inflow-dependent quickest multi-commodity contraflow problem with bounded cost using partial contraflow configuration.*

Proof. As Steps 1 and 8 are transformation of the network without violating the flow conservation and capacity constraints, they are feasible. According to Observation 4.3, the construction of the bow graph in Step 2 is also feasible. Step 5 decomposes the flow along the paths and cycles and removes the positive flows in cycles. Steps 6 and 7 reverse the necessary arc capacities and save unused capacities, which are feasible. Similarly, the feasibility of Steps 3 and 4 can be found in Theorems 4.14 and 4.15, respectively.

A dynamic multi-commodity flow problem reduces to a static flow problem on $\bar{\Pi}^b$ by using a length bound approximation that provides polynomial time bound. An approximate solution to the quickest flow with inflow-dependent transit time on $\bar{\Pi}^b$ can be obtained from Theorems 4.14 and 4.15. Moreover, any optimal solution on $\bar{\Pi}^b$ is equivalent to the feasible solution to given network Π . Thus, an approximate solution to the inflow-dependent quickest multi-commodity contraflow problem with bounded cost can be computed for the given network Π . \square

Example 4.11. Consider a network with two commodities in which commodity-1 is to be transhipped from s_1 to t_1 with demand $d_1 = 25$ and commodity-2 from s_2 to t_2 with demand $d_2 = 20$ units. The capacity and transit time of each arc are as indicated in Figure 4.13. The feasible paths for commodity-1 and commodity-2 are $P_1 = s_1 - u - v - t_1$ and $P_2 = s_2 - u - v - t_2$. In the case without contraflow, the bottleneck capacity on both paths are 2 units each. Inflow-dependent transit times on both paths are presented in Table 4.2.

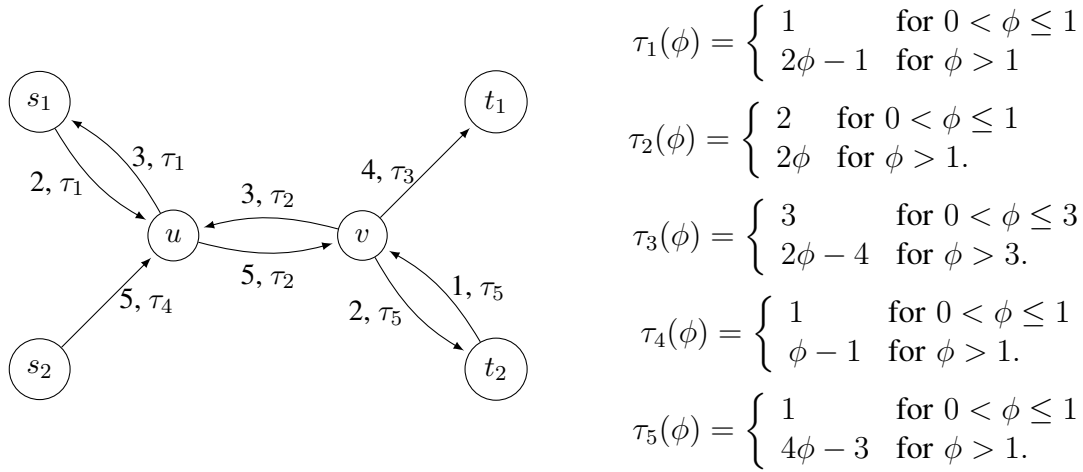


Figure 4.13: Two-commodity two-way network with symmetric inflow-dependent transit times.

Table 4.2: Inflow-dependent transit times on paths of given network without lane reversals.

P_1	$s_1 - u$	$u - v$	$v - t_1$	Total time	P_2	$s_2 - u$	$u - v$	$v - t_2$	Total time
$\Phi = 1$	1	2	3	6	$\Phi = 1$	1	2	1	4
$\Phi = 2$	3	4	3	10	$\Phi = 2$	1	4	5	10

To satisfy the demand $d_1 = 25$ without contraflow, 13-length bound paths are essential and flow is sent for next 13 time units so that demand is satisfied within $[25, 26)$ times. But there is no path of length more than 10, so flow is temporally repeated to this path for 13 unit times so that the demand is fulfilled in 23 time units. Similarly, to satisfy demand $d_2 = 20$, 10-length bound paths are essential and flow is send for the next 10 time units so that the demand is satisfied within $[19, 20)$ times. Thus, both demands are satisfied within time $T = 23$.

Figure 4.14(a) represents an auxiliary network of Figure 4.13 which is obtained by adding two-way capacities and transit times are obtained from equations (4.18). Similarly, Figure 4.14(b)

represents the saved arc capacities on arcs. The flow-dependent transit times for feasible flow on each path are presented in the Table 4.3. The bottleneck capacities of paths $P_1 = s_1 - u - v - t_1$ and $P_2 = s_2 - u - v - t_2$ are 4 and 3 units, respectively.

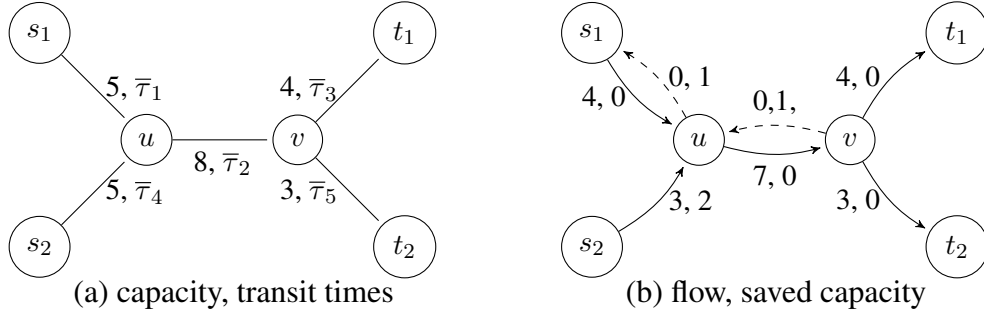


Figure 4.14: (a) represents the two-commodity auxiliary network and (b) represents the static flow with saved arc capacities.

Table 4.3: Inflow-dependent transit time on each path.

P_1	$s_1 - u$	$u - v$	$v - t_1$	Total time	P_2	$s_2 - u$	$u - v$	$v - t_2$	Total time
$\Phi = 1$	1	2	3	6	$\Phi = 1$	1	2	1	4
$\Phi = 2$	1	2	3	6	$\Phi = 2$	1	2	1	4
$\Phi = 3$	2	3	3	8	$\Phi = 3$	2	3	3	8
$\Phi = 4$	3	4	4	11					

To satisfy the demand $d_1 = 25$ units with lane reversals, the path of 9-length bound is essential, and flow is sent for the next 9 times so that demand is satisfied within time interval $[17, 18)$. As there is no path of length 9, we send flow on the 8-length bound path for the next 9 times and demand is satisfied in 17-time units. Similarly, to satisfy demand $d_2 = 20$ units, the path of 8-length bound is essential, and flow is sent for the next 7 times so that demand is satisfied within time interval $[14, 15)$. So, the total time to satisfy both demands after lane reversals is $T = 17$. By comparing the quickest time with and without contraflow, approximately 26.1% of the time is saved due to lane reversal strategy. \square

(ii) An FPTAS by Δ -Condensed Time Expanded Graph

Here, we present an FPTAS to solve the quickest MCF problem with inflow-dependent transit times and bounded cost by using partial contraflow configuration. We consider the Δ -condensed time expanded network as defined in Dhamala et al. (2020). As in Hall et al. (2007a), we develop an FPTAS in the network $\bar{\Pi}$ by defining lower, upper, and 2Δ -lengthened bow graphs as follows.

Let $\epsilon > 0$ be an arbitrary positive number. By rounding down $\tau_a(\phi)$ to the nearest multiple of Δ , a lower step function $\tau_a^\downarrow = \lfloor \frac{\tau_a(\phi)}{\Delta} \rfloor \Delta$, for $a \in \bar{\mathcal{A}}$ with $\phi \leq \bar{\kappa}_a$ is defined, where Δ is taken as $\frac{\epsilon^2 T}{n}$. With the help of this lower step function, the lower bow graph $\bar{\Pi}^{b,\downarrow}$ is constructed.

By lengthening the transit time of each lower bow arc by Δ , i.e., $\tau_a^\uparrow(\phi) = \tau_a^\downarrow(\phi) + \Delta$, upper Δ -lengthened bow graph $\bar{\Pi}^{b,\uparrow}$ is obtained. Similarly, by lengthening the transit time of each lower bow arc by 2Δ , i.e., $\tau_a^{\uparrow\uparrow}(\phi) = \tau_a^\downarrow(\phi) + 2\Delta$, upper 2Δ -lengthened bow graph $\bar{\Pi}^{b,\uparrow\uparrow} = (\mathcal{N}, \bar{\mathcal{A}}^{\uparrow\uparrow}, K, \bar{\kappa}', \tau^{\uparrow\uparrow}, d_i, S, D, T)$ with $\bar{\kappa}'_a = \Delta(\kappa_a + \kappa_{\bar{a}})$ is obtained.

For time horizon T , construct a Δ -condensed time expansion $\bar{\Pi}^F = (\mathcal{N}^F, \bar{\mathcal{A}}^F)$ of $\bar{\Pi}^{b,\uparrow}$, known as fan graph. Every arc $a = (u, v) \in \bar{\mathcal{A}}$ is represented in the bow graph $\bar{\Pi}^{b,\uparrow}$ by its expansion $\bar{\mathcal{A}}^{\uparrow\uparrow}$. Consequently, for each point of time $\theta \in \{\rho\Delta : \rho = 0, \dots, \lceil T/\Delta \rceil - 1\}$, the fan graph consists of a "fan" of arcs $\bar{\mathcal{A}}_a^F(\theta) = \{a_k(\theta) : a_k \in \bar{\mathcal{A}}_a^{\uparrow\uparrow}, \theta + \tau_{a_k}^{\uparrow\uparrow} \leq (\lceil T/\Delta \rceil - 1)\Delta\}$, where $a_k(\theta) = (u(\theta), v(\theta + \tau_{a_k}^{\uparrow\uparrow}))$.

For a static flow ϕ in fan graph $\bar{\Pi}^F$, define per capacity inflow value on arc $a \in \bar{\mathcal{A}}^F$ by $\mu_{a_k} = \frac{\phi_{a_k}}{\kappa_{a_k}}$. Similarly, as defined above, the static flow ϕ is weakly inflow-preserving if $\sum_{a_k \in \bar{\mathcal{A}}_a^F(\theta)} \mu_{a_k} \leq 1, \forall a \in \bar{\mathcal{A}}$ and $\theta \in \{\rho\Delta : \rho = 0, \dots, \lceil T/\Delta \rceil - 1\}$. Moreover, any weakly inflow-preserving static flow in $\bar{\Pi}^F$ corresponds to a weakly inflow-preserving flow over time in $\bar{\Pi}^{b,\uparrow}$ and vice-versa. The algorithmic framework is as follows.

Algorithm 23: An FPTAS for inflow-dependent quickest MCF with partial contraflow

Input : Given multi-commodity flow network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, K, \tau, d_i, S, D, T)$.

Output: The inflow-dependent quickest MCF with partial contraflow.

1. The auxiliary network $\bar{\Pi}$ is transformed to Δ -condensed auxiliary network $\bar{\Pi}^{b,\uparrow\uparrow} = (\mathcal{N}, \bar{\mathcal{A}}^{\uparrow\uparrow}, K, \bar{\kappa}', \tau^{\uparrow\uparrow}, d_i, S, D, T)$ with $\tau_a^{\uparrow\uparrow}(\phi) = \tau_a^\downarrow(\phi) + 2\Delta$ and $\bar{\kappa}'_a = \Delta(\kappa_a + \kappa_{\bar{a}})$.
 2. Compute approximate quickest weakly inflow-preserving flow on $\bar{\Pi}^F$ by using FPTAS as in Hall et al. (2007a).
 3. Decompose the flow along the $s_i - t_i$ paths and cycles and remove the cycles $\forall i$.
 4. Reverse $\bar{a} \in \bar{\mathcal{A}}$ up to capacity $\phi_a - \kappa_a$ iff $\phi_a > \kappa_a$ and $\phi_{\bar{a}} < \kappa_{\bar{a}}$, replaced κ_a by 0 whenever $a \notin \bar{\mathcal{A}}$, where $\phi_a = \sum_i \phi_a^i$ and $\phi_{\bar{a}} = \sum_i \phi_{\bar{a}}^i$. Similar for reverse direction.
 5. For each $a \in \bar{\mathcal{A}}$, if \bar{a} is reversed, $sc(\bar{a}) = \bar{\kappa}_a - \phi_a - \phi_{\bar{a}}$ and $sc(a) = 0$. If neither a nor \bar{a} is reversed, $sc(a) = \kappa_a - \phi_a > 0$, where $sc(a)$ is the saved capacity of a and similarly, $sc(\bar{a}) = \kappa_{\bar{a}} - \phi_{\bar{a}} > 0$.
 6. Transform the solution to the original network.
-

Theorem 4.17. *An approximate solution to the inflow-dependent quickest MCF problem with contraflow configuration can be computed in fully polynomial time complexity.*

Proof. Since all other steps except Steps 2 and 3 are computed in linear time, so the complexity of Algorithm 23 is dominated by Steps 2 and 3. Step 3 can be solved in $O(mn)$ time. An estimation of time can be found within $O(\log \frac{1}{\epsilon})$ geometric mean binary search steps. In every search step, a weakly inflow-preserving multi-commodity flow can be computed in a condensed fan graph $\bar{\Pi}^F$.

The fan graph contains $O(\frac{n}{\epsilon^2})$ layers with $O(\frac{n^2}{\epsilon^2})$ nodes and every arc is represented in each time

layer by a fan with at most $O(\frac{mn}{\epsilon^2})$ arcs. The number of arcs in fan graph lies in $O(\frac{mn^2}{\epsilon^4})$. Hence, $\bar{\Pi}^F$ is of polynomial size. Thus, a weakly inflow-preserving multi-commodity flow with partial contraflow configuration can be computed in fully polynomial time. \square

Example 4.12. Transform the auxiliary network in Figure 4.14(a) to Δ -condensed auxiliary network $\bar{\Pi}^{\uparrow\uparrow}$ with $\Delta = 2$, $\tau_a^{\uparrow\uparrow}(\phi) = \tau_a^{\downarrow}(\phi) + 2\Delta$ and $\bar{\kappa}'_a = \Delta(\kappa_a + \kappa_{\bar{a}})$ as shown in Figure 4.15. Using lane reversals, Table 4.4 represents the flow on each path and time to satisfy demands using Δ -condensed time expanded auxiliary network $\mathcal{N}^{a\uparrow\uparrow}$. Similarly, Table 4.5 represents the same as in Table 4.4 but without lane reversals which are calculated from Figure 4.13.

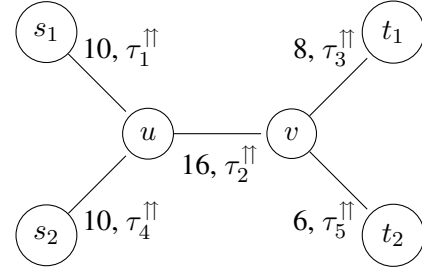


Figure 4.15: Condensed auxiliary network with capacity $\bar{\kappa}'$ and transit time $\tau^{\uparrow\uparrow}$.

In the case of contraflow configuration, the time to satisfy demand $d_1 = 25$ units is $T = 36$, and for demand $d_2 = 20$ units is $T = 32$ (cf. Table 4.4). Thus both demands are satisfied within time $T = 36$. If contraflow strategy is not applied, then time to satisfy demand $d_1 = 25$ units is $T = 42$, whereas for demand $d_2 = 20$ units is $T = 40$ (cf. Table 4.5), and so, both demands are satisfied within time $T = 42$. The graphical representation of inflow-dependent quickest time is presented in Figure 4.16. Hence, due to partial contraflow, the quickest time is reduced by 14.3%. \square

Table 4.4: Inflow-dependent transit times of paths after contraflow with $\Delta = 2$.

P_1	$s_1 - u$	$u - v$	$v - t_1$	τ_{P_1}	Time for $d_1 = 25$	P_2	$s_2 - u$	$u - v$	$v - t_2$	τ_{P_2}	Time for $d_2 = 20$
$\Phi = 1$	4	6	6	16	[64, 66]	$\Phi = 1$	4	6	4	14	[52, 54]
$\Phi = 2$	4	6	6	16	[40, 42]	$\Phi = 2$	4	6	4	14	[32, 34]
$\Phi = 3$	6	6	6	18	[34, 36]	$\Phi = 3$	6	6	6	18	[30, 32]
$\Phi = 4$	6	8	8	22	[34, 36]	$\Phi = 4$	6	8	8	22	[30, 32]
$\Phi = 5$	8	8	10	26	[34, 36]	$\Phi = 5$	8	8	10	26	[32, 34]
$\Phi = 6$	8	10	12	30	[38, 40]	$\Phi = 6$	8	10	12	30	[36, 38]
$\Phi = 7$	10	10	14	34	[40, 42]						
$\Phi = 8$	10	12	16	38	[44, 46]						

Table 4.5: Inflow-dependent transit times of paths before contraflow with $\Delta = 2$.

P_1	$s_1 - u$	$u - v$	$v - t_1$	τ_{P_1}	Time for $d_1 = 25$	P_2	$s_2 - u$	$u - v$	$v - t_2$	τ_{P_2}	Time for $d_2 = 20$
$\Phi = 1$	4	6	6	16	[64, 66]	$\Phi = 1$	4	6	4	14	[52, 54]
$\Phi = 2$	6	8	6	20	[48, 50]	$\Phi = 2$	4	8	8	20	[38, 40]
$\Phi = 3$	8	10	6	24	[40, 42]	$\Phi = 3$	6	10	12	28	[40, 42]
$\Phi = 4$	10	12	8	30	[42, 44]	$\Phi = 4$	6	12	16	34	[42, 44]

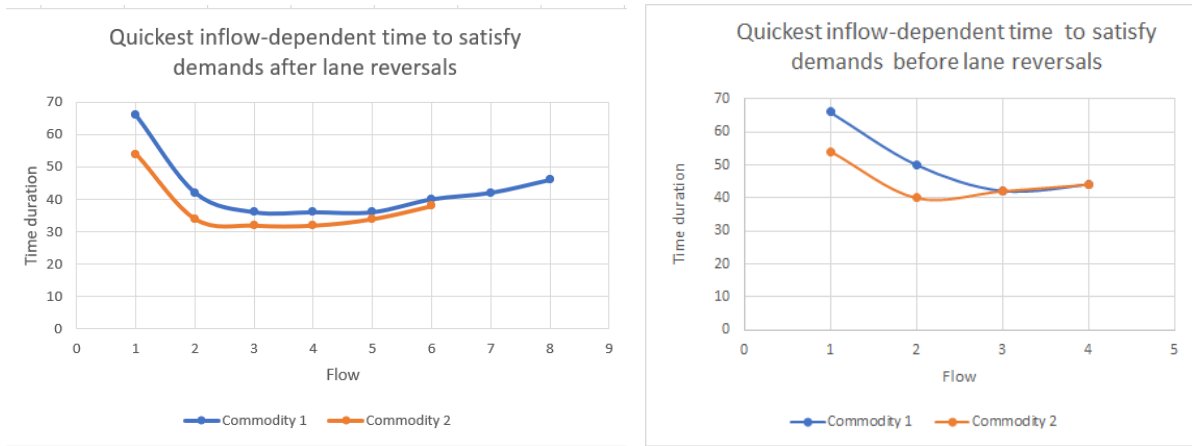


Figure 4.16: Graphical representation of inflow-dependent quickest time after and before lane reversals taking $\Delta = 2$. Due to lane reversals, quickest time is reduced significantly.

Chapter 5

Bi-level Facility Allocation Problem

Facility allocation problem concerns with establishment of facilities, specially at the time of evacuation, so that essential goods can be provided as soon as possible. Facilities can be established at nodes or arcs, due to which the capacity is reduced and can effect on the maximum flow transmission. Thus, selection of appropriate location with minimum loss in total flow transmission and optimal use of facilities is a major concern in facility allocation problem. By incorporating the maximum flow problem with location analysis, Hamacher et al. (2013) introduced single and multi-facility flow location problems. For single and multiple facility allocation, they have presented polynomial time algorithms and polynomial time heuristics, respectively. The maximum static and dynamic contraflow problems with facility location are solved in Dhungana & Dhamala (2019). The single and multiple quickest flow location problems and their solution strategies can be found in Nath et al. (2020).

Bi-level problems are two stage optimization problems in which first stage optimize the overall system under the best possible decision of the second stage. The first study in bi-level optimization can be found in von Stackelberg (1934). The exciting interest on this field of optimization is rapidly increased after the 1970s. For more detailed illustration, we refer to the papers of Anandalingam & Friesz (1992); Ben-Ayed (1993); Wen & Hsu (1991), survey papers of Colson et al. (2005); Colson et al. (2007) and the books of Bard (1998); Dempe (2002); Dempe (2020); Dempe & Zemkoho (2020).

In game theory, bi-level optimization problem is also known as leader-follower optimization problem with successive iteration of two players where the leader first optimizes its objective function knowing the follower's reaction, and the follower optimizes another objective function depending on the leader's selection.

The bi-level formulation of facility allocation problem is a major research gap in existing literature in which possibly best location can be decided to allocate the facilities and the loss in optimal flow value can be minimized as two levels of the problem. To deal with this, we formulate upper level problem which declares the appropriate location for the facility and the

lower level problem which maximize the flow on facilitated network. We present the solution procedure in two ways. The first is a naive approach with combinatorial problem for selecting the arcs to allocate the facility and find maximum dynamic flow on the network with facility allocation. The second one is Karush-Kuhn-Tucker (KKT) transformation approach.

5.1 Basic Notations

Consider a network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, s, t)$, where \mathcal{N} with $|\mathcal{N}| = n$ represents a set of n nodes and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ with $|\mathcal{A}| = m$ represents a set of m arcs. Here, $s \in \mathcal{N}$ and $t \in \mathcal{N}$ are the source (origin) and sink (destination) nodes and $\mathcal{I} = \mathcal{N} \setminus \{s, t\}$ represents the set of intermediate nodes. Each arc $a = (u, v) \in \mathcal{A}$ with $head(a) = v$ and $tail(a) = u$ has a capacity function $\kappa : \mathcal{A} \rightarrow \mathbb{R}^+$ that limits the flow on arc. Similarly, let $L \subseteq \mathcal{A}$ be a set of feasible locations and $\eta : L \rightarrow \mathbb{R}^+$ be the size of facility that is to be placed in some arc. We denote the set of outgoing arcs from node u and incoming arcs to node u by Γ_u^{out} and Γ_u^{in} , respectively. In case of dynamic network, two additional parameters are to be considered, one transit time $\tau : \mathcal{A} \rightarrow \mathbb{R}^+$ that measures the transmission time from u to v and another $\mathcal{T} = \{0, 1, \dots, T\}$ to represent time horizon T in discrete time settings. Thus the dynamic network is of the form $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, s, t, \tau, T)$.

5.2 Bi-level Problem Formulation with Facility Allocation

Let ϕ be a static flow on a network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, s, t)$ which is defined as a non-negative arc flow functions $\phi : \mathcal{A} \rightarrow \mathbb{R}^+$. The static flow model is the network flow satisfying the conditions (5.1a - 5.1c). The mathematical formulation of the maximum static flow as a linear programming problem is as follows.

$$\max |\phi| \tag{5.1a}$$

such that,

$$\sum_{a \in \Gamma_u^{out}} \phi_a - \sum_{a \in \Gamma_u^{in}} \phi_a = \begin{cases} |\phi| & \text{for } u = s \\ -|\phi| & \text{for } u = t \\ 0 & \text{for } u \in \mathcal{I} \end{cases} \tag{5.1b}$$

$$0 \leq \phi_a \leq \kappa_a \quad \forall a \in \mathcal{A} \tag{5.1c}$$

Here, equations have their usual meanings as in Section 2.2. The dynamic flow within the time horizon T can be obtained by temporally repetition of the static flow along the paths as

$$\sum_{P \in \mathcal{P}} (T + 1 - \tau_P) \phi_P = (T + 1) |\phi| - \tau_a \phi_a$$

with usual meaning of the symbols.

Now, for the purpose of bi-level formulation, we take γ and ϕ as two variables. The lower level problem is formulated as

$$\max_{\gamma} \{f(\gamma, \phi) : g(\gamma, \phi) \leq 0\} \quad (5.2)$$

which depends on the upper level variable ϕ . Here, f is a real valued function defined as $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ and $g = (g_1, \dots, g_l)$ is a vector valued function defined as $g : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^l$. Similarly, if $\Psi : \mathbb{R}^p \rightarrow 2^{\mathbb{R}^q}$ be a solution set mapping such that $\phi \in \Psi(\gamma)$, then the upper level optimization problem is of the form

$$\max_{\gamma, \phi} \{F(\gamma, \phi) : G(\gamma, \phi) \leq 0, \phi \in \Psi(\gamma)\}, \quad (5.3)$$

where $F : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ and $G : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^k$ with $G = (G_1, \dots, G_k)$. We refer to Dempe (2002); Dempe (2019) and references therein for detailed illustrations. With the help of these formulations, we introduce the bi-level formulation of maximum dynamic flow problem with allocation of facility at the arcs hereafter.

At the time of disasters, proper allocation of the location of emergency facilities for their support is very important. Let η represent the size of a facility that is to be placed at an appropriate arc in L . The upper level problem in network Π is

$$\max H(\gamma, \phi) \quad (5.4a)$$

such that,

$$0 \leq \eta \gamma_a \leq \kappa_a, \quad \forall a \in L \quad (5.4b)$$

$$\sum_{a \in L} \gamma_a = 1 \quad (5.4c)$$

$$\gamma_a = 0, \quad \forall a \in \mathcal{A} \setminus L \quad (5.4d)$$

$$\gamma_a \in \{0, 1\}, \quad \forall a \in L \quad (5.4e)$$

$$\phi \text{ solves the lower level problem depending on } \gamma \quad (5.4f)$$

where, $\gamma = (\gamma_a)_{a \in \mathcal{A}}$ and $\phi = (\phi_a)_{a \in \mathcal{A}}$. Constraint in (5.4b) represents that the facility is allocated at the arc with sufficient capacity. The single facility location is assured by equation (5.4c). Non-selection of arc outside of L for the facility allocation is represented by equation 5.4d, where (5.4e) represents the binary variable. The upper level objective function in (5.4a) is to maximize H defined by

$$H(\gamma, \phi) = (T + 1)|\phi^*| - \sum_{a \in \mathcal{A}} \tau_a \phi_a + \sum_{a \in L} \gamma_a w_a$$

where, w_a is a predefined reward function that depends on the situation of evacuation scenario. Thus, the objective of upper level problem is to maximize the flow out from the source by appropriate allocation of facility on arc. Here, $|\phi^*|$ is the value of static flow induced by ϕ after placement of the facility that is to be maximized in static flow computation.

The lower level problem is to obtain the maximum flow after reduction of the capacity at facility allocated arc by the size of facility as follows.

$$\max_{\phi^*, \phi} (T + 1)|\phi^*| - \sum_{a \in \mathcal{A}} \tau_a \phi_a \quad (5.5a)$$

such that,

$$\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a = \begin{cases} -|\phi^*| & \text{for } u = s \\ 0 & \text{for } u \in \mathcal{I} \\ |\phi^*| & \text{for } u = t \end{cases} \quad (5.5b)$$

$$0 \leq \phi_a \leq \kappa_a - \eta \gamma_a, \quad \forall a \in \mathcal{A} \quad (5.5c)$$

Here, the objective of lower level problem (5.5a) is to maximize the dynamic flow obtained by temporal repetition of static flow. The flow conservation at intermediate nodes and non-conservation of flow at source and sink are represented by equation (5.5b). The boundedness of the flow on each arc after placement of the facility is represented in (5.5c).

In case of multiple facility allocation, say r facilities of size η_i , $i = 1, \dots, r$ with $r \leq |L|$, the upper level constraint in (5.4c) is to be replaced by $\sum_{a \in L} \gamma_a = r$. Similarly, lower level constraint in (5.5c) is to be replaced by $0 \leq \phi_a \leq \kappa_a - \eta_{a,i} \gamma_a$, $\forall a \in \mathcal{A}$, $i = 1, \dots, r$, $\eta_{a,i} = \max\{\eta_i : \eta_i \text{ is associated with } a\}$. The objective function and rest of the constraints remain the same.

The dual formulation of lower level problem is as follows.

$$\min \sum_{a \in \mathcal{A}} \Theta_a (\kappa_a - \eta \gamma_a) \quad (5.6a)$$

such that,

$$\zeta_u - \zeta_v + \Theta_a \geq -\tau_a, \quad \forall a = (u, v) \in \mathcal{A} \quad (5.6b)$$

$$\zeta_s - \zeta_t \geq T + 1 \quad (5.6c)$$

$$\Theta_a \geq 0 \quad \forall a \in \mathcal{A} \quad (5.6d)$$

$$\zeta_u \in \mathbb{R} \quad (\text{unrestricted}) \quad (5.6e)$$

5.3 Solution Procedure

In this section, we present two approaches to solve the facility allocation problem. First one is a naive approach which selects an arc with some strategy to place the facility, finds the maximum dynamic flow over the time horizon and continues the process until the best solution is obtained. Another approach is the conversion of bi-level problem to single level one by using Karush-Kuhn-Tucker (KKT) transformation and solve by replacing complementarity condition by big- M method for mixed-integer reformulation.

5.3.1 Solution by Naive Approach

For a given subset $L \subseteq \mathcal{A}$ of possible locations, our concern here is to present a simple procedure to solve the facility allocation problem. The basic idea for this approach is from Stackelberg leadership model of a strategic game in which the leader makes the first move and then the follower reacts sequentially for the optimal output. The leader (upper level) iteratively chooses an arc for the allocation of a facility as long as the best optimal solution from follower (lower level) is produced.

Here, we present the pseudo codes of an algorithmic framework to solve the maximum dynamic flow problem with facility allocation. First and second steps inside ‘for loop’ of the algorithm are obtained by upper level problem and third one by lower level problem. This loop runs over all arcs $a' \in \mathcal{A}$ with $\kappa_{a'} \geq \eta$ to obtain the best optimal flow MDF^{opt} .

Algorithm 24: Naive algorithm for maximum dynamic flow with facility allocation

Input : Given a dynamic network $\Pi = (\mathcal{N}, \mathcal{A}, \kappa, s, t, \tau, T)$.

Output: MDF^{opt} = Maximum dynamic flow with facility allocation.

1. L = Set of feasible locations ($L \subseteq \mathcal{A}$).
 η = Size of facility.
 $\gamma = \{0, 1\}$, a decision variable.
 2. For $a' \in L$ with $\kappa_{a'} \geq \eta$:
Assign $\gamma_{a'} = 1$ and $\gamma_a = 0, \forall a \in L \setminus \{a'\}$.
Assign $\kappa_{a'} = \kappa_{a'} - \eta$.
 $\text{MDF}^{(a')}$ = Maximum dynamic flow after placement of facility at a' .
 3. $\text{MDF}^{\text{opt}} = \max \{\text{MDF}^{(a')} : a' \in L \text{ and } \gamma_{a'} = 1\}$
-

The time complexity of the algorithm depends on the number of iterations over the arcs in L and the complexity of the maximum dynamic flow, that is, $|L| \times O(\text{MDF})$, where $O(\text{MDF})$ is time complexity of maximum dynamic flow problem. As $|L| \leq m$ and maximum dynamic flow can be computed in polynomial time, overall time complexity of the algorithm with single facility allocation is polynomial, Hamacher et al. (2013). However, solving multiple (i.e., r) facility allocation problem is a combinatorial optimization problem with complexity $|L|P_r \times O(\text{MDF})$, where, $|L|P_r$ represents the number of permutations of $|L|$ locations taken r at a time.

5.3.2 Solution by KKT Transformation

Karush-Kuhn-Tucker (KKT) condition is one of the most commonly used approach to solve the bi-level programming problem which is only applicable if the lower level problem is a convex optimization problem. It transfers the problem into single level optimization problem. As both lower and upper level problems in our facility allocation model are linear, KKT transformation is possible. For this, consider the Lagrangian function of lower level problem as

$$\mathcal{L}(\gamma, \phi, \zeta, \Theta) = (T+1)|\phi^*| - \sum_{a \in \mathcal{A}} \tau_a \phi_a + \sum_{u \in \mathcal{I}} \zeta_u \left(\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a \right) + \sum_{a \in \mathcal{A}} \Theta_a (\kappa_a - \eta \gamma_a - \phi_a).$$

The objective function for KKT condition is the objective of upper level problem and the KKT constraints are the constraints for lower level problem (5.5b)- (5.5c) together with dual constraints (5.6b)-(5.6e) and the complementarity constraint

$$\Theta_a (\kappa_a - \eta \gamma_a - \phi_a) = 0 \quad \forall a \in \mathcal{A}.$$

This yields the mathematical program with complementarity constraints (MPCC) as follows.

$$\max_{\gamma, \phi, \zeta, \Theta} H(\gamma, \phi) \quad (5.7a)$$

such that,

$$0 \leq \eta \gamma_a \leq \kappa_a \quad \forall a \in L \quad (5.7b)$$

$$\sum_{a \in L} \gamma_a = 1 \quad (5.7c)$$

$$\sum_{a \in \Gamma_u^{in}} \phi_a - \sum_{a \in \Gamma_u^{out}} \phi_a = \begin{cases} -|\phi^*| & \text{for } u = s \\ 0 & \text{for } u \in \mathcal{I} \\ |\phi^*| & \text{for } u = t \end{cases} \quad (5.7d)$$

$$0 \leq \phi_a \leq \kappa_a - \eta \gamma_a \quad \forall a \in \mathcal{A} \quad (5.7e)$$

$$\zeta_u - \zeta_v + \Theta_a \geq -\tau_a \quad \forall a = (u, v) \in \mathcal{A} \quad (5.7f)$$

$$\zeta_s - \zeta_t \geq T + 1 \quad (5.7g)$$

$$\Theta_a (\kappa_a - \eta \gamma_a - \phi_a) = 0 \quad \forall a \in \mathcal{A}. \quad (5.7h)$$

$$\Theta_a \geq 0 \quad \forall a \in \mathcal{A} \quad (5.7i)$$

$$\zeta_u \in \mathbb{R} \quad (\text{non-restricted}) \quad (5.7j)$$

The complementarity constraints in (5.7h) can be replaced by the mixed-integer reformulation using sufficiently large big- M constants M' and M'' as follows.

$$\kappa_a - \eta \gamma_a - \phi_a \leq M'(1 - \beta_a), \quad \Theta_a \leq M'' \beta_a, \quad \beta_a \in \{0, 1\}, \quad \forall a \in \mathcal{A}.$$

This reformulation was introduced by Fortuny-Amat & McCarl (1981) so that resulting model of single level problem can be solved by standard mixed-integer solvers. However, a major concern here is to approximate the value of big- M . Pineda & Morales (2019) have shown that choosing too small big- M can result in sub-optimal solution. Similarly, too large values of big- M may cause infeasible solution for original bi-level problem (Kleinert & Schmidt (2020)). As M' is an upper bound of the primal variable ϕ_a on arcs, without loss of generality, we can set $M' = \max\{\kappa_a : a \in \mathcal{A}\}$. However, M'' is an upper bound of dual variable Θ_a and tuning such large enough constant for dual variable is a more challenging task. Trial-and-error tuning procedure is most commonly used in literature for the upper bound of the dual variable.

Due to the linear constraints in lower level problem, an ϵ bound method can be used to reformulate the complementarity constraint 5.7h and solve with relaxation in the sense of Scholtes (2001). The reformulation of complementarity constraint is

$$\Theta_a (\kappa_a - \eta \gamma_a - \phi_a) \leq \epsilon, \quad \forall a \in \mathcal{A},$$

where locally optimal solution of the problem is obtained for $\epsilon \downarrow 0$, (see also in Burtscheidt et al. (2020)). Not only for a single facility allocation, KKT transformation solves the multiple facility allocation problem with same pace.

Chapter 6

Summary and Conclusions

Disasters are unexpected circumstances, may be caused by nature or human errors, which create massive loss of infrastructures and lives. They also impact on overall economy of the people and nation. Efficient plannings for pre and post disasters are very essential though they are very challenging tasks. Different mathematical models and their applications can be found in the literature of applied mathematics. In this thesis, we have presented mathematical models for single as well as multi-commodity flow problems with and without intermediate storage, which are applicable for post disaster management.

At the time of disaster, every individual wants to move towards the safe zone as quickly and efficiently as possible. Because of which, flow at bottleneck arcs/paths creates the high congestion. Holding of the excess flow at comparatively safer intermediate shelters is the flow with intermediate storage. In Chapter 2, new solution strategy of the flow with intermediate storage using temporal repetition of flow is presented to solve the maximum as well as earliest arrival flow problems. To justify the result in practical application, we have presented case illustration of maximum dynamic flow with intermediate storage by taking a part of road network of Kathmandu using Python programming language. Similarly, contraflow is another important and commonly used technique which improves the flow transmission by the reversal of unused oppositely directed arcs towards the destination. Contraflow with symmetric transit times and asymmetric contraflow with orient dependent transit times can be found in literature. A different aspect of the asymmetric contraflow problem in which transit time of an arc remains the same whatever be its orientation is introduced by using anti-parallel path decomposition. The maximum and quickest contraflow with anti-parallel path decomposition are solved in polynomial time complexity whose case illustrations using Python are presented by taking real road network of a part of Kathmandu.

Congestion elimination at the crossing of paths plays a vital role to smooth the flow, which can be possible in abstract network by switching of the paths. Integrating the abstract network flow and flow with intermediate storage, we introduced the flow models and solution strategies

for maximum static, lexicographic maximum static and maximum dynamic flow problems and presented the polynomial time solution strategies (cf. Chapter 3). It helps to smooth the flow on non-crossing paths by holding the excess flow at intermediate shelters. Similarly, we have proposed the abstract flow model with partial switching and its solution strategy to improve the flow transmission at the destination.

As a major concern of the study, we have incorporated the flow with intermediate storage in multi-commodity flow problems, where flow of different commodities are transshipped from respective sources to their corresponding sinks. The polynomial time algorithm for static MCF and pseudo-polynomial time algorithm for dynamic MCF with intermediate storage are presented in Chapter 4. At the time of disaster, every individual may not hurt equally. We have presented a priority based multi-commodity evacuation planning problem whose solution is obtained by using TRF in polynomial time approximation and using time expanded network in pseudo-polynomial time. The sharing of capacity in bundle arc is one of the challenging issue in multi-commodity flow problem. We have proposed a proportional capacity sharing technique which reduces the multi-commodity flow problem to commodity-wise single commodity flow problems. Using this sharing, polynomial time approximation to the maximum and quickest multi-commodity flow problems are possible. Similarly, we have presented pseudo-polynomial time solutions of maximum and quickest MCF problems by defining flow-dependent capacity sharing. By incorporating the partial contraflow in two-way network with flow-dependent transit time on arcs, we have presented a polynomial time approximation by using length bound approach and a fully polynomial time approximation by using Δ -condensed time expanded network.

Finally, our concern was to provide some supports to the evacuees through the allocation of facilities at appropriate locations. To address this problem with mathematical model, we have designed a bi-level facility allocation model in which lower level problem solves maximum flow problem and the optimal location for the placement of the facilities is assured by the upper level problem (cf. Chapter 5).

Problems for Further Research. We have solved the maximum and quickest multi-commodity flow problems in general network topology. However, solving these problems in abstract network by switching of paths is the problem for further research and can be a hard problem as the switched path may violate the commodity-wise source-sink flow. Similarly, declaration of the number of arcs to be reversed for symmetric as well as asymmetric contraflow configuration, instead of reversing all arcs at time zero, for the best solution can also be another problem for further research and can be a hard problem as it is combinatorial problem.

References

- R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, Englewood Cliffs, (1993). Link: https://www.dl.behinehyab.com/Ebooks/NETWORK/NET005_354338_www.behinehyab.com.pdf.
- A. Ali, R. Helgason, J. Kennington and H. Lall, Computational comparison among three multi-commodity network flow algorithms, *Operations Research*, 28(4), (1980), 995-1000. Link: <https://www.jstor.org/stable/170337>.
- G. Anandalingam, T. Friesz, Hierarchical optimization: an introduction, *Annals of Operations Research*, 34(1), (1992), 1–11. DOI: <https://doi.org/10.1007/BF02098169>.
- J. E. Aronson, A survey of dynamic network flows, *Annals of Operations Research*, 20, (1989), 1–66. DOI: <https://doi.org/10.1007/BF02216922>.
- A. Arulsevan, Network model for disaster management, Ph.D. thesis. University of Florida Gainesville, (2009).
- A. Assad, Multi-commodity network flows a survey, *Networks*, 8(1), (1978), 37-91. DOI: <https://doi.org/10.1002/net.3230080107>.
- J. Bard, Practical bilevel optimization, Red. by P. Pardalos and R. Horst. Vol. 30. *Nonconvex Optimization and Its Applications*, Boston, MA: Springer US, (1998). DOI: 10.1007/978-1-4757-2836-1.
- N. Baumann, E. M. Köhler, Approximating earliest arrival flows with flow-dependent transit times, *Discrete Applied Math*, 155, (2007), 161-171. DOI: <https://doi.org/10.1016/j.dam.2006.04.030>.
- Ben-Ayed, O. Bilevel linear programming, *Computers & Operations Research*, 20(5), (1993), 485–501. DOI: 10.1016/0305-0548(93)90013-9.
- P. P. Bhandari, S. R. Khadka, Evacuation contraflow problems with not necessarily equal transit time on anti-parallel arcs, *American Journal of Applied Mathematics*, 8, (2020), 230–235. DOI: 10.11648/j.ajam.20200804.18.
- J. Bracken, J. McGill (1973), *Mathematical Programs with Optimization Problems in the Constraints*, In: *Operations Research*, 21(1), 37–44. DOI: 10.1287/opre.21.1.37.
- R. E. Burkard, K. Dlaska, B. Klinz, The quickest flow problem, *Zeitschrift für Operations Research*, 37(1), (1993), 31-58. DOI: <https://doi.org/10.1007/BF01415527>.
- J. Burtscheidt, M. Claus, S. Dempe, Risk-averse models in bilevel stochastic lin-

- ear programming, *SIAM Journal on Optimization*, 30(1), (2020), 377-406. DOI: <https://doi.org/10.1137/19M1242240>
- M. Carey, A constraint qualification for a dynamic traffic assignment model, *Transportation Science*, 20, (1986), 55-58. DOI: <https://doi.org/10.1287/trsc.20.1.55>
- Y. L. Chen, Y. H. Chin, The quickest path problem, *Computers and Operation Research*, 17, (1990), 153-161. DOI: [https://doi.org/10.1016/0305-0548\(90\)90039-A](https://doi.org/10.1016/0305-0548(90)90039-A).
- B. Colson, P. Marcotte, G. Savard, Bilevel programming: a survey, *4OR*, 3(2), (2005), 87–107. DOI: 10.1007/s10288-005-0071-0.
- B. Colson, P. Marcotte, G. Savard, An overview of bilevel optimization, *Annals of Operations Research*, 153(1), (2007), 235–256. DOI: 10.1007/s10479-007-0176-2.
- M. Daskin, Network and discrete location: models, algorithms and applications, *Journal of the Operational Research Society*, 48(5), (1997), 763–764. DOI: <https://doi.org/10.1057/palgrave.jors.2600828>
- S. Dempe, *Foundations of bilevel programming*, Springer Science & Business Media, (2002). Link: <https://link.springer.com/book/10.1007/b101970>.
- S. Dempe, Computing locally optimal solutions of the bilevel optimization problem using the KKT approach, In *International conference on mathematical optimization theory and operations research*, (2019), 147–157. DOI: DOI:10.1007/978-3-030-22629-9_11.
- S. Dempe, Bilevel optimization: theory, algorithms, applications and a bibliography, In: *Bilevel Optimization: Advances and Next Challenges*. Ed. by S. Dempe and A. Zemkoho. Springer Optimization and Its Applications. Cham: Springer International Publishing, (2020), 581–672. DOI: 10.1007/978-3-030-52119-6_20.
- S. Dempe, A. Zemkoho, eds. *Bilevel optimization: advances and next challenges*, Vol. 161, Springer Optimization and Its Applications, Cham: Springer International Publishing, (2020). DOI: 10.1007/978-3-030-52119-6.
- T. N. Dhamala, U. Pyakurel, S. Dempe, A critical survey on the network optimization algorithms for evacuation planning problems, *International Journal of Operations Research*, 15, (2018), 101–133. DOI: 10.6886/IJOR.201809_15(3).0002.
- T. N. Dhamala, S. P. Gupta, D. P. Khanal, U. Pyakurel, Quickest multi-commodity flow over time with partial lane reversals, *Journal of Mathematics and statistics*, 16, (2020), 198-211. DOI: <https://doi.org/10.3844/jmssp.2020.198.211>.
- T. N. Dhamala, M. C. Adhikari, D. P. Khanal, U. Pyakurel, Generalized maximum flow over time with intermediate storage, *Annals of Operations Research*, (2024). DOI: <https://doi.org/10.1007/s10479-023-05773-w>
- R. C. Dhungana, T. N. Dhamala, Maximum FlowLoc problems with network reconfiguration, *International Journals of Operation Research*, 16(1), (2019), 13-26. DOI: [https://doi.org/10.6886/IJOR.201903_16\(1\).0002](https://doi.org/10.6886/IJOR.201903_16(1).0002).
- E. W. Dijkstra, A note on two problems in connexion with graphs. *Numerische mathematik*, 1, (1959), 269–271. DOI: <https://doi.org/10.1007/BF01386390>

- E. A. Dinic, Algorithm for solution of a problem of maximum flow in a network with power estimation, *Soviet Mathematics - Doklady*. *Doklady*, 11, (1970), 1277–1280. English translation by RF. Rinehart.
- J. Edmonds, R. M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the ACM*, 19(2), (1972), 248-264. DOI: <https://doi.org/10.1145/321694.321699>.
- L. Fleischer, Universally maximum flows with piecewise constant capacities, *Networks*, 38(3), (2001), 115-125. DOI: <https://doi.org/10.1002/net.1030>.
- L. Fleischer, É. Tardos, Efficient continuous-time dynamic network flow algorithms, *Operations Research Letters*, 23, (1998), 71–80. DOI: [https://doi.org/10.1016/S0167-6377\(98\)00037-6](https://doi.org/10.1016/S0167-6377(98)00037-6).
- L. R. Ford, D. R. Fulkerson, Maximal flow through a network, *Canadian journal of Mathematics* 8, (1956), 399–404. DOI: <https://doi.org/10.4153/CJM-1956-045-5>.
- L. R. Ford, D. R. Fulkerson, *Flows in networks*, Princeton university press, (1962). Link: <https://www.rand.org/content/dam/rand/pubs/reports/2007/R375.pdf>.
- J. Fortuny-Amat, B. McCarl, A representation and economic interpretation of a two-level programming problem, In: *The Journal of the Operational Research Society*, 32(9), (1981), 783–792. DOI: 10.1057/jors.1981.156.
- D. Gale, Transient flows in networks, *Michigan Mathematical Journal*, 6(1), (1959), 59–63. DOI: <https://doi.org/10.1307/MMJ%2F1028998140>.
- E. Gerasimenko, V. Kureichik, E. Kuliev, Maximum Dynamic Flow Model for Hesitant Fuzzy Evacuation with Intermediate Storage at Nodes, In: Kahraman C., Cebi S., Cevik Onar S., Oztaysi B., Tolga A. C., Sari I. U. (eds) *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation. INFUS 2021. Lecture Notes in Networks and Systems*, vol 307, (2022), Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-85626-7_81.
- A. V. Goldberg, R. E. Tarjan, A new approach to the maximum-flow problem, *Journal of the Association for Computing Machinery*, 35(4), (1988), 921-940. DOI: <https://doi.org/10.1145/48014.61051>.
- A. V. Goldberg, R. E. Tarjan, Finding minimum-cost circulations by successive approximation, *Mathematics of Operations Research*, 15(3), (1990), 430-466. Link: <https://www.jstor.org/stable/3689990>.
- A. Hall, S. Hippler and M. Skutella, Multi-commodity flows over time: Efficient algorithms and complexity, *Theoretical Computer Science*, 379, (2007), 387-404. DOI: <https://doi.org/10.1016/j.tcs.2007.02.046>.
- A. Hall, K. Langkau, M. Skutella, An FPTAS for quickest multi-commodity flows with inflow-dependent transit times, *Algorithmica*, 47(3), (2007a), 299-321. DOI: <https://doi.org/10.14279/depositonce-14268>.
- G. Handler, I. Zang, A dual algorithm for the constrained shortest path problem, *Networks*, 10,

- (1980), 293-309. DOI: DOI: 10.1002/net.3230100403.
- H. W. Hamacher, S. Haller, B. Rupp, Flow location (FlowLoc) problems: dynamic network flows and location models for evacuation planning. *Annals of Operation Research*, 207, (2013), 161-180. DOI: <https://doi.org/10.1007/s10479-011-0953-9>.
- H. Hamacher, S. Tjandra, Mathematical modeling of evacuation problems: a state of the art, In M. Schreckenberger & S.D. Sharma (Eds.), *Pedestrian and Evacuation Dynamics*, (pp 227–266), (2002). Link: <https://d-nb.info/1027386792/34>.
- A. Hoffman, A generalization of max flow - min cut, *Mathematical Programming*, 6, (1974), 352–359. DOI: <https://doi.org/10.1007/BF01580250>.
- B. Hoppe, É. Tardos, Polynomial time algorithms for some evacuation problems, In: *Fifth Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'94)*, (1994), 433–441. Link: <https://dl.acm.org/doi/10.5555/314464.314583>.
- B. Hoppe, E. Tardos, The quickest transshipment problem, *Mathematics of Operations Research*, 25, (2000), 36-62. DOI: <https://doi.org/10.1287/moor.25.1.36.15211>.
- J. P. Kappmeier, Generalizations of flows over time with applications in evacuation optimization, Ph.D. Thesis, Technical University, Berlin, Germany, (2015).
- J. P.W. Kappmeier, J. Matuschke, B. Peis, Abstract flows over time: A first step towards solving dynamic packing problems, *Theoretical Computer Science*, 544, (2014), 74–83. DOI: <https://doi.org/10.1016/j.tcs.2014.04.012>.
- H. Kellerer, R. Mansini, U. Pferschy, M. G. Speranza, An efficient fully polynomial approximation scheme for the subset-sum problem, *Journal of Computer and System Sciences*, 66(2), (2003), 349–370. DIO: [https://doi.org/10.1016/S0022-0000\(03\)00006-0](https://doi.org/10.1016/S0022-0000(03)00006-0).
- J. Kennington, A survey of linear cost multi-commodity network flows, *Operations Research*, 26(2), (1978), 209-236. DOI: <https://doi.org/10.1287/opre.26.2.209>.
- D. P. Khanal, U. Pyakurel, T. N. Dhamala, Prioritized multi-commodity flow model and algorithm, Presented on: *International Symposium on Analytic Hierarchy Process 2020 (ISAHP 2020)*. DOI: <https://doi.org/10.13033/isahp.y2020.049>.
- D. P. Khanal, U. Pyakurel, T. N. Dhamala, Maximum multi-commodity flow with intermediate storage, *Mathematical Problems in Engineering*, (2021). DOI: <https://doi.org/10.1155/2021/5063207>.
- D. P. Khanal, U. Pyakurel, T. N. Dhamala, S. Dempe, Efficient algorithms for abstract flow with partial switching, *Operations Research Forum*, Springer Nature, (2022) 3:55(1-17). DOI: <https://doi.org/10.1007/s43069-022-00168-2>.
- D. P. Khanal, U. Pyakurel, T. N. Dhamala, S. Dempe, Maximum Multi-Commodity Flow with Proportional and Flow-dependent Capacity Sharing, *Comput. Sci. Math. Forum*, (2022), 2, 5. DOI: <https://doi.org/10.3390/IOCA2021-10904>.
- D. P. Khanal, U. Pyakurel, S. Dempe, Dynamic contraflow with orientation dependent transit times allowing intermediate storage, *The Nepali Mathematical Sciences Report*, 38(2), (2021a), 1-12. DOI: <http://doi.org/10.3126/nmsr.v38i2.42700>.

- D. P. Khanal, U. Pyakurel, S. Dempe, Temporally Repeated Maximum Dynamic Flow with Intermediate Storage, Preprint 2022-01, Fakultät für Mathematik und Informatik, TU Bergakademie Freiberg, Germany.
- S. Kim, S. Shekhar, Contraflow network reconfiguration for evacuation planning: a summary of results, In: Proceedings of 13th ACM Symposium on Advances in Geographic Information Systems GIS, vol. 05, (2005), 250–259. DOI: <https://doi.org/10.1145/1097064.1097099>.
- S. Kim, S. Shekhar, M. Min, Contraflow transportation network reconfiguration for evacuation route planning, IEEE transactions on Knowledge and Data Engineering, 20, (2008), 1115–1129. DOI: <https://doi.org/10.1109/TKDE.2007.190722>.
- T. Kleinert, M. Schmidt, M., Why there is no need to use a big- in linear bilevel optimization: A computational study of two ready-to-use approaches, Technical Report (2020). URL: http://www.optimization-online.org/DB_HTML/2020/10/8065.html.
- E. Köhler, K. Langkau, M. Skutella, Time- expanded graphs for flow-dependent transit times, In Proceedings of the 10th Annual European Symposium on Algorithms (ESA), 599–611, Volume 2461 of Lecture Notes in Computer Science, Springer, Berlin, (2002). DOI: https://doi.org/10.1007/3-540-45749-6_53.
- E. Köhler, M. Skutella, Flows over time with load-dependent transit times, SIAM Journal of Optimization, 15, (2005), 1185–1200. DOI: <https://doi.org/10.1137/S105262340343264>.
- B. Kotnyek, An annotated overview of dynamic network flows, Rapport de recherche 4936, INRIA Sophia Antipolis, (2003). Link: <https://inria.hal.science/inria-00071643/document>.
- I. S. Kotsireas, A. Nagurney, P. M. Pardalos, Dynamics of disasters-algorithmic approaches and applications, Springer Optimization and Its Applications, (2018). Link: <https://link.springer.com/book/10.1007/978-3-319-97442-2>.
- M. Lin, P. Jaillet, On the quickest flow problem in dynamic networks: A parametric min-cost flow approach, In: Proceeding of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia: Society for Industrial and Applied Mathematics, (2015), 1343–1356. DOI: <https://doi.org/10.1137/1.9781611973730.89>.
- M. Martens, Path-Constrained Network Flows, Ph.D. Thesis, Technical University, Berlin, Germany, (2007).
- M. Martens, S. T. McCormick, A polynomial algorithm for weighted abstract flow, In International Conference on Integer Programming and Combinatorial Optimization, Springer, (2008), 97–111. Link: https://link.springer.com/chapter/10.1007/978-3-540-68891-4_7.
- S. T. McCormick, A polynomial algorithm for abstract maximum flow, In SODA, Citeseer, (1996), 490–497. Link: <https://dl.acm.org/doi/10.5555/313852.314108>.
- N. Megiddo, Combinatorial optimization with rational objective functions, Mathematics of Operations Research Programming, 4, (1979), 414–424. DOI: <https://doi.org/10.1287/moor.4.4.414>.
- D. K. Merchant, G. L. Nemhauser, A model and an algorithm for dynamic traf-

- fic assignment problems, *Transportation Science*, 12, (1978), 183-199. DOI: <https://doi.org/10.1287/trsc.12.3.183>.
- E. Minieka, Maximal, lexicographic, and dynamic network flows, *Operations Research*, 21, (1973), 517–527. DOI: <https://doi.org/10.1287/opre.21.2.517>.
- H. N. Nath, U. Pyakurel, T. N. Dhamala, Network reconfiguration with orientation-dependent transit times, *International Journal of Mathematics and Mathematical Sciences*, (2021), Article ID 6613622, 11 pages, DOI: <https://doi.org/10.1155/2021/6613622>.
- H. N. Nath, U. Pyakurel, T. N. Dhamala, S., Dynamic network flow location models and algorithms for evacuation planning, *Journals of Industrial and Management Optimization*, (2020), DOI: <http://doi.org/10.3934/jimo.2020102>.
- S. Nayak, *Fundamentals of optimization techniques with algorithms*, Elsevier Science, (2020). DOI: <http://doi.org/10.1016/b978-0-12-821126-7.00009-7>.
- R. G. Ogier, Minimum delay routing in continuous-time dynamic networks with piecewise constant capacities, *Networks*, 18, (1988), 303-318. DOI: <https://doi.org/10.1002/net.3230180405>.
- S. Pineda, J. M. Morales, Solving linear bilevel problems using big- M s: Not all that glitters is gold, In: *IEEE Transactions on Power Systems*, (2019). DOI: <https://doi.org/10.1109/TPWRS.2019.2892607>.
- U. Pyakurel, S. Dempe, Network flow with intermediate storage: models and algorithms, *SN Operations Research Forum*, 1, (2020), 1–23. DOI: DOI <https://doi.org/10.1007/s43069-020-00033-0>.
- U. Pyakurel, S. Dempe, Universal Maximum Flow with Intermediate Storage for Evacuation Planning, In I. S. Kotsireas, A. Nagurney, P. M. Pardalos, & A. Tsokas A. (Eds.), *Dynamics of Disasters*, Springer Optimization and Its Applications, vol 169 (2021) . Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-64973-9_14.
- U. Pyakurel, T. N. Dhamala, Continuous time dynamic contraflow models and algorithms, *Advances in Operations Research*, vol. 2016, Article ID 368587, (2016), 7 pages. DOI: <https://doi.org/10.1155/2016/7902460>.
- U. Pyakurel, T. N. Dhamala, S. Dempe, Efficient continuous contraflow algorithms for evacuation planning problems, *Annals of Operations Research*, 254 (2017), 335–364. DOI: <https://doi.org/10.1007/s10479-017-2427-1>.
- U. Pyakurel, S. P. Gupta, D. P. Khanal, T. N. Dhamala, Efficient algorithms on multi-commodity flow over time problems with partial lane reversals, *International Journal of Mathematics and Mathematical Sciences*, Hindawi, (2020), DOI: <https://doi.org/10.1155/2020/2676378>.
- U. Pyakurel, D. P. Khanal, T. N. Dhamala, Abstract network flow with intermediate storage for evacuation planning, *European Journal of Operational Research*, 305(3):1178-1193. DOI: <https://doi.org/10.1016/j.ejor.2022.06.054>.
- U. Pyakurel, H. N. Nath, S. Dempe, T. N. Dhamala, Efficient dynamic flow algorithms for

- evacuation planning problems with partial lane reversal, *Mathematics*, 7, (2019), 1–29. DOI: <https://doi.org/10.3390/math7100993>.
- U. Pyakurel, H. N. Nath, T. N. Dhamala, Efficient contraflow algorithms for quickest evacuation planning, *Science China Mathematics*, 61, (2018), 2079–2100. DOI: <https://doi.org/10.1007/s11425-017-9264-3>.
- S. Rebennack, A. Arulseivan, L. Elefteriadou, P. M. Pardalos, Complexity analysis for maximum flow problems with arc reversals, *Journal of Combinatorial Optimization*, 19, (2010), 200–216. DOI: <https://doi.org/10.1007/s10878-008-9175-8>.
- J. B. Rosen, S. - Z. Sun, G. - L. Xue, Algorithms for the quickest path problem and the enumeration of quickest Paths, *Computers and Operations Research*, 18, (1991), 579–584. DOI: [https://doi.org/10.1016/0305-0548\(91\)90063-W](https://doi.org/10.1016/0305-0548(91)90063-W).
- S. Ruzika, H. Sperber, M. Steiner, Earliest arrival flows on series-parallel graphs, *Networks*, 10, (2011), 169–173. DOI: <https://doi.org/10.1002/net.20398>.
- S. Scholtes, Convergence properties of a regularization scheme for mathematical programs with complementarity constraints, *SIAM Journal on Optimization*, 11, (2001), 918–936. DOI: <https://doi.org/10.1137/S1052623499361233>.
- M. Skutella, An introduction to network flows over time, In *Research trends in combinatorial optimization*, Springer, (2009) 451–482. Link: https://link.springer.com/chapter/10.1007/978-3-540-76796-1_21.
- Y. Sheffi, *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Techniques*, Prentice-Hall, Inc.: Englewood Cliffs, NJ, USA, (1984).
- S. A. Tjandra, Dynamic network optimization with application to the evacuation problem. PhD thesis, University of Kaiserslautern, Germany, (2003).
- L. Vicente, P. Calamai, Bilevel and Multilevel Programming: A Bibliography Review, *Journal of Global Optimization*, 5(3), (1994), 291–306. DOI: 10.1007/BF01096458.
- H. von Stackelberg, *Marktform und gleichgewicht*, Springer, Wien, (1934). English Translation: *The theory of the market economy*, Oxford University, Oxford, (1952). Link: <https://www.abebooks.com/Theory-Market-Economy-Stackelberg-Translated-German/12012907202/bd>.
- A. Weber, *Theory of the location industries, Über den Standort der Industrien*, University Chicago Press (1909), English translation by C.J. Friedrich (1929).
- U.-P. Wen, S.-T. Hsu, Linear bi-Level programming problems - a review, *Journal of the Operational Research Society*, 42(2), (1991), 125–133. DOI: <https://doi.org/10.1057/jors.1991.23>.
- W. L. Wilkinson, An algorithm for universal maximal dynamic flows in a network, *Operations Research*, 19, (1971), 1602–1612. DOI: <https://doi.org/10.1287/opre.19.7.1602>.

Appendix A

Data for Case Illustration I

For the purpose of case illustration, we consider a part of Kathmandu and Lalitpur Metropolitan Cities surrounded by ring road as an evacuation zone and the image with road networks is taken from Google. The number of lanes and vehicle traversal time (in minutes) for each arc/path segment of the network are presented in Table A.1 herein

Table A.1: Arcs/road segments with respective no. of lanes and vehicle transit times.

S.N.	Arc	No. of lanes	Tr. Time (min.)	S.N.	Arc	No. of lanes	Tr. Time (min.)
1	0-1	6	4	24	10-11	6	2
2	0-2	6	2	25	10-43	6	0.5
3	0-3	2+1=3	1	26	10-46	1	3
4	0-5	4	1.5	27	11-12	2	1
5	0-6	2	2	28	11-13	6	2
6	1-7	1	3	29	11-45	1	1.5
7	1-9	6	1	30	11-47	2	1
8	2-3	2	2	31	11-48	1	1.5
9	2-10	6	1.5	32	12-13	2	0.5
10	3-4	2	2	33	16-22	2	0.5
11	3-11	2+1=3	2.5	34	13-24	6	1
12	4-15	2	0.5	35	14-13	2	1
13	4-16	1	1	36	14-20	2	0.5
14	5-4	2	0.5	37	15-14	2	0.5
15	5-17	4	0.5	38	15-21	2	1
16	6-5	2	0.5	39	16-14	2	0.5
17	6-7	2	0.5	40	17-15	2	0.5
18	6-18	2	0.5	41	17-22	4	0.5
19	7-19	1	3	42	18-17	2	0.5
20	8-7	2	2.5	43	18-23	2	0.5
21	8-28	4	3	44	19-18	2	0.5
22	9-8	4	1.5	45	19-28	2	1
23	9-30	6	3	46	20-13	2	1

S.N.	Arc	No. of lanes	Tr. Time (min.)	S.N.	Arc	No. of lanes	Tr. Time (min.)
47	20-25	2	1	92	40-41	2	0.5
48	20-35	2	1	93	41-42	6	2
49	20-36	1	1	94	42-68	8	2
50	21-20	4	0.5	95	43-44	8	0.5
51	21-26	2	0.5	96	44-46	1	0.5
52	22-21	4	0.5	97	44-50	8	2
53	22-26	4	0.5	98	45-46	2	0.5
54	23-22	2	0.5	99	46-44	2	1
55	23-27	2	0.5	100	46-50	2	1
56	24-48	1	1	101	47-45	2	0.5
57	24-49	2	1	102	47-48	1	0.5
58	24-63	2	1	103	48-45	2	2
59	24-64	2	2	104	48-49	1	0.5
60	24-66	1	3.5	105	49-55	2+1=3	1
61	25-24	4	0.5	106	49-56	2	1
62	25-36	4	0.5	107	49-63	1	2
63	26-20	4	0.5	108	50-51	8	0.5
64	26-34	2	0.5	109	50-54	2	0.5
65	26-35	2	1	110	50-55	2	1
66	27-26	4	1	111	51-52	8	0.5
67	27-29	2	0.5	112	52-53	8	0.5
68	28-27	4	0.5	113	53-58	8	1
69	28-29	2	0.5	114	54-52	2	1
70	29-31	4	1.5	115	54-53	2	1
71	29-34	4	1.5	116	54-57	2	1
72	30-29	4	2.5	117	55-56	2	1
73	30-31	6	2.5	118	56-57	2	0.5
74	31-32	6	3	119	56-59	2+2=4	0.5
75	31-33	2	2	120	57-58	2	1
76	32-41	6	1	121	58-60	8	1.5
77	33-32	4	1	122	59-60	1	1
78	33-40	2	1	123	59-61	2	1
79	34-33	2	2	124	59-62	2	1
80	34-35	2	0.5	125	60-61	8	0.5
81	35-37	2	0.5	126	60-68	1	2
82	35-39	2	0.5	127	61-62	8	0.5
83	36-37	4	0.5	128	62-66	8	0.5
84	37-38	4	0.5	129	63-62	2	1.5
85	37-64	1	0.5	130	63-66	1	0.5
86	37-65	1	1	131	64-65	2	0.5
87	38-42	4	2	132	65-66	2	1
88	38-65	4	1	133	65-67	4	1
89	39-38	2	0.5	134	66-67	8	0.5
90	39-42	2	2.5	135	67-68	8	0.5
91	40-39	2	1.5				

Appendix B

Data for Case Illustration II

For the purpose of case illustration, we consider a part of Kathmandu Metropolitan City as an evacuation zone and the image with road networks is taken from Google. The capacity and vehicle traversal time (in minutes) for each arc/path segment of the network are presented in Table B.1 herein.

Table B.1: Arcs/road segments with respective capacities and vehicle transit times.

S.N.	Arc	Capacity	Tr. Time
1	0-1	2	1.5
2	1-0	2	1
3	0-2	2	2
4	2-1	2	1
5	0-4	2	0.5
6	0-7	1	0.2
7	7-0	1	0.2
8	0-8	1	0.5
9	8-0	1	0.5
10	0-10	1	1
11	10-0	1	1
12	0-20	2	1.5
13	4-5	2	1.5
14	4-7	1	0.5
15	7-4	1	0.5
16	7-6	2	1.5
17	6-7	3	1
18	7-9	1	0.5
19	9-7	1	0.5
20	8-9	1	0.2
21	9-8	1	0.2
22	8-10	2	0.2
23	34-36	2	0.5

S.N.	Arc	Capacity	Tr. Time
24	9-11	1	0.2
25	11-9	1	0.2
26	12-11	2	0.5
27	11-10	3	0.5
28	10-13	2	0.5
29	13-14	2	0.5
30	11-13	1	0.5
31	13-11	1	0.5
32	20-21	2	1.5
33	21-22	2	0.5
34	22-21	2	0.5
35	22-19	2	1.5
36	19-10	2	0.5
37	10-23	2	1.5
38	24-17	2	0.5
39	17-10	2	1.5
40	16-17	1	1
41	17-16	1	1
42	22-23	2	1
43	23-22	2	1
44	23-24	2	0.5
45	24-23	2	0.5
46	24-18	2	1

S.N.	Arc	Capacity	Tr. Time
47	18-24	2	1
48	24-47	2	1
49	47-24	2	1
50	2-3	4	0.5
51	3-2	4	0.5
52	3-5	4	0.2
53	5-3	4	0.2
54	6-5	4	0.5
55	5-6	4	0.5
56	6-12	4	0.5
57	12-6	4	0.5
58	12-14	4	0.5
59	14-12	4	0.5
60	14-15	4	0.5
61	15-14	4	0.5
62	15-16	4	0.2
63	16-15	4	0.2
64	16-18	4	0.5
65	18-16	4	0.5
66	3-25	2	1
67	25-2	2	1
68	25-27	2	0.5
69	27-28	2	1
70	26-25	2	1.5
71	28-26	2	0.5
72	26-28	2	0.5
73	30-28	2	0.2
74	28-30	2	0.2
75	27-29	2	0.5
76	29-27	2	0.5
77	31-5	2	1
78	6-32	2	0.5
79	32-6	2	0.5
80	31-29	2	0.5
81	29-31	2	0.5
82	29-30	2	1
83	30-29	2	1
84	32-33	1	1
85	33-32	1	1
86	33-35	1	0.2
87	35-33	1	0.2
88	35-18	2	1

S.N.	Arc	Capacity	Tr. Time
89	18-35	2	1
90	33-14	2	0.5
91	15-35	2	0.5
92	29-34	2	1.5
93	33-34	2	0.2
94	35-36	2	0.5
95	36-35	2	0.5
96	36-37	2	0.2
97	37-36	2	0.2
98	37-42	2	1
99	42-37	2	1
100	42-41	2	0.2
101	41-42	2	0.2
102	41-45	3	1
103	46-40	3	1
104	30-38	2	1.5
105	38-30	2	1.5
106	38-39	2	1
107	39-38	2	1
108	39-49	2	1
109	49-39	2	1
110	18-44	4	1
111	44-43	4	0.5
112	43-18	4	1
113	47-48	2	1
114	48-47	2	1
115	48-44	3	0.5
116	44-48	3	0.5
117	48-49	6	1
118	49-46	6	0.5
119	46-45	6	0.2
120	45-44	6	0.5
121	43-42	2	0.5
122	42-43	2	0.5
123	41-40	2	0.2
124	40-41	2	0.2
125	39-40	2	0.5
126	40-39	2	0.5
127	20-19	2	0.2
128	32-31	2	0.2
129	31-32	2	0.2
130	37-38	1	1
131	38-37	1	1

Appendix C

List of Publications

- Dhamala, T. N., Gupta, S. P., Khanal, D. P. and Pyakurel, U., Quickest Multi-commodity Flow Over Time with Partial Lane Reversals, *Journal of Mathematics and Statistics*, 16:198-211, (2020). DOI: <https://doi.org/10.3844/jmssp.2020.198.211>
- Pyakurel, U., Gupta, S. P., Khanal, D. P. and Dhamala, T. N., Efficient Algorithms on Multicommodity Flow over Time Problems with Partial Lane Reversals, *International Journal of Mathematics and Mathematical Sciences, Hindawi*, (2020). DOI: <https://doi.org/10.1155/2020/2676378>.
- Gupta, S. P., Khanal, D. P., Pyakurel, U. and Dhamala, T. N., Approximate Algorithms for Continuous-Time Quickest Multi-Commodity Contraflow Problem, *The Nepali Mathematical Sciences Report*, 37:30-46, (2020). DOI: <https://doi.org/10.3126/nmsr.v37i1-2.34068>.
- Khanal, D. P., Pyakurel, U. and Dhamala, T. N., Prioritized Multi-Commodity Flow Model and Algorithm, *International Symposium on Analytic Hierarchy Process (ISAHP 2020)*. DOI: <https://doi.org/10.13033/isahp.y2020.049>.
- Khanal, D. P., Pyakurel, U. and Dhamala, T. N., Maximum Multicommodity Flow with Intermediate Storage, *Mathematical Problems in Engineering, Hindawi* (2021). DOI: <https://doi.org/10.1155/2021/5063207>.
- Khanal, D. P., Pyakurel, U. and Dempe, S., Dynamic Contraflow with Orientation Dependent Transit Times Allowing Intermediate Storage, *The Nepali Mathematical Sciences Report*, 38(2):1-12, (2021). DOI: <http://doi.org/10.3126/nmsr.v38i2.42700>.
- Khanal, D. P., Pyakurel, U., Dhamala, T. N. and Dempe S., Maximum Multi-Commodity Flow with Proportional and Flow-dependent Capacity Sharing, *Comput. Sci. Math. Forum* 2022, 2, 5. DOI: <https://doi.org/10.3390/IOCA2021-10904>.
- Khanal, D. P., Pyakurel, U., Dhamala, T. N. and Dempe S., Efficient Algorithms for Ab-

stract Flow with Partial Switching, *Operations Research Forum, Springer Nature* 3:55(1-17) (2022) . DOI: <https://doi.org/10.1007/s43069-022-00168-2>.

- Khanal, D. P., Pyakurel, U., Dhamala, T. N. and Dempe, S., Abstract Temporally Repeated Flow with Intermediate Storage, *The Nepali Mathematical Sciences Report*, 39(2):1-12, (2022). DOI: <https://doi.org/10.3126/nmsr.v39i2.51695>.
- Pyakurel U., Khanal, D. P. and Dhamala, T. N., Abstract Network Flow with Intermediate Storage for Evacuation Planning, *European Journal of Operational Research*, 305(3):1178-1193, (2023). DOI: <https://doi.org/10.1016/j.ejor.2022.06.054>.

Research Article

Maximum Multicommodity Flow with Intermediate Storage

Durga Prasad Khanal,¹ Urmila Pyakurel ,² and Tanka Nath Dhamala ²

¹Saraswati Multiple Campus, Tribhuvan University, Kathmandu, Nepal

²Central Department of Mathematics, Tribhuvan University, P.O. Box 13143, Kathmandu, Nepal

Correspondence should be addressed to Urmila Pyakurel; urmilapyakurel@gmail.com

Received 6 April 2021; Revised 5 June 2021; Accepted 23 June 2021; Published 21 July 2021

Academic Editor: Ioannis T. Christou

Copyright © 2021 Durga Prasad Khanal et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The multicommodity flow problem deals with the transshipment of more than one commodity from respective sources to corresponding sinks without violating the capacity constraints. Due to the capacity constraints, flows out from the sources may not reach their sinks, and so, the storage of excess flows at intermediate nodes plays an important role in the maximization of flow values. In this paper, we introduce the maximum static as well as maximum dynamic multicommodity flow problems with intermediate storage. We present polynomial and pseudopolynomial time algorithms for the former and latter problems, respectively. We also present the solution procedures to these problems in contraflow network having symmetric as well as asymmetric arc transit times. We transform the solutions in continuous-time settings by using natural transformation.

1. Introduction

Network is a topological structure with links (arcs), with its crossings (nodes) being its components. The transportation network is one of the relevant examples of a network topology, in which road segments are considered as the arcs and their crossings as nodes. Any kinds of entities moving on the road are considered as flows, and their initial and final destinations are considered as the source and sink, respectively. Each arc has nonnegative capacity, which limits the flow on arc. Dynamic network has one more attribute on arc, i.e., transit time, which represents the time to send the flow from one node to another one. Ford and Fulkerson are the pioneers of the network flow over time (so-called dynamic flow) problems [1, 2].

The transshipment of several different commodities from respective sources to corresponding sinks through a network without violating the capacity constraints on the arcs is known as multicommodity flow problem. Vehicle routine in transportation, production planning, supply chains for essential goods, and message routing in telecommunication are some examples of multicommodity flow problem. On the basis of temporal dimension, multicommodity flow problem can be classified as static

multicommodity flow problem and dynamic multicommodity flow problem [3–6]. If we maximize the supply-demand in a fixed time horizon, then the problem becomes a maximum dynamic multicommodity flow problem. The static multicommodity flow problem is polynomial time solvable by using the ellipsoid or interior point method, whereas dynamic multicommodity flow problem is \mathcal{NP} -hard [7]. By using time expanded network, Kappmeier [8] provided the solution of maximum dynamic multicommodity flow problem and multisource single sink multicommodity earliest arrival transshipment problem in pseudopolynomial time complexity. Priority-based multicommodity flow problem and polynomial time solution strategy are presented in [9].

Maximum flow problem with intermediate storage is extremely relevant in large scale disaster management. In evacuation models, one wishes to shift maximum evacuees from danger zones (sources) to safety places (sinks) as quickly and efficiently as possible. Thus, at the time of evacuation, if the number of evacuees out from sources is greater than the minimum cut capacity, then the excess evacuees can be placed at intermediate shelters that are comparatively safer than the danger zone. The various applications of the network flow with intermediate storage are

evacuation planning, demand-supply chain of goods, water supply system, etc. Pyakurel and Dempe [10] introduced the concept of maximum static and maximum dynamic flow problems with intermediate storage and presented polynomial time algorithms to solve them. They also presented polynomial time algorithm for dynamic contraflow problem with intermediate storage. In case of multisource multisink network, Pyakurel et al. [11] solved the prioritized maximum flow problem with intermediate storage and presented polynomial time algorithm to solve the problem, where priority is given to the farthest element from the source. Recently, Pyakurel and Dempe [12] presented efficient algorithms for universal maximum dynamic flow problem with intermediate storage in general as well as two-terminal series parallel networks.

In two-way network, contraflow (lane reversal) is one of the best techniques to increase the outbound capacities of arcs and minimize the overall time horizon, in which arcs are reversed towards the destination [13]. Rebennack et al. [14] provided the models and polynomial time algorithms for maximum and quickest flow problems in a two-terminal network by reverting the arcs at time zero and keeping them fixed afterward by using analytical approach for discrete-time settings. In continuous-time settings, Pyakurel and Dhamala [15] introduced the dynamic contraflow model. By using the natural transformation of Fleischer and Tardos [16], they have presented efficient algorithms to solve the maximum, quickest, and earliest arrival flow problems with lane reversals.

Pyakurel et al. [17] introduced the concept of partial lane reversals, in which only necessary arc capacities are reversed to increase the flow value, and unused arc capacities are saved for other emergency proposes like logistic supports and facility locations. Dhamala et al. [18] presented approximation algorithms for quickest multicommodity flow over time problem with partial lane reversals using length bound flow and condensed time expanded network in discrete-time settings. Continuous-time solutions of these problems are found in [19]. Similarly, Pyakurel et al. [20] presented polynomial time algorithm for maximum static and pseudopolynomial algorithm for maximum dynamic multicommodity flow problems with partial lane reversals.

In this paper, we aim to find the solution of discrete-time maximum multicommodity flow problem with intermediate storage by integrating the concept of multicommodity flow problem and the maximum flow problem with intermediate storage. We present polynomial time algorithm for static multicommodity flow problem and pseudopolynomial time algorithm for dynamic multicommodity flow problem by allowing the storage of excess flow at intermediate nodes. We extend the results for contraflow configuration with symmetric as well as asymmetric transit times and also in continuous-time settings.

Our models are designed with the following limitations: at each intermediate node, inflow must be greater or equal to the outflow. At each arc, flow must not exceed the capacity. The storage capacity of intermediate nodes must be at least the sum of incoming arc capacities. Every commodity must transship from respective sources to their corresponding

sinks. Objects within a commodity group are homogeneous and between the commodity groups are heterogeneous.

We organize the paper as follows. Section 2 provides the basic terminologies used in the paper and the mathematical formulation of flow models. In Section 3, we present a polynomial time algorithm to solve the maximum static multicommodity flow problem with intermediate storage, and in Section 4, we solve the maximum dynamic multicommodity flow problem with intermediate storage in pseudopolynomial time complexity. For two-way multicommodity network, we present a solution procedure of these problems in Section 5 within the same time complexity. Similarly, in Section 6, we extend the results of dynamic flow problems in continuous-time settings by using natural transformation. The paper is concluded in Section 7.

2. Basic Terminologies and Mathematical Models

Consider a dynamic network $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$, where V and $A \subseteq V \times V$ represent the sets of nodes and arcs with $|V| = n$ and $|A| = m$, respectively. Let $s_i \in S \subset V$ and $t_i \in D \subset V$ be the source and sink nodes with respect to commodity $i \in K = \{1, 2, \dots, k\}$ and $I = V \setminus \{S, D\}$ the set of intermediate nodes. Here, d_i represents the amount of supply from the source node s_i for each commodity $i \in K$ that is to be sent to the corresponding sink t_i and the intermediate nodes I . Each arc $a = (v, w) \in A$ with head $(a) = w$ and tail $(a) = v$ is equipped with a capacity function $\mathbf{u}: A \rightarrow \mathcal{R}^+$ that restricts the flow of commodity and a nonnegative transit time function $\tau: A \rightarrow \mathcal{R}^+$ that measures the time to transship the flow from node v to node w . Similarly, $\mathbf{b}: V \rightarrow \mathcal{R}^+$ represents the storage capacity function of nodes that is used to hold the flow at sources and sinks, together with the storage of excess flow leaving from the source s_i but not reaching the sink t_i at intermediate nodes. Capacity of arcs (roads) and the storage capacity of nodes (shelters) are the controlling parameters of our model, which control the flow at arcs and nodes, respectively. Let $\delta^{\text{out}}(v)$ and $\delta^{\text{in}}(v)$ be the set of outgoing arcs from node v and incoming arcs to node v , respectively. The time period T given in advance is denoted by $\mathcal{T} = \{0, 1, \dots, T\}$ in discrete-time settings and $\mathcal{T} = [0, T + 1)$ in continuous-time settings. In static flow, the transit time is considered as the cost, and time parameter T is absent.

Throughout the paper, we consider that the storage capacity of sources and sinks is sufficiently large, i.e., $\mathbf{b}_{s_i} = \mathbf{b}_{t_i} \leq \infty$ and that of intermediate nodes is finite. If the sum of incoming arc capacities of an intermediate node $v \in I$ is more than the sum of outgoing arc capacities, then the excess flow is used to store at v . Moreover, for the uniqueness of the solution, the storage capacity of $v \in I$ should be $\mathbf{b}_v \geq \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$.

2.1. Static Multicommodity Flow Model. The static multicommodity flow function g on the given network $\mathcal{N} = (V, A, K, c, \mathbf{u}, \mathbf{b}, d_i, S, D)$ is the sum of nonnegative arc flow functions $g_a^i: A \rightarrow \mathcal{R}^+$ and the excess flow functions

$g_v^i: I \rightarrow \mathcal{R}^+$, for each $i \in K$, satisfying conditions (1)–(5). The linear programming formulation of static multicommodity flow with intermediate storage is as follows:

$$\max d_i = \sum_{a \in \delta^{\text{out}}(s_i)} g_a^i = \sum_{a \in \delta^{\text{in}}(t_i)} g_a^i + \sum_{v \in I: \mathbf{b}_v \geq 0} g_v^i, \quad (1)$$

such that

$$\sum_{a \in \delta^{\text{in}}(v)} g_a^i - \sum_{a \in \delta^{\text{out}}(v)} g_a^i \geq 0, \quad \forall v \in I, i \in K, \quad (2)$$

$$0 \leq g_a = \sum_{i \in K} g_a^i \leq \mathbf{u}_a, \quad \forall a \in A, \quad (3)$$

$$0 \leq g_v = \sum_{i \in K} g_v^i \leq \mathbf{b}_v, \quad \forall v \in I, \quad (4)$$

$$\sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a \leq \mathbf{b}_v, \quad \forall v \in I. \quad (5)$$

Objective function in equation (1) is to maximize the total flow out from each source, for all $i \in K$, which is equal to the sum of inflow at the sink and the excess flow at

intermediate nodes. Equation (2) represents the nonconservation of flow at intermediate nodes. The constraint in (3) represents the bundle constraint on each arc that is bounded by its capacity, and the constraints in (4) represent the excess flow at each intermediate node, which is bounded by the storage capacity. Similarly, the constraint in (5) represents that the storage capacity of intermediate node $v \in I$ is at least the sum of incoming arc capacities to v . The cost of static flow g associated with arc a and commodity i with cost coefficient c_a^i is defined as

$$c(g) = \sum_{i \in K} \sum_{a \in A} c_a^i g_a^i. \quad (6)$$

2.2. Dynamic Multicommodity Flow Model. For a given dynamic network \mathcal{N} with constant transit time τ on each arc a , the multicommodity flow over time function ψ is the sum of nonnegative arc flow functions $\psi^i: A \times \mathcal{T} \rightarrow \mathcal{R}^+$ and the storage flow functions $\psi_v^i: I \times \mathcal{T} \rightarrow \mathcal{R}^+$ for each $i \in K$, satisfying constraints (7)–(11). The linear programming formulation of dynamic multicommodity flow with intermediate storage is as follows:

$$\max d_i = \sum_{a \in \delta^{\text{out}}(s_i)} \sum_{\theta=0}^T \psi_a^i(\theta) = \sum_{a \in \delta^{\text{in}}(t_i)} \sum_{\theta=\tau_a}^T \psi_a^i(\theta - \tau_a) + \sum_{v \in I: \mathbf{b}_v \geq 0} \psi_v^i(T), \quad (7)$$

such that

$$\sum_{a \in \delta^{\text{in}}(v)} \sum_{\beta=\tau_a}^{\theta} \psi_a^i(\beta - \tau_a) - \sum_{a \in \delta^{\text{out}}(v)} \sum_{\beta=0}^{\theta} \psi_a^i(\beta) \geq 0, \quad (8)$$

$$\forall v \in I, i \in K, \theta \in \mathcal{T},$$

$$0 \leq \psi_a(\theta) = \sum_{i \in K} \psi_a^i(\theta) \leq \mathbf{u}_a, \quad \forall a \in A, \theta \in \mathcal{T}, \quad (9)$$

$$0 \leq \psi_v(\theta) = \sum_{i \in K} \psi_v^i(\theta) \leq \mathbf{b}_v, \quad \forall v \in I, \theta \in \mathcal{T}, \quad (10)$$

$$\sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a \leq \mathbf{b}_v \leq T \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a, \quad \forall v \in I. \quad (11)$$

Equation (7) is an objective function that maximizes the total flow out from the source in time horizon T , for each $i \in K$, which is equal to the sum of inflow at sink and the excess flow at intermediate nodes. Equation (8) represents the nonconservation of flow at intermediate nodes for each time step θ . In any instance of time θ , the bundle constraint in (9) is bounded by arc capacity, and the constraint in (10) represents that the excess flow at each intermediate node is bounded by the storage capacity. Similarly, the constraint in (11) represents the lower and upper bounds of the storage capacity of intermediate node $v \in I$. The cost of discrete dynamic flow ψ associated with arc a and commodity i with cost coefficient c_a^i is defined as

$$c(\psi) = \sum_{i \in K} \sum_{a \in A} c_a^i \sum_{\theta=0}^T \psi_a^i(\theta). \quad (12)$$

3. Maximum Static Multicommodity Flow

In this section, we introduce the maximum static multicommodity flow problem with intermediate storage and present a polynomial time algorithm to solve it.

Problem 1. For a given static network $\mathcal{N} = (V, A, K, c, \mathbf{u}, \mathbf{b}, d, S, D)$, the maximum static multicommodity flow problem with intermediate storage finds the maximization of flow leaving from each source s_i , for all $i \in K$, which is to be sent to the corresponding sink t_i via $s_i - t_i$ paths by allowing the storage of maximum excess flow at intermediate nodes with storage capacity $\mathbf{b}_v \geq \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$.

As the solution strategy, we first reduce the multicommodity flow problem into k independent single commodity flow problems by reallocating the capacity of bundle arcs using the resource directive decomposition method. It reallocates the capacity of bundle arc for each commodity in such a way that the objective is optimal. The decomposition algorithm to minimal-cost multicommodity flow problem can be used for minimum cost flow problem, which was the motivation for the development of the original Dantzig–Wolfe decomposition method [21] (see Bazaraa et al. [22]). For each $i \in K$, we used to store the maximum flow at

sink t_i and the excess flow at intermediate nodes $v \in I$ with priority order. As in Pyakurel and Dempe [10], we have a single sink for each commodity $i \in K$, which is considered as the most appropriate place to store the flow. So, the first priority is given to the sink to transship as much flow as possible. To store the excess flow at intermediate nodes, we set the priority order as follows: for each $v \in I$ with storage capacity $\mathbf{b}_v \geq \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$, calculate the shortest distance $d_{[s_i, v]}$, for each i , by using algorithm of Dijkstra [23]. We consider the path with the minimum cost as the shortest path, and the priority is given to the farthest node among the nodes with shortest distance. That is, $\forall v_1, v_2 \in I$ if $d_{[s_i, v_1]} > d_{[s_i, v_2]}$ then v_1 is higher in priority than v_2 and it is denoted by $v_1 > v_2$. It is to be noted that the nodes lying in the bundle arcs may have different priority ordering with respect to the commodity.

For each prioritized node $v \in I$, we create dummy port v'_i (since the node $v \in I$ lying in the bundle arc contains the flow of more than one commodity, so dummy ports are represented commodity-wise, i.e., v'_i) with cost $c_{[v, v'_i]} = d_{[v, v'_i]} = 0$ and capacity $\mathbf{u}_{[v, v'_i]} = \mathbf{b}_v = \mathbf{b}_{v'_i}$, where $\mathbf{u}_{[v, v'_i]}$ and $c_{[v, v'_i]}$ are the arc capacity and cost of dummy arc (v, v'_i) , respectively. Every dummy port v'_i with respect to commodity i has the same priority order as v has. Associated with each commodity i , the collection of dummy ports $\{v'_i\}$ together with the sink t_i forms a modified network $\mathcal{N}'_i = (V'_i, A'_i, c, \mathbf{u}, \mathbf{b}, d_i, s_i, D'_i)$ with single source s_i and multiple sink $D'_i = \{t_i \cup \{v'_i\}\}$. For an instance, if $t_i > v_{i,1} > \dots > v_{i,r}$ is priority order of nodes with respect to commodity i , then $D'_i = \{t_i = v'_{i,0}, v'_{i,1}, \dots, v'_{i,r}\}$.

Now, we present a polynomial time algorithm to solve Problem 1 by using the algorithm of Pyakurel and Dempe [10] for single source multisink network $\mathcal{N}'_i, \forall i \in K$.

Theorem 1. *Algorithm 1 solves the maximum static multicommodity flow problem with intermediate storage optimally.*

Proof. Before proving the optimality, we first prove the feasibility of the algorithm. Step 1 is the use of decomposition algorithm to reduce the multicommodity flow problem to single commodity flow problem, and Step 2 calculates the shortest distances by using Dijkstra's algorithm, so both steps are feasible. Steps 3, 4, and 6 are prioritization of nodes, modification of network, and transformation of solution, which can be solved in linear time, and so they are feasible. Similarly, according to Pyakurel and Dempe [10], Step 5 provides feasible flow with intermediate storage for each commodity $i \in K$. Thus, the solution obtained from Algorithm 1 is feasible.

The optimality of algorithm is assured by Step 5. For each commodity $i \in K$, lexicographic maximum static flow in prioritized sink D'_i is obtained optimally as the single commodity flow problem solved by Pyakurel and Dempe [10]. So, the sum of optimal single commodity flows $\sum_i d_i$ is optimal multicommodity flow with intermediate storage. \square

Theorem 2. *Maximum static multicommodity flow problem with intermediate storage can be solved in polynomial time complexity by using Algorithm 1.*

Proof. The decomposition algorithm in Step 1 is obtained in polynomial time complexity, and the shortest distance can be obtained in $O(n^2)$ time by using Dijkstra's algorithm. The prioritization of nodes and creating dummy ports can be obtained in linear time. For each single commodity i , Step 5 can be solved in polynomial time complexity of $O(\delta mn)$ for $0 < \delta < n$, [10]. Therefore, Algorithm 1 solves the maximum static multicommodity flow problem with intermediate storage in polynomial time with complexity $O(n^2) + O(k\delta mn)$, where $|K| = k$. \square

Example 1. Consider a two-commodity network with capacity and cost on each arc as shown in Figure 1(a), where the numbers aside the nodes represent the node capacities. Using Dijkstra's algorithm, we find the shortest distance of each intermediate node and fix the priority order with farther-in-distance-higher-in-priority. So, the priority orders are $t_1 > y > x$ and $t_2 > x > y$ for commodity 1 and commodity 2, respectively. After priority ordering, we denote $D'_1 = \{t_1, y'_1, x'_1\}$ and $D'_2 = \{t_2, x'_2, y'_2\}$ as the set of prioritized dummy ports for commodity 1 and commodity 2, respectively, which is presented in Figure 1(b).

While decomposing the flow on the bundle arc (x, y) , flows of 3 and 2 units are assigned for commodity 1 and commodity 2, respectively. By using Algorithm 1, the maximum amount of flow leaving the source s_1 is 4 units out of which 2 units of flow are reached at sink t_1 and the intermediate nodes y and x hold 1 unit each. Similarly, 7 units of flow are transmitted from the source s_2 , out of which 6 units of flow are reached at the sink t_2 and the intermediate nodes x and y hold 1 and 0 units, respectively. At last, the solution is transformed to the original network by removing the dummy ports and dummy arcs and sending back the flow to its respective nodes. The amount of flow stored at sinks and the intermediate nodes is $g_{t_1} = 2, g_{t_2} = 6, g_x = 2$, and $g_y = 1$. If the intermediate storage is not permitted, then the flow of $g_{t_1} = 2$ and $g_{t_2} = 6$ units can only be transshipped.

4. Maximum Dynamic Multicommodity Flow

This section deals with the maximum dynamic multicommodity flow problem, where storage of the excess flow at intermediate nodes is allowed. We present a pseudopolynomial time algorithm based on the time expanded network of Kappmeier [8] to solve the problem.

Problem 2. For a given dynamic network $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$, the maximum dynamic multicommodity flow problem with intermediate storage is to maximize the flow leaving each source s_i , for all $i \in K$, which is to be sent to the corresponding sink t_i via $s_i - t_i$ paths by allowing the storage of maximum excess flow at the intermediate nodes with storage capacity $\sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a \leq \mathbf{b}_v \leq T \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$ within time horizon T .

Input: given static network $\mathcal{N} = (V, A, K, c, \mathbf{u}, \mathbf{b}, d_i, S, D)$.

Output: maximum static multicommodity flow with intermediate storage in \mathcal{N} .

- (1) Reconfigure the multicommodity flow problem into k independent single commodity flow problems by reallocating the capacity of bundle arcs using the resource directive decomposition.
- (2) For each $v \in I$ with $\mathbf{b}_v \geq \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$, compute commodity-wise shortest distance $d_{[s_i, v]}$ by using Dijkstra's algorithm.
- (3) Fix the priority order as $t_i > v_{i,1} > \dots > v_{i,r}$ with respect to commodity i with first priority to the sink t_i and priority for intermediate elements as $d_{[s_i, v_{i,m}]} > d_{[s_i, v_{i,m+1}]} \Rightarrow v_{i,m} > v_{i,m+1}$, for $m = 1, \dots, r-1$.
- (4) For each $i \in K$, construct the modified network $\mathcal{N}'_i = (V'_i, A'_i, c, \mathbf{u}, \mathbf{b}, d_i, s_i, D'_i)$ with single source s_i and multiple sinks with dummy ports $D'_i = \{t_i = v_{i,0}, v_{i,1}, \dots, v_{i,r}\}$.
- (5) For $i = 1, \dots, k$:
 Compute the lexicographic maximum static flow in \mathcal{N}'_i with priority order of Step 3 according to [10].
- (6) Transform the solution to the original network \mathcal{N} by removing the dummy ports and the dummy arcs.

ALGORITHM 1: Maximum static multicommodity flow algorithm with intermediate storage (MSMCF AIS).

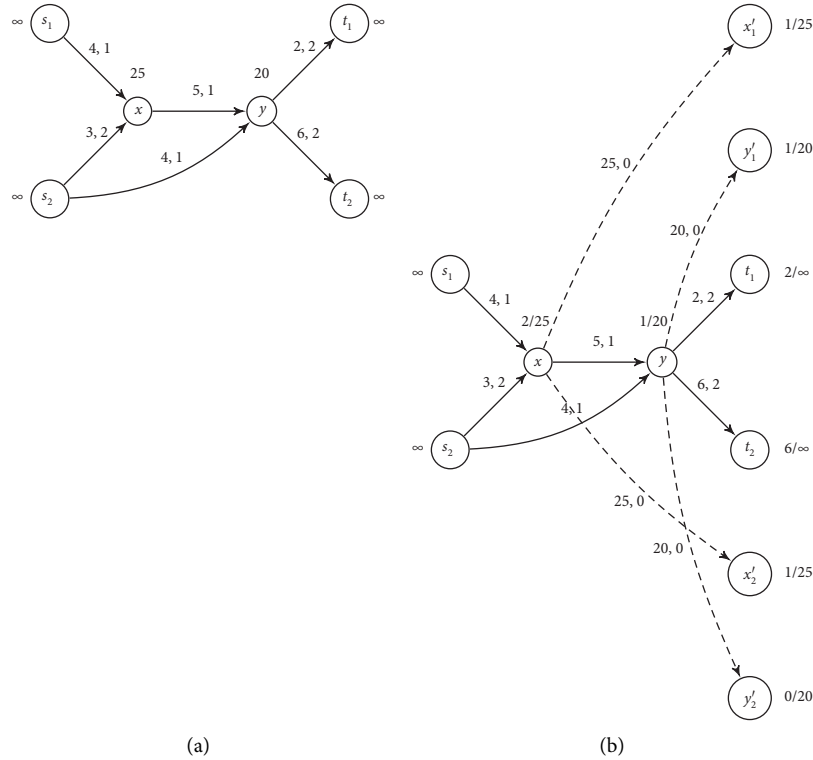


FIGURE 1: (a) A two-commodity network with capacity and cost on the arcs and storage capacity on the nodes. (b) The modified network of (a) with prioritized dummy ports.

As in Section 3, we first reduce the multicommodity flow problem into k independent subproblems and fix the priority order of intermediate nodes. Static solution is obtained in the modified single source and multisink network $\mathcal{N}'_i = (V'_i, A'_i, \mathbf{u}, \mathbf{b}, \tau, d_i, s_i, D'_i, T)$ for all $i \in K$, by using Algorithm 1. To obtain the dynamic solution, we use the static multicommodity flow on time expanded network as in Kappmeier [8].

For this, we construct a temporary sink \bar{t}_i with infinite capacity and join each dummy port $v'_i \in D'_i$ with $\mathbf{u}_{[v'_i, \bar{t}_i]} = g^i_{v'_i}(\theta)$ and $\tau_{[v'_i, \bar{t}_i]} = 0$, where $g^i_{v'_i}(\theta)$ is the static flow of

commodity i at v'_i at time θ . The choice of $\mathbf{u}_{[v'_i, \bar{t}_i]} = g^i_{v'_i}(\theta)$ is taken to assure that the flow while sending back from the dummy ports must be on respective nodes. Now, for each $i \in K$, the new network $\tilde{\mathcal{N}}'_i$ is obtained by adding temporary sink and arcs in \mathcal{N}'_i so that it becomes a single source single sink network with prioritized intermediate nodes. Let $\mathcal{N}^T = (V_T, A_T = A_M \cup A_H \cup A^s \cup A^t, K, \mathbf{u}, \mathbf{b}, \tau, d_i, \hat{S}, \hat{D}, T)$ be the time expanded network of new network $\tilde{\mathcal{N}}' = \cup_i \tilde{\mathcal{N}}'_i$, where $\tilde{\mathcal{N}}'_i$ is obtained by including temporary sinks, supersource, and supersink in \mathcal{N}'_i . The components in time expanded network \mathcal{N}^T are defined as follows:

$$\begin{aligned}
V_T &= \{v_\theta: v \in V, \theta \in \mathcal{T}\} \cup \{\{v_{i,\theta}'\} \cup \{\bar{t}_{i,\theta}\}: i \in K, \theta \in \mathcal{T}\} \cup \{s_i^*, \bar{t}_i^*: i \in K\} \cup \{\bar{s}, \bar{t}\}, \\
A_M &= \{(v_\theta, w_{\theta+\tau_a}): a = (v, w) \in A, \theta \in \mathcal{T}\} \cup \{(v_\theta, v_{i,\theta}') \cup (v_{i,\theta}', \bar{t}_{i,\theta}): v \in I, \theta \in \mathcal{T}\}, \\
A_H &= \{(v_\theta, v_{\theta+1}): v \in V, \theta \in \mathcal{T}\} \cup \{(v_{i,\theta}', v_{i,\theta+1}') \cup (\bar{t}_{i,\theta}, \bar{t}_{i,\theta+1}): i \in K, \theta \in \mathcal{T}\}, \\
A^s &= \{(\bar{s}, s_i^*): i \in K\} \cup \{(s_i^*, s_{i,\theta}): i \in K, \theta \in \mathcal{T}\}, \\
A^t &= \{(\bar{t}, \bar{t}_i^*): i \in K\} \cup \{(\bar{t}_i^*, \bar{t}_{i,\theta}): i \in K, \theta \in \mathcal{T}\}, \\
\widehat{S} &= \{\bar{s}, s_i^*, s_{i,\theta}: i \in K, \theta \in \mathcal{T}\}, \\
\widehat{D} &= \{\bar{t}, t_i^*, \bar{t}_{i,\theta}: i \in K, \theta \in \mathcal{T}\}.
\end{aligned} \tag{13}$$

Kappmeier [8] has shown that the static multi-commodity flow on the time expanded network is equivalent to the dynamic multicommodity flow on the original network.

Theorem 3. (see [8]). For a given dynamic network $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$ with time horizon T , a feasible static $\widehat{S} - \widehat{D}$ multicommodity flow g^T in the time expanded network \mathcal{N}^T yields the feasible dynamic multicommodity flow ψ in \mathcal{N} such that $|\psi| = |g^T|$.

We now present an algorithm to solve problem 2 by using time expanded network.

Theorem 4. Algorithm 2 provides an optimal solution to the maximum dynamic multicommodity flow problem with intermediate storage in pseudopolynomial time complexity.

Proof. At first, we prove the feasibility of Algorithm 2. As Step 1 is a reconfiguration of given network by using decomposition algorithm, and Step 2 is its modification including dummy ports and temporary sinks, so these steps provide the feasible solution. Due to Theorem 3, construction of time expanded network in Step 3 is feasible, and the feasibility of Step 4 is obtained by Algorithm 1. The transformation of solution in original network is also feasible. The optimality of algorithm is assured by the optimality of Step 4, which is as similar to [8].

Next, we prove the computational time of Algorithm 2, which is dominated by the complexity of Step 4. Here, the computational time complexity of Step 4 is $O(T \times \text{MSMCFAIS})$, where $O(\text{MSMCFAIS})$ is the complexity of maximum static multicommodity flow algorithm with intermediate storage. From Theorem 2, $O(\text{MSMCFAIS}) = O(n^2) + O(k\delta mn)$ for $0 < \delta < n$. With Step 4 being polynomial in input size of the network, but not bounded in T , Algorithm 2 solves the maximum dynamic multicommodity flow problem with intermediate storage in pseudopolynomial time. \square

Example 2. Consider a two-commodity network from Figure 1 by considering the cost on each arc as the transit time with time horizon $T = 5$. The prioritization of intermediate nodes and creating dummy ports are similar to Example 1. For each commodity $i = 1, 2$, the problem is

reduced to single source and multisink single commodity flow problem due to dummy ports (see Figure 1(b)). By adding temporary sink \bar{t}_i , it reduces to commodity-wise single source single sink (i.e., $s_i - \bar{t}_i$) problem, which is shown in Figure 2. We calculate the maximum static multicommodity flow with intermediate storage using Algorithm 1 and then repeat the procedure with intermediate storage in time expanded network as similar to Kappmeier [8]. Here, the maximum static flow is calculated from minimum cost flow by considering the transit time as cost. Figure 3 represents the time expanded network of Figure 2 with time horizon $T = 5$. At last, we obtain the maximum dynamic flow with intermediate storage by removing the dummy arcs and replacing the flow of dummy ports to their respective nodes.

For commodity 1, total amount of flow leaving the source s_1 within the time horizon $T = 5$ along the path $s_1 - x - y - t_1$ is 20 units, out of which 4, 8, and 8 units are transshipped with priority order at t_1 , y and x , respectively. For commodity 2, flows are sent through two paths $s_2 - x - y - t_2$ and $s_2 - y - t_2$ with priority order $t_2 > x > y$. It is to be noted that while sending flow from the path $s_2 - x - y - t_2$, flow leaving s_2 at $\theta = 0$ sends 2 units of flow at t_2 by storing 1 unit at x . After next iteration onward, flow cannot reach the sink, and so it is to be stored at x but not at y because x is higher in priority than y . Similarly, path $s_2 - y - t_2$ first sends 4 units of flow thrice at t_2 and then holds the flow at y for next two times. Total amount of flow leaving the source s_2 through two paths within $T = 5$ is 32 units, out of which 14, 10, and 8 units are transshipped at t_2 , x and y , respectively. The detailed information of the flow leaving from sources at different time steps θ , which are to be stored at sinks and the intermediate nodes, is presented in Table 1.

Here, the amount of flow reaching the sinks is the maximum multicommodity flow without intermediate storage. So, if intermediate storage is prohibited, then only 4 units of flow of commodity 1 and 14 units of flow of commodity 2 can be reached at their respective sinks within the time $T = 5$.

5. Multicommodity Contraflow Problems

In this section, we investigate the multicommodity flow problem with contraflow configuration, where the storage of excess flow at intermediate nodes is allowed. In a two-way network, contraflow means the reversal of oppositely directed arcs towards the destination node to improve the flow and reduce the overall

Input: given dynamic network $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$.
Output: maximum dynamic multicommodity flow with intermediate storage in \mathcal{N} .

- (1) Reconfigure the multicommodity flow problem into k independent single commodity flow problems by reallocating the capacity of bundle arcs by using the resource directive decomposition and considering the transit times as cost.
- (2) Construct a modified new network $\tilde{\mathcal{N}}'$ by including temporary sinks, supersource, and supersink in \mathcal{N}' .
- (3) Using new network $\tilde{\mathcal{N}}'$, construct the time expanded network $\mathcal{N}^T = (V_T, A_T = A_M \cup A_H \cup A^S \cup A^t, K, \mathbf{u}, \mathbf{b}, \tau, d_i, \hat{S}, \hat{D}, T)$.
- (4) Calculate the maximum static multicommodity flow with intermediate storage on the time expanded network \mathcal{N}^T by using Algorithm 1.
- (5) Transform the solution to the original network \mathcal{N} by removing the dummy ports, temporary sinks, supersource, supersink, and dummy arcs.

ALGORITHM 2: Maximum dynamic multicommodity flow algorithm with intermediate storage (MDMCFAS).

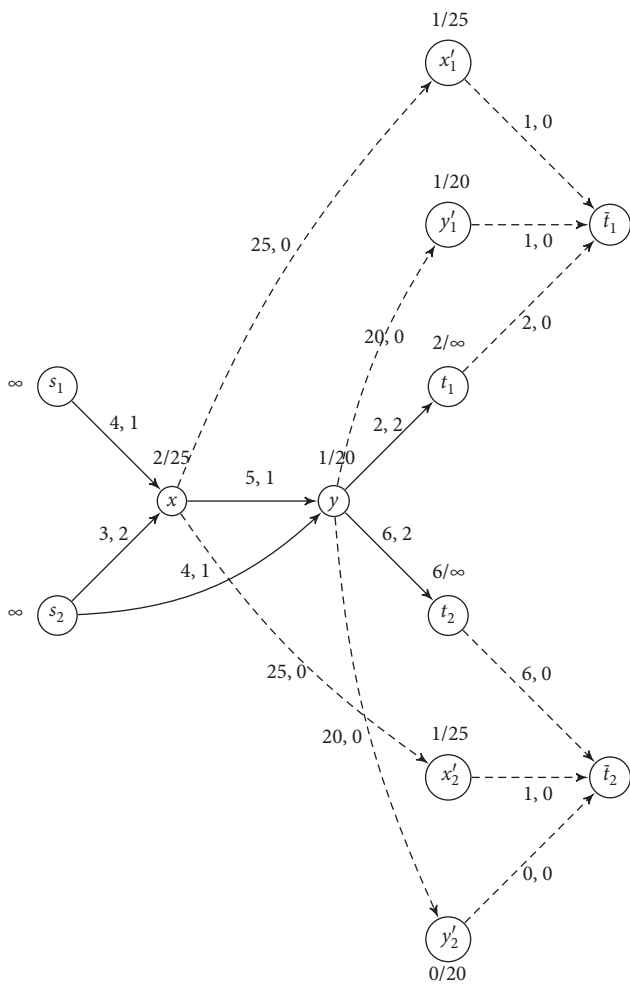


FIGURE 2: Modified new network $\tilde{\mathcal{N}}'$ with dummy ports and temporary sinks.

time horizon. We discuss two different aspects of transit times (or cost for static), symmetric and asymmetric, between the pair of nodes with oppositely directed arcs.

5.1. *Contraflow with Symmetric Transit Times.* Let $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$ be a dynamic multicommodity network having symmetric transit times on

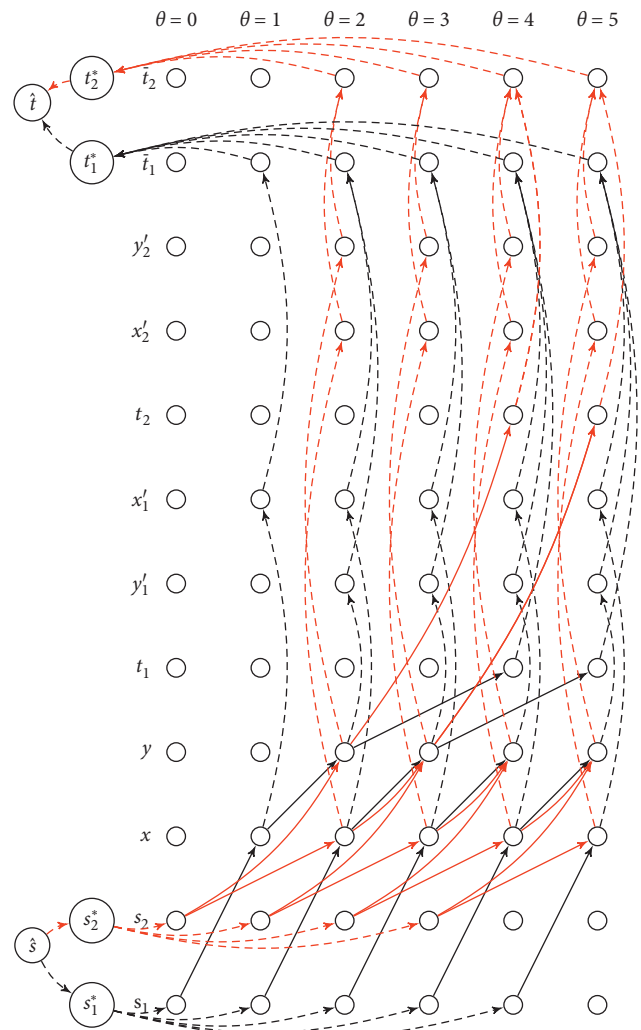


FIGURE 3: Time expanded multicommodity network flow with intermediate storage, where black and red colors are for commodity 1 and commodity 2, respectively. Dotted arcs represent the dummy arcs.

antiparallel arcs, i.e., $\tau_a = \tau_{\bar{a}}$, where $\bar{a} = (w, v)$ is oppositely directed arc of $a = (v, w)$. To solve the problem in contraflow network, we transform the given network to an auxiliary network as follows.

TABLE 1: Multicommodity flow with intermediate storage in each time θ .

Start time at s_1	Commodity 1				Start time at s_2	Commodity 2			
	Path: $s_1 - x - y - t_1$					Path: $s_2 - x - y - t_2$			
	Flow at			Reaching time at last node		Flow at			Reaching time at last node
	x	y	t_1		x	y	t_2		
$\theta=0$	1	1	2	$\theta=4$ at t_1	$\theta=0$	1	0	2	$\theta=5$ at t_2
$\theta=1$	1	1	2	$\theta=5$ at t_1	$\theta=1$	3	\times	\times	$\theta=3$ at x
$\theta=2$	1	3	\times	$\theta=4$ at y	$\theta=2$	3	\times	\times	$\theta=4$ at x
$\theta=3$	1	3	\times	$\theta=5$ at y	$\theta=3$	3	\times	\times	$\theta=5$ at x
$\theta=4$	4	\times	\times	$\theta=5$ at x					
						Path: $s_2 - y - t_2$			
					$\theta=0$	\times	0	4	$\theta=3$ at t_2
					$\theta=1$	\times	0	4	$\theta=4$ at t_2
					$\theta=2$	\times	0	4	$\theta=5$ at t_2
					$\theta=3$	\times	4	\times	$\theta=4$ at y
					$\theta=4$	\times	4	\times	$\theta=5$ at y
Total	8	8	4	Total $d_1 = 20$	Total	10	8	14	Total $d_2 = 32$

For a given two-way multicommodity network \mathcal{N} with symmetric transit times, the corresponding auxiliary network is denoted by $\overline{\mathcal{N}}$ with network topology $\overline{\mathcal{N}} = (V, \overline{A}, K, \overline{\mathbf{u}}, \overline{\mathbf{b}}, \overline{\tau}, \overline{d}_i, S, D, T)$, containing undirected edges $\overline{A} = \{(v, w) : (v, w) \text{ or } (w, v) \in A\}$. The capacity of an arc in an auxiliary network is the sum of capacities of arcs a and \overline{a} such that $\overline{\mathbf{u}}_a = \mathbf{u}_a + \mathbf{u}_{\overline{a}}$, where $\mathbf{u}_a = 0$ if $a \notin A$. The transit time of an arc in an auxiliary network is

$$\overline{\tau}_a = \begin{cases} \tau_a, & \text{if } a \in A, \\ \tau_{\overline{a}}, & \text{otherwise.} \end{cases} \quad (14)$$

All other parameters are the same as in \mathcal{N} . Contrary to the general network, incoming arcs to the sources s_i and outgoing arcs from the sinks t_i may be present in the contraflow network for all $i \in K$.

We now present the maximum dynamic contraflow problem with intermediate storage herein.

Problem 3. For a given dynamic network $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$, the maximum dynamic multicommodity contraflow problem with intermediate storage is to find the maximum flow leaving from each source s_i , $\forall i \in K$, which is to be sent to their respective sinks t_i via $s_i - t_i$ paths by allowing the storage of maximum excess flow at intermediate nodes $v \in I$ with storage capacity $\sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a \leq \mathbf{b}_v \leq T \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a$, $\forall a \in \overline{A}$ within the given time horizon T by reverting the direction of arcs at time zero.

To solve the problem, we present an algorithm based on the time expanded network of an auxiliary network $\overline{\mathcal{N}}$ as follows.

We first transform the given two-way network into an auxiliary network $\overline{\mathcal{N}}$. As in Section 3, we decompose the multicommodity flow problem to k single commodity flow problems and then fix the priority order of each intermediate node. On each cycle free path of auxiliary network $\overline{\mathcal{N}}$, we solve the maximum dynamic multicommodity flow problem, for each $i \in K$, as described in Section 4.

Here, Steps 1, 3, and 4 of Algorithm 3 can be computed in $O(E)$ time. In Step 2, we can find a pseudopolynomial time solution for dynamic multicommodity contraflow problem with intermediate storage in $\overline{\mathcal{N}}$ (as in Theorem 4). So, the maximum dynamic multicommodity contraflow problem with intermediate storage can be solved in pseudopolynomial time by using Algorithm 3.

Theorem 5. Algorithm 3 solves the maximum dynamic multicommodity contraflow problem with intermediate storage in pseudopolynomial time complexity.

Remark 1. If we consider the cost instead of transit time and apply Algorithm 1 in Step 2 of Algorithm 3, then the solution of maximum static multicommodity contraflow problem with intermediate storage can be obtained in polynomial time complexity.

5.2. Contraflow with Orientation-Dependent Transit Times. If the transit times on antiparallel arcs of a two-way network are not identical, then it is known as the network with asymmetric transit times. For a network with asymmetric transit times, if the transit times of arcs in an auxiliary network are taken as the orientation of the arcs, then it is known as orientation-dependent transit time. Nath et al. [24] considered the orientation-dependent asymmetric transit times of reversed lanes in general form and presented strongly polynomial time algorithms to solve the single source single sink maximum dynamic and quickest contraflow problems. Here, we discuss about the multicommodity contraflow problem with intermediate storage by taking orientation-dependent transit times.

Let $\mathcal{N} = (V, A, K, \mathbf{u}, \mathbf{b}, \tau, d_i, S, D, T)$ be a two-way dynamic network with asymmetric nonzero transit times τ on arcs, so that $\tau_a \neq \tau_{\overline{a}}$. We construct an auxiliary network $\overline{\mathcal{N}} = (V, \overline{A}, K, \overline{\mathbf{u}}, \overline{\mathbf{b}}, \overline{\tau}, d_i, S, D, T)$, where \overline{A} is obtained by reverting the direction of arcs \overline{a} at time zero. The arc capacity $\overline{\mathbf{u}}$ and transit time $\overline{\tau}_a$ can be obtained as follows:

$$\overline{\mathbf{u}}_a = \mathbf{u}_a + \mathbf{u}_{\overline{a}}, \quad (15)$$

Input: given two-way dynamic multicommodity network \mathcal{N} .

Output: maximum dynamic multicommodity contraflow with intermediate storage.

- (1) Construct an auxiliary network $\overline{\mathcal{N}}$ of \mathcal{N} .
- (2) Compute the maximum dynamic multicommodity flow with intermediate storage in $\overline{\mathcal{N}}$ by using Algorithm 2.
- (3) Decompose the flow along $s_i - t_i$ paths and cycles, and remove the flows in cycles $\forall i \in K$.
- (4) An arc \bar{a} is reversed if and only if the flow along arc a is greater than its capacity, or if there is a nonnegative flow along arc $a \notin A$.

ALGORITHM 3: Maximum dynamic multicommodity contraflow algorithm with intermediate storage.

where $u_a = 0$ if $a \notin A$ and

$$\bar{\tau}_a = \begin{cases} \tau_a & \text{if the orientation of arc is along } a, \\ \tau_{\bar{a}} & \text{if the orientation of arc is along } \bar{a}. \end{cases} \quad (16)$$

By using Algorithm 3, the optimal solution for maximum static and maximum dynamic multicommodity contraflow problems with intermediate storage can be obtained for a given network with asymmetric transit times, where transit time is considered as a cost in static problem.

If, on the other hand, the transit time attributes on the evacuation network are antisymmetric, but one wishes to adopt contraflow approach with the same transit time on the reversed arcs as it was before its reversal, Algorithm 3 obviously works well as that can be interpreted as symmetric transit time solution. It can be applied if two nodes in a network have parallel connections with different transit times.

6. Continuous Dynamic Multicommodity Flow

We discussed the maximum dynamic flow and contraflow problems with intermediate storage in Section 4 and Section 5, respectively, where transit times are taken in discrete settings. Actually, discrete dynamic flow function ψ assigns the flow from the source node at each time step $\theta = 0, \dots, T$ satisfying the capacity constraints. In this section, we formulate the dynamic multicommodity flow with intermediate storage in continuous-time settings. A continuous dynamic

flow function ψ^c with intermediate storage is defined as the flow rate per unit time that leaves from the source at each moment of time by allowing the storage of excess flow at intermediate nodes without violating the capacity constraints.

By using natural transformation, Fleischer and Tardos [16] established the relation between discrete and continuous flow models. This natural transformation defines the continuous dynamic flow for time interval $[\theta, \theta + 1]$ with $\psi_a^c[\theta, \theta + 1] = \psi_a(\theta)$, where $\psi_a(\theta)$ is the amount of discrete dynamic flow entering arc $a \in A$ at each time step $\theta = 0, \dots, T$. Here, we use this logic to transform the discrete-time multicommodity flow to continuous-time multicommodity flow with intermediate storage as follows: any discrete multicommodity flow over time ψ_a^i with integral time horizon T is equivalent to the continuous multicommodity flow over time $\psi_a^{i,c}[\theta, \theta + 1]$ by incorporating the flow ψ entering an arc a at time step $\theta \leq T - \tau_a$ as a constant flow rate on arc a during the unit time interval $[\theta, \theta + 1]$ by allowing the storage of excess flow at intermediate nodes. Mathematically,

$$\int_{\theta}^{\theta+1} \psi_a^{i,c}(\alpha) d\alpha = \psi_a^i(\theta), \quad \forall a \in A, i \in K. \quad (17)$$

We formulate the dynamic multicommodity flow models with intermediate storage in continuous settings as follows:

$$\max d_i = \sum_{a \in \delta^{\text{out}}(s_i)} \int_{\theta=0}^T \psi_a^{i,c}(\theta) d\theta = \sum_{a \in \delta^{\text{in}}(t_i)} \int_{\theta=\tau_a}^T \psi_a^{i,c}(\theta - \tau_a) d\theta + \sum_{v \in I: \mathbf{b}_v \geq 0} \int_{\theta=0}^T \psi_v^{i,c}(\theta) d\theta, \quad (18)$$

such that

$$\sum_{a \in \delta^{\text{in}}(v)} \int_{\beta=\tau_a}^{\theta} \psi_a^{i,c}(\beta - \tau_a) d\beta - \sum_{a \in \delta^{\text{out}}(v)} \int_{\beta=0}^{\theta} \psi_a^{i,c}(\beta) d\beta \geq 0, \quad \forall v \in I, i \in K, \theta \in \mathcal{T}, \quad (19)$$

$$0 \leq \psi_a^c(\theta) = \sum_{i \in K} \psi_a^{i,c}(\theta) \leq u_a, \quad \forall a \in A, \theta \in \mathcal{T}, \quad (20)$$

$$0 \leq \psi_v^c(\theta) = \sum_{i \in K} \psi_v^{i,c}(\theta) \leq \mathbf{b}_v, \quad \forall v \in I, \theta \in \mathcal{T}, \quad (21)$$

$$\sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a \leq \mathbf{b}_v \leq T \sum_{a \in \delta^{\text{in}}(v)} \mathbf{u}_a, \quad \forall v \in I. \quad (22)$$

Here, equation (18) represents an objective function, which is to maximize the total amount of flow out from the source in continuous-time settings, which is sent to the sink and the intermediate nodes. The nonconservation of the flow is represented by equation (19). Equations (20)–(22) have their usual meanings as in Section 2.

Dynamic flow problems defined in Section 4 and Section 5 can be solved in continuous-time settings by using this natural transformation with their respective algorithms within the same time complexity.

7. Conclusion

If the amount of flow out from the source node is more than the minimum cut capacity, then the excess flow cannot reach the sink. To deal with this problem, maximum static, maximum dynamic, and maximum dynamic contraflow problems with the storage of excess flow at intermediate nodes have been studied in two-terminal general network.

In this paper, we have investigated the multi-commodity flow models with intermediate storage in static as well as dynamic networks. We have introduced the maximum static and maximum dynamic multi-commodity flow problems. We have presented a polynomial time algorithm to solve the maximum static multicommodity flow problem and a pseudopolynomial time algorithm for the maximum dynamic multi-commodity flow problem by allowing the storage of excess flow at intermediate nodes. Moreover, we have presented an algorithm to solve the maximum dynamic multi-commodity contraflow problem with symmetric as well as asymmetric transit times. By using natural transformation in multicommodity network, we solved the maximum dynamic flow and maximum dynamic contraflow problems with intermediate storage in continuous-time settings. To the best of our knowledge, the maximum multicommodity flow problems with intermediate storage and their solution strategies for the static flow, dynamic flow, and contraflow problems are introduced for the first time.

The universally maximum multicommodity flow problem (the earliest arrival flow problem) with intermediate storage is harder problem, which is of interest for the further research. This problem is \mathcal{NP} -hard even in case of two-terminal series parallel networks. Together with this, we are interested to work on maximum flow and earliest arrival flow problems with orientation-dependent transit times and flow-dependent transit times. We are also interested in implementing these techniques as a case study in Kathmandu road network.

Data Availability

The authors have not used any additional data in this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

References

- [1] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.
- [2] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, USA, 1962.
- [3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithm and Applications*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.
- [4] A. Ali, R. Helgason, J. Kennington, and H. Lall, "Technical note-computational comparison among three multi-commodity network flow algorithms," *Operations Research*, vol. 28, no. 4, pp. 995–1000, 1980.
- [5] A. A. Assad, "Multicommodity network flows—a survey," *Networks*, vol. 8, no. 1, pp. 37–91, 1978.
- [6] J. L. Kennington, "A survey of linear cost multicommodity network flows," *Operations Research*, vol. 26, no. 2, pp. 209–236, 1978.
- [7] A. Hall, S. Hippler, and M. Skutella, "Multicommodity flows over time: efficient algorithms and complexity," *Theoretical Computer Science*, vol. 379, no. 3, pp. 387–404, 2007.
- [8] P. W. Kappmeier, *Generalizations of flows over time with application in evacuation optimization*, PhD Thesis, Technical University, Berlin, Germany, 2015.
- [9] D. Khanal, U. Pyakurel, U. Pyakurel, and T. Dhamala, "Prioritized multi-commodity flow model and algorithm," in *Proceedings of the International Symposium on Analytic Hierarchy Process 2020 (ISAHP)*, London, UK, August 2020.
- [10] U. Pyakurel and S. Dempe, "Network flow with intermediate storage: models and algorithms," *SN Operations Research Forum*, vol. 1, no. 4, 2020.
- [11] U. Pyakurel, S. Wagle, and M. C. Adhikari, "Efficient lane reversals for prioritized maximum flow" *International Journal of Innovative Science, Engineering & Technology*, vol. 7, pp. 354–363, 2020.
- [12] U. Pyakurel and S. Dempe, "Universal maximum flow with intermediate storage for evacuation planning," in *Dynamics of Disasters*, I. S. Kotsireas, A. Nagurney, P. M. Pardalos, and A. Tsokas, Eds., Springer, Berlin, Germany, 2021.
- [13] 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, no. 3, pp. 101–133, 2018.
- [14] S. Rebennack, A. Arulsevan, 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.
- [15] 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.
- [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] 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.
- [18] T. N. Dhamala, S. P. Gupta, D. P. Khanal, and U. Pyakurel, "Quickest multi-commodity flow over time with partial lane reversals," *Journal of Mathematics and Statistics*, vol. 16, no. 1, pp. 198–211, 2020.
- [19] S. P. Gupta, D. P. Khanal, U. Pyakurel, and T. N. Dhamala, "Approximate algorithms for continuous-time quickest multi-commodity contraflow problem," *The Nepali Mathematical Sciences Report*, vol. 37, no. 1-2, pp. 30–46, 2020.
- [20] U. Pyakurel, S. P. Gupta, D. P. Khanal, and T. N. Dhamala, "Efficient algorithms on multicommodity flow over time problems with partial lane reversals," *International Journal of Mathematics and Mathematical Sciences*, vol. 2020, Article ID 2676378, 13 pages, 2020.
- [21] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960, <http://www.jstor.org/stable/167547>.
- [22] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, John Wiley & Sons, Hoboken, NY, USA, 4th edition, 2010.
- [23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [24] H. N. Nath, U. Pyakurel, and T. N. Dhamala, "Network reconfiguration with orientation-dependent transit times," *International Journal of Mathematics and Mathematical Sciences*, vol. 2021, Article ID 6613622, 11 pages, 2021.

**DYNAMIC CONTRAFLOW WITH ORIENTATION DEPENDENT
TRANSIT TIMES ALLOWING INTERMEDIATE STORAGE**

**DURGA PRASAD KHANAL¹, URMILA PYAKUREL², STEPHAN
DEMPE³**

¹ *Saraswati Multiple Campus, Tribhuvan University, Kathmandu, Nepal*

² *Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal*

³ *Faculty of Mathematics and Computer Science, TU Bergakademie Freiberg, Germany*

Emails: ¹durgapsdkhanal@gmail.com, ²urmilapyakurel@gmail.com,

³dempe@math.tu-freiberg.de

² Corresponding author: urmilapyakurel@gmail.com

Abstract: Contraflow is one of the best and widely accepted techniques in evacuation planning problems, where the reversal of arcs is made to increase the amount and decrease the time of flow transmission. At the time of evacuation, if the flow value leaving the source node exceeds the bottleneck capacity of the network, then the storage of excess flow at comparatively safer intermediate nodes can be a milestone to save the life of evacuees. In this paper, we introduce the maximum dynamic contraflow problem in a general network and earliest arrival contraflow problem in two-terminal series-parallel network with asymmetric transit times on anti-parallel arcs. We present polynomial time solution strategies with orientation dependent transit times by allowing the storage of flow at intermediate nodes.

Key Words: Maximum flow, earliest arrival flow, intermediate storage, contraflow, symmetric and asymmetric transit times.

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

1. INTRODUCTION

Network flow over time problems, also known as dynamic flow problems, deal with the transshipment of flow from one point to other using running time i.e., traversal time. Ford and Fulkerson are the pioneers of this problem which has been introduced more than sixty years ago, [9]. Maximum dynamic flow problem concerns the shifting of maximum amount of flow from the origin node (source) to the destination node (sink) in the given time horizon. Evacuation planning problems can be modeled as the maximum dynamic flow problem, which is also applicable in transportation management, communication system and production planning etc. Similarly, earliest arrival flow problem treats with the maximization of flow at each time step, [11]. For the EAF problem, Minieka [21] and Wilkinson [29] provided the algorithms by adopting the successive shortest path augmenting algorithm which needed exponential time. In a two-terminal general network, Pyakurel and Dhamala [25] presented optimal solution to the earliest arrival flow problem for discrete and continuous time settings. The detailed illustrations of maximum flow can be found in the books [1, 10], book series of [20], survey papers [2, 19, 7, 14] and the references therein.

Existence of the excess flow occurs if the amount of flow leaving the source node is greater than the minimum cut capacity. In this case, the excess flow cannot reach the destination node and the storage of such flow at appropriate intermediate nodes is an important issue. The maximum flow problem with intermediate storage is a very relevant task in large scale disaster management problems. By introducing

the concept of intermediate storage, Pyakurel and Dempe [23] provided the maximum static and maximum dynamic flow problems and presented the polynomial time solution strategies to solve the problems. Not only in disaster management and evacuation planning, transshipment of flow with intermediate storage is highly applicable for different demand-supply chains like commodity supply, electricity distribution, water supply and so on.

In a two-way network topology, contraflow (lane reversal) means the reversing of the direction of arcs to increase the amount of flow transmission and reduce the total traversal time. Rebennack et al. [27] presented a polynomial time algorithm to solve the maximum flow problem in two-terminal network by reverting the direction of arcs at time zero and kept fixed afterwards. Various problems with contraflow can be found in [3, 4, 17, 18]. The maximum dynamic contraflow (MDCF) problem with intermediate storage and its polynomial solution can be found in Pyakurel and Dempe [23]. By reverting only necessary arcs, Dhamala et al. [5] presented algorithms to solve the quickest multi-commodity contraflow problem, where given amount of flow is to be transshipped in minimum possible time. The maximum multi-commodity flow over time problem with partial contraflow is presented by Pyakurel et al. [26]. Similarly, in general as well as two-terminal series-parallel networks, efficient algorithms for the universal maximum flow and contraflow problems with intermediate storage can be found in Pyakurel and Dempe [24]. For these contraflow problems, networks are considered with symmetric transit times in anti-parallel arcs.

For contraflow network with asymmetric transit times, if the transit time of reversal arc depends on the orientation of arc, then it is known as orientation dependent transit times (ODTT). Different transit times on anti-parallel arcs exist in the real network scenario of private as well as public transportation due to the speed limits, time card system, condition of roads and government policies. To illustrate ODTT with an example, consider a two-way dynamic network with asymmetric transit times on arcs as presented in Figure 1(i). The numbers on each arc represent the capacity u and transit time τ , whereas numbers on the nodes are storage capacities b . The other four figures are the possible auxiliary networks with ODTT. Figure 1(ii) is formed by taking directions along (x, y) and (x, t) , whereas Figure 1(iii) with directions along (y, x) and (x, t) . Similarly, Figure 1(iv) is formed by taking directions along (x, y) and (t, x) , whereas Figure 1(v) is with directions along (y, x) and (t, x) . Nath et al. [22] presented the concept of contraflow with orientation dependent transit times, where asymmetric transit times of reversed lanes in general form is considered. They have presented strongly polynomial time algorithms to solve the maximum dynamic and quickest contraflow problems in single source single sink network. Gupta et al. [12] extended the analytical solution of generalized dynamic contraflow problem by considering the asymmetric transit times on arcs and presented an efficient algorithm to solve the problem. The earliest arrival transshipment contraflow problem in multi-source single sink network and the prioritized maximum dynamic partial contraflow problem in single source multi-sink network by considering the non-symmetric transit times on the arcs can be found in [13]. Similarly, for maximum multi-commodity contraflow problems with symmetric and asymmetric transit times, Khanal et al. [16] presented the pseudo polynomial time algorithms to solve the problems.

In this paper, we present the models for static as well as dynamic flow by incorporating the intermediate storage of the flow. We introduce the MDCF problem with ODTT by allowing the storage of excess flow at intermediate nodes and present a polynomial time algorithm to solve it. We also introduce an earliest arrival contraflow (EACF) problem with ODTT and propose a solution in polynomial time complexity by allowing the intermediate storage.

The paper is organized as follows. In Section 2, we present a contraflow network and its auxiliary network with symmetric as well as asymmetric transit times. We also formulate static and dynamic flow models with intermediate storage. The MDCF with ODTT and EACF with ODTT are presented in Section 3 and Section 4, respectively, which permit the storage of flow at intermediate nodes. We conclude the paper in Section 5.

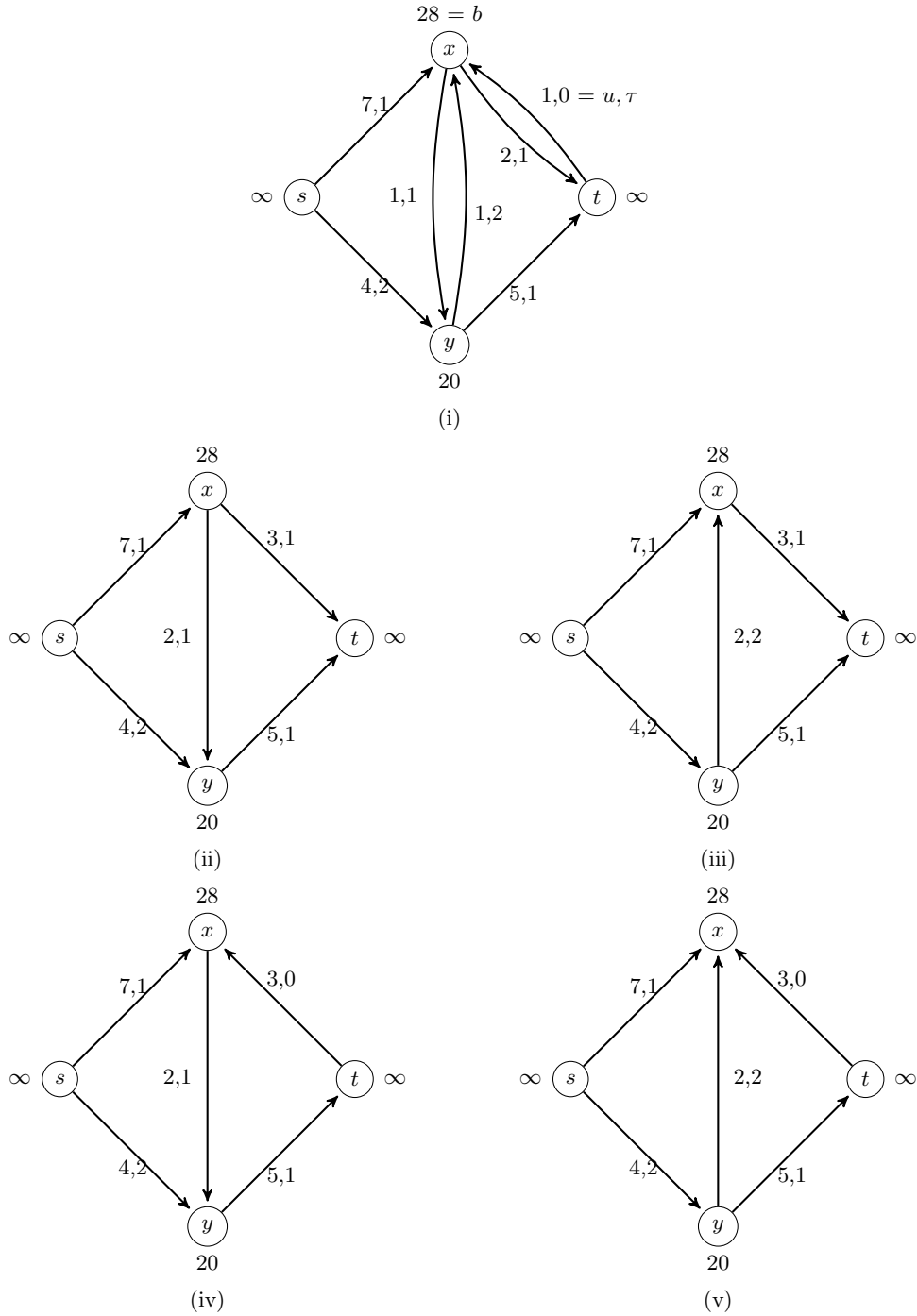


FIGURE 1. (i) represents a contraflow network with asymmetric transit times and (ii)-(v) represent possible orientation dependent auxiliary networks of (i). Numbers aside the nodes are the storage capacities.

2. BASIC TERMINOLOGIES AND FLOW MODELS

2.1. **Contraflow Network.** Consider a two-way network topology $\mathcal{N} = (V, A, u, b, \tau, s, t, T)$, where V represents the set of nodes and $A \subseteq V \times V$ represents the set of arcs. Let $|V| = n$ and $|A| = m$. Here, $s \in V$ and $t \in V$ are the source (origin) and sink (destination) nodes and $I = V \setminus \{s, t\}$ represents the set of intermediate nodes. Each arc $a = (v, w) \in A$ with head(a) = w and tail(a) = v has a capacity function

$u : A \rightarrow \mathcal{R}^+$ that limits the flow on arc and a non-negative transit time function $\tau : A \rightarrow \mathcal{R}^+$ that measures the transmission time from v to w . Similarly, $\overleftarrow{a} = (w, v)$ is an oppositely directed arc of $a = (v, w)$ with capacity $u_{\overleftarrow{a}}$ and transit time $\tau_{\overleftarrow{a}}$. Let $b : V \rightarrow \mathcal{R}^+$ represents the storage capacity function of the nodes. We denote the set of outgoing arcs from node v and incoming arcs to node v by $\delta^{out}(v)$ and $\delta^{in}(v)$, respectively. The time period T given in advanced is denoted by $\mathcal{T} = \{0, 1, \dots, T\}$ in discrete time settings. In static flow, the time parameters are absent.

Auxiliary Network with Symmetric Transit Times. Consider a dynamic network \mathcal{N} having anti-parallel arcs of symmetric transit times i.e., $\tau_a = \tau_{\overleftarrow{a}}$. The auxiliary network of given network \mathcal{N} is denoted by $\overline{\mathcal{N}} = (V, \overline{A}, \overline{u}, b, \overline{\tau}, s, t, T)$, containing undirected edges $\overline{A} = \{(v, w) : (v, w) \text{ or } (w, v) \in A\}$. The capacity \overline{u}_a of an arc is the sum of capacities of anti-parallel arcs a and \overleftarrow{a} , i.e., $\overline{u}_a = u_a + u_{\overleftarrow{a}}$, where $u_a = 0$ if $a \notin A$. Similarly, the transit time of an arc in this network is obtained as

$$\overline{\tau}_a = \begin{cases} \tau_a & \text{if } a \in A \\ \tau_{\overleftarrow{a}} & \text{otherwise.} \end{cases}$$

All others parameters are as same in \mathcal{N} .

Auxiliary Network with ODTT. In a two-way dynamic network, if the transit times on anti-parallel arcs are not identical, then it is known as the network with asymmetric transit times. In contraflow configuration, the flow on anti-parallel arcs traverses in one of the directions but not the both directions. In auxiliary network, if we consider the transit time of the arc depending on its orientation, then it is defined as the contraflow with orientation dependent transit times. For a two-way dynamic network \mathcal{N} with asymmetric transit time on arcs, an auxiliary network $\overline{\mathcal{N}} = (V, \overline{A}, \overline{u}, b, \overline{\tau}, s, t, T)$ is constructed in such a way that \overline{A} contains the undirected arcs obtained by reverting the direction of arcs \overleftarrow{a} at time zero. The arc capacity \overline{u}_a and transit time $\overline{\tau}_a$ are obtained by $\overline{u}_a = u_a + u_{\overleftarrow{a}}$ where $u_a = 0$ if $a \notin A$ and

$$\overline{\tau}_a = \begin{cases} \tau_a & \text{if the orientation of arc is along } a \\ \tau_{\overleftarrow{a}} & \text{if the orientation of arc is along } \overleftarrow{a} \text{ or } a \notin A \end{cases}$$

It is to be noted that if there is no arc along the orientation of auxiliary network, then transit time of auxiliary arc is taken as the symmetric reversal of the opposite arc. Throughout the paper, our assumption is that the storage capacity of terminals are significant enough, i.e., $b_s = b_t \leq \infty$ and the intermediate nodes have finite storage capacity. The excess flow at intermediate nodes exists if and only if the sum of outgoing arc capacities of an intermediate node $v \in I$ is less than the sum of incoming arc capacities. Moreover, the storage capacity b_v of intermediate node $v \in I$ should be at least the sum of incoming arc capacities of v for unique solution. Otherwise, it provides an alternate optimal solution (see explanation of Figure 2).

2.2. Static Flow Model with Intermediate Storage. The static flow ψ with intermediate storage on a given two-way network $\mathcal{N} = (V, A, u, b, c, s, t)$ is the collection of non-negative arc flow functions $\psi_a : A \rightarrow \mathcal{R}^+$ and the excess flow functions $\psi_v : I \rightarrow \mathcal{R}^+$. The static flow model with intermediate storage is the network flow satisfying the conditions (2.1 - 2.4). As in [23], the mathematical formulation of the maximum static flow with intermediate storage, as a linear programming problem, is as follows.

$$(2.1) \quad \max |\psi| = \sum_{a \in \delta^{out}(s)} \psi_a = \sum_{a \in \delta^{in}(t)} \psi_a + \sum_{v \in I: b_v \geq 0} \psi_v$$

such that,

$$(2.2) \quad \sum_{a \in \delta^{in}(v)} \psi_a - \sum_{a \in \delta^{out}(v)} \psi_a = \psi_v \quad \forall v \in I$$

$$(2.3) \quad 0 \leq \psi_a \leq u_a \quad \forall a \in A$$

$$(2.4) \quad 0 \leq \psi_v \leq b_v \quad \forall v \in I$$

Equation (2.1) is an objective function that is to maximize the total outflow from the source which must be equal to the sum of inflow at sink and the total excess flow at intermediate nodes. Here, $|\psi|$ denotes the total value of the flow. The excess flow at intermediate nodes is represented by Equation (2.2) and

Equation (2.3) represents the boundedness of the flow on each arc. The left inequality in constraint (2.4) represents the non-conservation of flow where the right inequality shows the boundedness of the excess flow by the storage capacity of the node. For existence of unique solution, the storage capacity of intermediate node $v \in I$ must be at least the sum of incoming arc capacities to v . Mathematically, it is represented as $\sum_{a \in \delta^{in}(v)} u_a \leq b_v \quad \forall v \in I$. We define the cost c of static flow ψ associated with arc a as

$$c(\psi) = \sum_{a \in A} c_a \psi_a.$$

At the time of an emergency, the general assumption is that the flow towards the danger zone is almost empty and so we reverse the direction of arcs towards the safe zone (destination). After reversing the direction of arcs using orientation dependent cost, the contraflow model can be obtained replacing $a \in A$ by $a \in \bar{A}$ and u_a by \bar{u}_a . At last, the flow obtained in the auxiliary network is transformed to the original network.

2.3. Dynamic Flow Model with Intermediate Storage. The flow over time function Φ defined on a dynamic network \mathcal{N} with arc traversal time τ is the collection of non-negative arc flow function $\Phi_a : A \times \mathcal{T} \rightarrow \mathcal{R}^+$ and the excess flow function $\Phi_v : I \times \mathcal{T} \rightarrow \mathcal{R}^+$. The dynamic contraflow model with ODTT τ in \mathcal{N} allowing intermediate storage is a network flow satisfying the conditions (2.5 - 2.8). As in [23], the mathematical model for the maximum dynamic flow allowing the intermediate storage, with linear programming formulation, is as follows.

$$(2.5) \quad \max |\Phi| = \sum_{a \in \delta^{out}(s)} \sum_{\theta=0}^T \Phi_a(\theta) = \sum_{a \in \delta^{in}(t)} \sum_{\theta=\tau_a}^T \Phi_a(\theta - \tau_a) + \sum_{v \in I: b_v \geq 0} \Phi_v(T)$$

such that,

$$(2.6) \quad \sum_{a \in \delta^{in}(v)} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta - \tau_a) - \sum_{a \in \delta^{out}(v)} \sum_{\beta=0}^{\theta} \Phi_a(\beta) = \Phi_v(\theta) \quad v \in I, \theta \in \mathcal{T}$$

$$(2.7) \quad 0 \leq \Phi_a(\theta) \leq u_a \quad \forall a \in A, \theta \in \mathcal{T}$$

$$(2.8) \quad 0 \leq \Phi_v(\theta) \leq b_v \quad \forall v \in I, \theta \in \mathcal{T}$$

In given time horizon T , the objective function in Equation (2.5) refers to the maximization of the total flow out from the source which must be equal to the sum of inflow at sink and the total excess flow at intermediate nodes. The excess flow stored at intermediate nodes is represented by Equation (2.6). Similarly, the constraint in (2.7) represents the boundedness of the arc flow by its capacity at each time step θ . The left inequality of the constraint in (2.8) represents the non-conservation of flow and its right inequality represents the boundedness of the excess flow at the intermediate node by its storage capacity. The necessary (lower bound) and sufficient (upper bound) storage capacity of each intermediate node $v \in I$ is $\sum_{a \in \delta^{in}(v)} u_a \leq b_v \leq T \sum_{a \in \delta^{in}(v)} u_a \quad \forall v \in I$. The lower bound is essential for the existence of unique solution. The cost c of discrete dynamic flow Φ is defined as

$$c(\Phi) = \sum_{a \in A} c_a \sum_{\theta=0}^T \Phi_a(\theta).$$

For EACF problem with ODTT by allowing intermediate storage, the linear programming formulation is as follows.

$$(2.9) \quad \max |\Phi(\theta)| = \sum_{a \in \delta^{out}(s)} \sum_{\beta=0}^{\theta} \Phi_a(\beta) = \sum_{a \in \delta^{in}(t)} \sum_{\beta=\tau_a}^{\theta} \Phi_a(\beta - \tau_a) + \sum_{v \in I: b_v \geq 0} \Phi_v(\theta)$$

subject to the constraints (2.6 - 2.8).

As in static case, if we replace $a \in A$, τ_a and u_a in above model by $a \in \bar{A}$, $\bar{\tau}_a$ and \bar{u}_a , respectively, then the dynamic contraflow model can be obtained with orientation dependent transit times, where the traversal time of the arcs in auxiliary network is depending on its orientation.

While shifting the flow from the source, first priority is given to the sink because the assumption is that the sink is most appropriate place with respect to location and infrastructure. The priority of intermediate nodes is set with farther-in-distance-higher-in-priority. We fix the priority of nodes before the contraflow configuration because the alternative lane reversals in the network grows exponentially with the number of arc on the network which increases its computational complexity.

For the uniqueness of the solution, the minimum storage capacity of intermediate nodes should be at least the sum of incoming arc capacities i.e., $b_v \geq \sum_{a \in \delta^{in}(v)} u_a$. We now verify it by the following example.

Example 1. Consider the network presented in Figure 2 with unit cost/transit time on each arc except (s, w) . Let the cost of arc (s, w) be zero. The first priority is given to the sink node. Due to equal cost from the source, second priority is given to v or y , the third priority is for u or x and fourth priority is given to w . Firstly, we consider 2 units of storage capacity in each intermediate node. If the flow is sent to the sink through paths $s-u-v-w-t$ and $s-w-t$, then 7, 2, 2, 2, 0 and 1 units of flow can be stored at prioritized nodes t, v, y, u, x and w , respectively. Again, by sending the flow to the sink through paths $s-x-y-w-t$ and $s-w-t$, the amount of flow stored at prioritized nodes t, v, y, u, x and w are 7, 2, 0, 2, 0 and 2 units, respectively. So, in previous pattern of flow, 7, 4, 2, and 1 units are stored at first, second third and fourth prioritized nodes whereas 7, 2, 2, and 2 units are stored at respective prioritized nodes in case of later pattern, respectively. Thus, instead of a unique optimal solution it provides an alternate optimal solution. Secondly, if we consider the storage capacity of 10 units in each intermediate node, then 7, 5, 2 and 0 units of flow can be stored at first, second, third and fourth prioritized nodes, respectively, in either of the flow patterns.

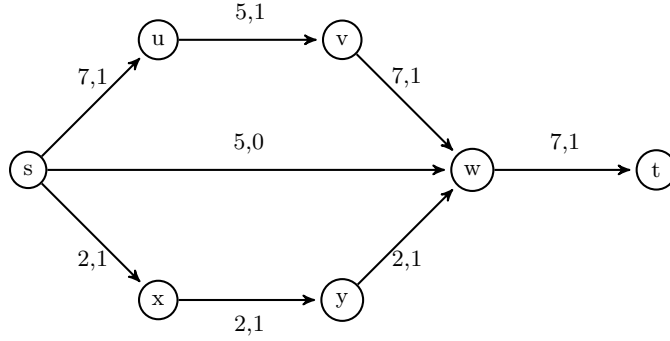


FIGURE 2. Network with capacity and unit cost/transit time on each arc.

3. MDCF WITH ODTT ALLOWING INTERMEDIATE STORAGE

In this section, we introduce the MDCF problem in which time to traverse through the arcs in auxiliary network are orientation dependent and the storage of flow at intermediate nodes is allowed. We present a polynomial time solution strategy to solve the problem.

Problem 1. For a given two-way dynamic network $\mathcal{N} = (V, A, u, b, \tau, s, t, T)$ with auxiliary network $\bar{\mathcal{N}}$, the MDCF problem with ODTT by allowing the storage of flow at intermediate nodes is to maximize the flow out from the source s which is to be transshipped to the sink t via $s-t$ paths together with the storage of maximum excess flow at the intermediate nodes with storage capacity $\sum_{a \in \delta^{in}(v)} \bar{u}_a \leq b_v \leq T \sum_{a \in \delta^{in}(v)} \bar{u}_a$ within the time horizon T , where the reversal of the arcs are made at time zero and transit times on auxiliary network are orientation dependent.

To solve the problem, the priority order of the nodes is fixed on the basis of farther-in-distance-higher-in-priority order as described in Pyakurel and Dempe [23], where distance of each intermediate node from the source is obtained by using Dijkstra's algorithm ([8]) by considering transit time as cost. That is, $\forall v_1, v_2 \in I$

if $d_{(s,v_1)} > d_{(s,v_2)}$, then v_1 is more relevant and highly prioritized than v_2 and is denoted by $v_1 \succ v_2$. We then transform the given network into an auxiliary network $\bar{\mathcal{N}}$ with orientation dependent transit times. The flow is sent from the source with first priority to the sink and then successive order of prioritized intermediate nodes.

For each prioritized node $v \in I$, we create the dummy port v' with capacity $u_{(v,v')} = b_v = b_{v'}$, where $u_{(v,v')}$ is the arc capacity of dummy arc (v, v') . The transit time of dummy arc is $\tau_{(v,v')} = 0$. Every dummy port v' has the same priority order as v has. The modified auxiliary network $\bar{\mathcal{N}}' = (V', \bar{A}', \bar{\tau}, \bar{u}, b, s, D)$ is constructed having single source s and multiple sink $D = \{t \cup \{v'\}\}$, where $\{v'\}$ is the collection of dummy ports. Here, $V' = V \cup \{v'\}$, and $\bar{A}' = \bar{A} \cup \{(v, v')\}$. For an instance, if $t \succ v_1 \succ \dots \succ v_r$ be priority order of intermediate nodes and sink, then $D = \{t = v'_0, v'_1, \dots, v'_r\}$. We construct a new modified network by introducing a super sink t^* having sufficiently large (infinite) storage capacity and join each $v' \in D$ by an arc with zero transit time and infinite capacity (See Figure 3). Using lexicographic maximum flow, the maximum dynamic flow with intermediate storage in an auxiliary network is obtained as in [23], where the reversal of arcs are orientation dependent. Finally, flow in the auxiliary network is transformed into the original network.

We present a polynomial time algorithm to solve the problem as follows.

Algorithm 1: MDCF algorithm with ODTT allowing the intermediate storage

Input : Given two-terminal contraflow network \mathcal{N} .

Output: MDCF with ODTT allowing intermediate storage.

- (1) Fix the priority order of intermediate nodes using Dijkstra's algorithm ([8]).
 - (2) Construct an auxiliary network $\bar{\mathcal{N}}$ of \mathcal{N} using orientation dependent transit times.
 - (3) Compute the maximum dynamic flow in $\bar{\mathcal{N}}$ by allowing the storage of flow at intermediate nodes using [23].
 - (4) Decompose the flow along s - t paths and cycles, and remove the flows in cycles.
 - (5) An arc \overleftarrow{a} is reversed with orientation dependent transit times if and only if the flow along arc a is greater than its capacity or if there is a non-negative flow along arc $a \notin A$.
 - (6) By removing the dummy ports and arcs, transform the solution to the original network.
-

Theorem 3.1. *Algorithm 1 provides the feasible solution to MDCF problem with ODTT by allowing the storage of flow at intermediate nodes optimally.*

Proof. At first, we show the feasibility and then the optimality of Algorithm 1. Steps (1) and (2) are the use of Dijkstra's algorithm and the construction of an auxiliary network, respectively, which are feasible. In Step 3, finding the maximum dynamic flow with intermediate storage is feasible by [23]. As steps (4), (5) and (6) are construction of cycle free paths, condition of arc reversals and transformation of solution, which are feasible. So, Algorithm 1 provides the feasible solution. Next, the optimality of algorithm is dominated by the optimality of Step (3), which is optimal as proven by Pyakurel and Dempe [23]. So, Algorithm 1 solves the MDCF with ODTT by allowing the storage of flow at intermediate nodes optimally. \square

Theorem 3.2. *MDCF problem with ODTT can be solved by Algorithm 1 allowing the intermediate storage in polynomial time complexity.*

Proof. The time complexity of Dijkstra's algorithm in Step 1 is $O(m \log n)$ and that of shorting algorithm is $O(n \log n)$. Steps (2) and (4) of Algorithm 1 can be obtained in linear time. Step (3) calculates a priority based maximum dynamic flow in s - D network in $O(n \times MCF(n, m))$ time, where $MCF(n, m)$ represents the time complexity of single source single sink minimum cost flow in original network, which can be solved within $O(m \log n(m + n \log n))$ time. So, Algorithm 1 solves the MDCF problem with ODTT by allowing the intermediate storage in polynomial time complexity. \square

Example 2. Consider the dynamic contraflow network given in Figure 1(i) with time horizon $T = 4$. Using Dijkstra's algorithm [8], the priority ordering is set as $t \succ y \succ x$. Four different auxiliary networks with ODTT are presented in Figure 1(ii)-(v). Figure 1(ii) represents an auxiliary network with orientation along (x, y) and (x, t) which sends 19, 8 and 13 units of flow at t, y and x in $T = 4$, whereas Figure 1(iii) is with orientation along (y, x) and (x, t) which sends 17, 4 and 19 units of flow at t, y and x in $T = 4$, respectively. Similarly, Figure 1(iv) is obtained by taking orientation in auxiliary network along (x, y) and (t, x) which sends 10, 8 and 22 units of flow at t, y and x in $T = 4$, whereas Figure 1(v) is with orientation along (y, x) and (t, x) which sends 8, 4 and 28 units of flow at t, y and x in $T = 4$, respectively. In each orientation, total 40 units of flow is pushed from source in given time horizon but Figure 1(ii) is only the network which pushes maximum flow at sink and the prioritized farthest intermediate node. Thus, the best solution with ODTT is obtained from Figure 1(ii), whose new modified network with super sink is presented in Figure 3. The detailed illustration of flow in this network is given in Table 1 below.

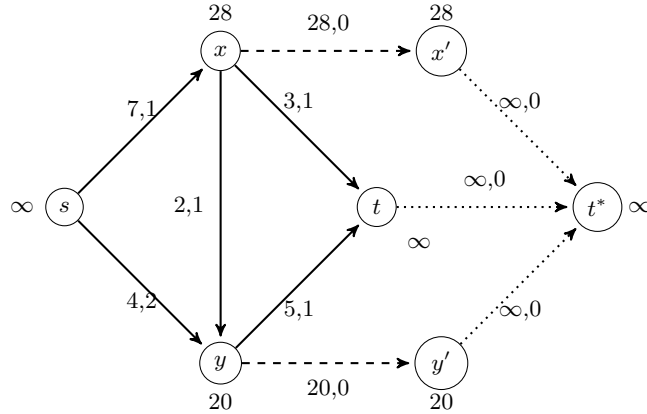


FIGURE 3. New modified network of Figure 1(ii).

TABLE 1. MDCF with ODTT allowing intermediate storage at time θ .

Path	Start time at s	flow at			Reaching time at last node
		x	y	t	
$s-x-t$	$\theta = 0$	2	×	3	$\theta = 2$ at t
	$\theta = 1$	2	×	3	$\theta = 3$ at t
	$\theta = 2$	2	×	3	$\theta = 4$ at t
	$\theta = 3$	7	×	×	$\theta = 4$ at x
$s-y-t$	$\theta = 0$	×	0	4	$\theta = 3$ at t
	$\theta = 1$	×	0	4	$\theta = 4$ at t
	$\theta = 2$	×	4	×	$\theta = 4$ at y
$s-x-y-t$	$\theta = 0$	0	1	1	$\theta = 3$ at t
	$\theta = 1$	0	1	1	$\theta = 4$ at t
	$\theta = 2$	0	2	×	$\theta = 4$ at y
	Total	13	8	19	Total=40

4. EACF WITH ODTT ALLOWING INTERMEDIATE STORAGE

We aim to introduce an EACF problem in a two-terminal series-parallel network by allowing the storage of flow at intermediate nodes and adopting the orientation dependent transit times in auxiliary network. To solve the problem, we present a polynomial time solution strategy.

Problem 2. For a given two-terminal series-parallel network $\mathcal{N} = (V, A, u, b, \tau, s, t, T)$ with auxiliary network $\bar{\mathcal{N}}$, an EACF problem with ODTT permitting the storage of flow at the intermediate nodes is to maximize the flow leaving the source s at each time step θ , that is to be transshipped to the sink t via s - t paths together with the storage of maximum excess flow at the intermediate nodes with storage capacity $\sum_{a \in \delta^{in}(v)} \bar{u}_a \leq b_v \leq T \sum_{a \in \delta^{in}(v)} \bar{u}_a$ within time horizon T , where the reversal of the arcs are made at time zero and transit times on auxiliary network are orientation dependent.

To solve the problem, we first transform the given s - t network into s - D network by creating dummy ports as described in Section 3. Considering t^* as the super sink and joining it to each node in D with infinite capacity and zero transit time, it is reduced to the single source single sink s - t^* network. As in Pyakurel and Dempe [24], the minimum cost circulation flow with time bound of Ruzika et al. [28] provides the solution of EACF problem with ODTT by allowing the intermediate storage of the flow in polynomial time complexity, where transit times are orientation dependent. Hereafter, we present an algorithm which provides the solution to an EAF Problem 2 with ODTT by allowing the intermediate storage.

Algorithm 2: EACF algorithm with ODTT allowing the intermediate storage

Input : Given two-terminal series-parallel contraflow network \mathcal{N} .

Output: EACF with ODTT by allowing the intermediate storage.

- (1) Fix the priority order of nodes and construct an auxiliary network $\bar{\mathcal{N}}$ of \mathcal{N} using orientation dependent transit times.
 - (2) Compute earliest arrival flow with intermediate storage in $\bar{\mathcal{N}}$ by using [24] within the time bound of [28].
 - (3) Decompose the flow along s - t paths and cycles, and remove the flows in cycles.
 - (4) An arc \overleftarrow{a} is reversed with orientation dependent transit times if and only if the flow along arc a is greater than its capacity or if there is a non-negative flow along arc $a \notin A$.
 - (5) By removing the dummy ports and arcs, transform the solution to the original network.
-

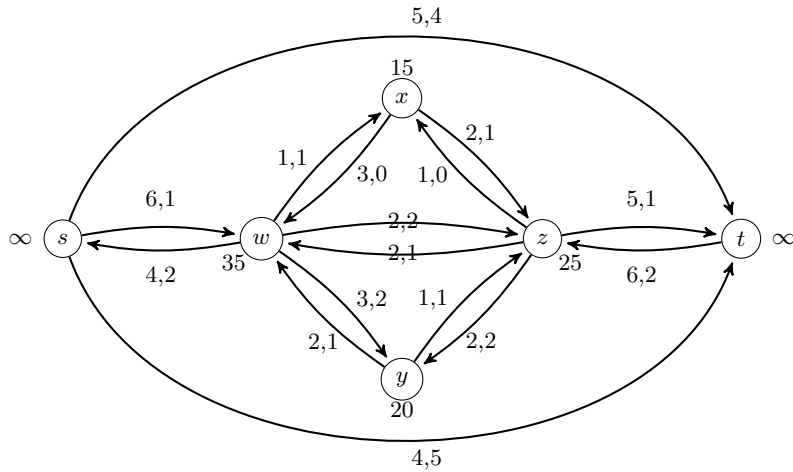
Theorem 4.1. The optimal solution to the EACF problem with ODTT by allowing the intermediate storage can be obtained by using Algorithm 2.

Proof. The feasibility and optimality of Steps (1), (3), (4) and (5) can be obtained as in Theorem 3.1. In step (2), we compute the earliest arrival flow in auxiliary network with orientation dependent transit times by using algorithm of Pyakurel and Dempe [24] with time bound of Ruzika et al. [28], each providing optimal flow. So Algorithm 2 provides optimal solution to EACF with ODTT by allowing the intermediate storage. \square

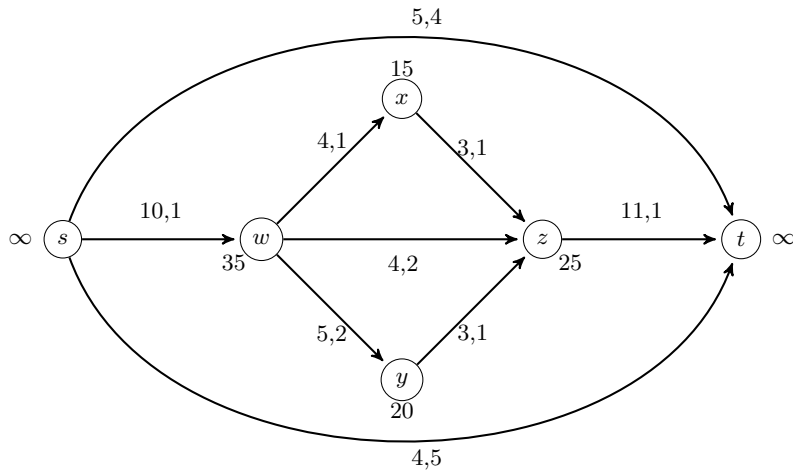
The computational time bound of Step (2) determines the overall time complexity of Algorithm 2. The minimum cost flow problem in series-parallel graph can be solved in $O(nm + m \log m)$ time for single source sink network. So EACF problem with ODTT by allowing intermediate storage can be solved in $O(n \times (nm + m \log m))$ time.

Theorem 4.2. Algorithm 2 solves the EACF problem with ODTT by allowing the intermediate storage in polynomial time.

Example 3. Consider a two-terminal series-parallel contraflow network that is given in Figure 4(i), where numbers aside the nodes represent the storage capacity of nodes. The priority ordering of nodes with respect to the distance is $t \succ z \succ y \succ x \succ w$ (as the shortest distance of z and y from s are same, priority of these two nodes may interchange). The auxiliary network with orientation dependent transit times is constructed in Figure 4(ii). Let the time horizon be $T = 5$. Then, the total amount of earliest arrival flow at t , z , y , x and w in time horizon $T = 5$ are 30, 9, 2, 7 and 16 units, respectively.



(i) Two-way series-parallel network with capacity, transit time on arc



(ii) Auxiliary network with capacity, transit time on arc

FIGURE 4. (i) represents a two-terminal series-parallel contraflow network and (ii) represents an auxiliary network of (i) with ODTT.

In series-parallel graph, every cycle in the residual network has a non-negative cycle length, which is a main advantage of this graph structure. Because of this, the minimum cost circulation flow problem introduced by Ford and Fulkerson [10] for the maximum dynamic flow problem can be solved in the auxiliary network \bar{N} . The temporally repeated $s-t$ flow, thus obtained is an optimal solution to the EACF problem on a two-terminal series-parallel graph in polynomial time. So, the polynomial solution of EACF problem with ODTT by allowing the intermediate storage is possible by using Pyakurel and Dempe [24] and applying the ODTT on \bar{N} . Moreover, for general networks, pseudo-polynomial time solution strategy exists to solve EACF problem by using time expanded network, [6, 25, 29] without permitting the intermediate storage. The EACF problem with ODTT by allowing the storage of flow at intermediate nodes in general network can be obtained by using algorithm of Hoppe and Tardos [15] in modified $s - D$ network.

5. CONCLUSION

At the time of evacuation, improvement of flow values and reduction of the time horizon by the reversal of arcs is one of the best approaches. At the meantime, settlement of flow at intermediate nodes can be a milestone to shift the maximum evacuees from the danger zone (source). In literature, the maximum static and maximum dynamic contraflow problems with and without intermediate storage of flow have been studied in two terminal general network. Similarly, earliest arrival contraflow problem with intermediate storage has been studied in two-terminal series-parallel graph. All of these contraflow problems have been acquired symmetric transit times on the anti-parallel arcs.

In this paper, we present the flow models with asymmetric transit times on anti-parallel arcs by allowing the storage of flow at intermediate nodes. By adopting the orientation dependent transit times on the anti-parallel arcs, we introduce the maximum dynamic contraflow and earliest arrival contraflow problems. We also present the polynomial time algorithms to solve the problems. Though the priority order of the nodes in our solution strategy is on the basis of distance before the contraflow configuration, our algorithms can be used for any other priority order.

Acknowledgments The first author (Durga Prasad Khanal) thanks to the German Academic Exchange Service - DAAD for Research Grants - Bi-nationally Supervised Doctoral Degrees/Cotutelle, 2021/22 and the second author (Urmila Pyakurel) thanks to the Alexander von Humboldt Foundation for Digital Cooperation Fellowship (August 1, 2021 – January 31, 2022).

REFERENCES

- [1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithm and Applications*, Prentice Hall, Englewood Cliffs, 1993.
- [2] J.E. Aronson, A survey of dynamic network flows, *Annals of Operations Research*, vol. 20, pp. 1-66, 1998.
- [3] A. Arulsevan, *Network Model for Disaster Management*, PhD thesis, University of Florida USA, 2009.
- [4] N. Baumann, E.M. Köhler, Approximating earliest arrival flows with flow-dependent transit times, *Discrete Applied Math*, vol. 155, pp. 161-171, 2007.
- [5] T.N. Dhamala, S.P. Gupta, D.P. Khanal, U. Pyakurel, Quickest multi-commodity flow over time with partial lane reversals, *Journal of Mathematics and statistics*, vol. 16, pp. 198-211, 2020.
- [6] T.N. Dhamala, U. Pyakurel, Earliest arrival contraflow problem on series-parallel graphs, *International Journal of Operations Research*, vol. 10, pp. 1-13, 2013.
- [7] T.N. Dhamala, U. Pyakurel, S. Dempe, A critical survey on the network optimization algorithms for evacuation planning problems, *International Journal of Operations Research*, vol. 15 no.3, pp. 101-133, 2018.
- [8] E.W. Dijkstra, A note on two problems in connection with graph, *Numer. Math.* vol. 1, no. 1, pp. 269-271, 1959.
- [9] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics*, vol. 8, pp. 399-404, 1956.
- [10] L.R. Ford, D.R. Fulkerson, *Flows in Networks*. Princeton University Press, Princeton, New jersey, 1962.
- [11] D. Gale, Transient flows in networks, *Michigan Mathematical Journal*, vol. 6, pp. 59-63, 1959.
- [12] S.P. Gupta, U. Pyakurel, T.N. Dhamala, Generalized dynamic contraflow with non-symmetric transit times, *American Journal of Computational and Applied Mathematics*, vol. 11, no. 1, pp. 12-17, 2021.
- [13] S.P. Gupta, U. Pyakurel, T.N. Dhamala, Network flows with arc reversals and non-Symmetric transit times, *American Journal of Mathematics and Statistics*, vol. 11, no. 2, pp. 27-33, 2021.
- [14] H.W. Hamacher, S.A. Tjandra, Mathematical modeling of evacuation problems: a state of the art, *Pedestrian and Evacuation Dynamics*, pp. 227-266, 2002.

- [15] B. Hoppe, E. Tardos, The quickest transshipment problem, *Mathematics of Operations Research*, vol. 25, pp. 36-62, 2000.
- [16] D.P. Khanal, U. Pyakurel, T.N. Dhamala, Maximum Multicommodity Flow with Intermediate Storage, *Mathematical Problems in Engineering, Hindawi*, 2021, DOI: <https://doi.org/10.1155/2021/5063207>
- [17] S. Kim, S. Shekhar, Contraflow network reconfiguration for evacuation planning: a summary of results, *In: Proceedings of 13th ACM Symposium on Advances in Geographic Information Systems GIS*, vol. 05, pp. 250–259, 2005.
- [18] S. Kim, S. Shekhar, M. Min, Contraflow transportation network reconfiguration for evacuation route planning, *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1115-1129, 2008.
- [19] B. Kotnyek, *An annotated overview of dynamic network flows*, Rapport de recherche 4936, INRIA Sophia Antipolis, 2003.
- [20] I.S. Kotsireas , A. Nagurney, P.M. Pardalos, Dynamics of disasters-algorithmic approaches and applications, *Springer Optimization and Its Applications*, 2018.
- [21] E. Minieka, Maximal Lexicographic and Dynamic Network flows, *Operations Research*, vol. 21, pp. 517-527, 1973.
- [22] H.N. Nath, U. Pyakurel, T.N. Dhamala, Network reconfiguration with orientation dependent transit times, *International Journal of Mathematics and Mathematical Sciences, Hindawi*, Vol. 2021, Article ID 6613622, 2021, DOI: <https://doi.org/10.1155/2021/6613622>.
- [23] U. Pyakurel, S. Dempe, Network Flow with Intermediate Storage: Models and Algorithms, *SN Operations Research Forum*, 2020, DOI:10.1007/s43069-020-00033-0.
- [24] U. Pyakurel, S. Dempe, Universal Maximum Flow with Intermediate Storage for Evacuation Planning, *In: Kotsireas I.S., Nagurney A., Pardalos P.M., Tsokas A. (eds) Dynamics of Disasters. Springer Optimization and Its Applications*, vol. 169, Springer, Cham. 2021, https://doi.org/10.1007/978-3-030-64973-9_14.
- [25] U. Pyakurel, T.N Dhamala, Models and algorithms on contraflow evacuation planning network problems, *International Journal of Operations Research*, vol. 12, pp. 36-46, 2015.
- [26] U. Pyakurel, S.P. Gupta, D.P. Khanal, T.N. Dhamala, Efficient algorithms on multi-commodity flow over time problems with partial lane reversals, *International Journal of Mathematics and Mathematical Sciences, Hindawi*, 2020, DOI: <https://doi.org/10.1155/2020/2676378>.
- [27] S. Rebennack, A. Arulselvan, L. Elefteriadou, P.M. Pardalos, Complexity analysis for maximum flow problems with arc reversals, *Journal of Combinatorial Optimization*, vol. 19, pp. 200-216, 2010.
- [28] S. Ruzika, H. Sperber, M. Steiner, Earliest arrival flows on series- parallel graphs *Networks*, vol. 10, pp. 169–173, 2011.
- [29] W.L. Wilkinson, An algorithm for universal maximal dynamic flows in a network, *Operations Research*, vol. 19, pp. 1602-1612, 1971.



Efficient Algorithms for Abstract Flow with Partial Switching

Durga Prasad Khanal¹ · Urmila Pyakurel²  · Tanka Nath Dhamala² · Stephan Dempe³

Received: 29 September 2021 / Accepted: 27 August 2022

© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2022

Abstract

The tragic circumstances caused by natural or human-made (unexpected human or technological errors) hazard such as earthquakes, floods, glaciers, fires or industrial explosions causing significant physical damages, loss of lives or destruction of environment as well as economic and social life of people are known as disasters. Planned evacuation is essential to save the maximum number of evacuees in minimum time, which also helps in minimize losses. Due to mass dispatch (movement) of people aftermath of disaster, traffic scenario at the intersection of roads may create the disappointing situation if the vehicles have to wait for hours to cross the intersection. The main reason behind this is the lack of crossing elimination. In this paper, we discuss the partial switching property on an abstract network, in which crossing effect of roads is eliminated to transship optimal flow of evacuees. Due to the switching property, crossing of the flows at the intersections is diverted to non-crossing sides which can be a milestone to smooth the flows during evacuation. We present polynomial time solution procedures to solve abstract maximum static and dynamic flow problems with partial switching of paths. We also introduce the abstract quickest flow and quickest contraflow problems with partial switching and present polynomial time algorithms to solve the problems. For disaster management, maximum, quickest and contraflow problems on partially switched paths play an important role as the flow on a path system without crossing effect is very essential during evacuation process.

Keywords Abstract network · Partial switching · Evacuation planning · Maximum flow · Quickest flow · Contraflow

This article is part of the Topical Collection on *Dynamics of Disasters*

✉ Urmila Pyakurel
urmilapyakurel@gmail.com

¹ Saraswati Multiple Campus, Tribhuvan University, Kathmandu, Nepal

² Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal

³ Faculty of Mathematics and Computer Science, TU Bergakademie Freiberg, Freiberg, Germany

1 Introduction

Disasters, both natural and human-made, are uncertain and can cause terrible disruptions, which in turn result in damages and casualties. According to United Nations International Strategy for Disaster Reduction (UNISDR) [33], “a disaster is a sudden, calamitous event that seriously disrupts the functioning of a community or society and causes human, material and economic or environmental losses that exceed the community’s or society’s ability to cope using its own sources. Though often caused by nature, disasters can have human origins”.

In network optimization, an evacuation scenario is observed by the network topology where paths are represented by the chain of arcs and their crossings by the nodes. On the basis of representation, a network can be classified in two ways: the node-arc system, termed as classical network and the path-element system, termed as abstract network. In this paper, we deal the problems on an abstract network. Network with capacitated elements and linearly ordered subset of elements, called paths, is an abstract network. Each element is equipped with integral movement capacity which transships the flow from an element to its adjacent element. In this network, each path must satisfy the switching property: when two paths cross at an element then there must be a path that is a subset of the first path up to the crossing element and a subset of the second path after the crossing element.

By reviewing the first proof of max-flow-min-cut theorem of Ford and Fulkerson [6], Hoffman [12] introduced the concept of abstract flow in terms of paths rather than on arcs. McCormick [24] provided a polynomial time algorithm by using an oracle where input is an arbitrary subset of elements whose output is either a path contained in that subset or states that no such path exists. He used the augmenting path structure satisfying the complementary slackness condition. Using the additional attribute of weight on paths, Martens and McCormick [23] extended the result of [24] in more general case. For single as well as multi-commodity flows, Martens [22] presented unsplittable and k -splittable abstract network flows in his PhD thesis. By solving weighted abstract flow problem and constructing a temporally repeated flow from its solution, Kappmeier et al. [13] have shown that the abstract maximum flow over time can be obtained in polynomial time complexity. The polynomial time algorithm for lexicographic abstract maximum flow and its use in proving the existence of abstract earliest arrival flow can be found in PhD thesis of Kappmeier [14].

At the time of disasters, flow improvement with contraflow configuration is one of the best techniques in which unused arcs are reversed towards the destination to increase the outbound capacity of paths towards the destination and reduce the evacuation time. Although authors in Bretschneider and Kimms [3] allowed the movement of rescuer towards danger zone if it leads to faster evacuation, but as in Vogiatzis et al. [32], we do not allow any movement towards the risky zone. That is, in two-way network topology, our assumption is that paths towards the source element (danger zone) are almost empty. The optimal use of such paths by reversing towards the sink element (safe zone) can be an important tool at the time of disaster, because every individual desires to leave the danger

zone as quickly and efficiently as possible. The notion of contraflow technique was introduced by Kim and Shekhar [18]. They have proved the NP-completeness of the contraflow problem and presented a heuristic to solve large-scale evacuation instances. Pyakurel et al. [28] introduced the concept of continuous maximum abstract contraflow problem for classical network and presented polynomial time algorithms for static as well as dynamic cases. Similarly, by saving unused capacities of the elements, Pyakurel et al. [31] introduced the partial contraflow approach in abstract network and presented efficient algorithms for static, lexicographically maximum static, maximum dynamic and earliest arrival partial contraflow problems.

In classical network, Pyakurel and Dempe [26] introduced the maximum dynamic contraflow problem with intermediate storage and presented a polynomial time algorithm to solve the problem. They also have investigated universal maximum dynamic flow with intermediate storage and presented efficient algorithms in general as well as two-terminal series parallel networks in [27]. Similarly, Khanal et al. [16] presented pseudo-polynomial time algorithms for maximum multi-commodity flow and contraflow problems with intermediate storage where contraflow is configured for the network with symmetric as well as asymmetric transit times on arcs. For the network with asymmetric transit times on anti-parallel arcs, maximum dynamic contraflow problem in a general network and earliest arrival contraflow problem in two-terminal series-parallel network allowing the intermediate storage of flow can be found in [17]. Recently, Pyakurel et al. [29] presented the solution strategy of maximum static, lexicographic maximum static and maximum dynamic flow problems in an abstract network topology by incorporating the storage of flow at intermediate shelters.

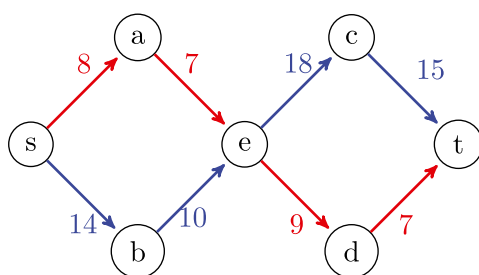
Quickest flow problems are inverse to maximum flow problems, in which a given amount of flow is sent towards safety in the smallest time possible. For the prior or the post disaster management, mathematical models and their solution strategies with objective of minimum clearance time are very realistic because every individual intend to reach the safe places as quickly as possible. Using binary search to the maximum flow solution of Ford and Fulkerson [7], Burkard et al. [4] provided the first polynomial time bound for the quickest flow problem in classical network. They also presented strongly polynomial time bound for this problem by incorporating the parametric search to the minimum cost flow problem. Lin and Jaillet [19] extended the cost-scaling algorithm of Goldberg and Tarjan [8] which solves the quickest flow problem within the same bound as the min-cost flow problem. Pyakurel et al. [30] used this cost-scaling algorithm to solve the quickest contraflow problem. By using length bound approximation and condensed time expanded network, Dhamala et al. [5] solved the quickest multi-commodity contraflow problem with partial lane reversals.

Minimizing the conflict of flows at the intersections of roads and smoothing the flow during evacuation are important tasks. Andreas and Smith [2] introduced the concept of evacuation tree to minimize the conflicts at intersections and ensure the seamlessness of the evacuation process. Similarly, for the fast and safe evacuation, Achrekar and Vogiatzis [1] introduced the time-dynamic evacuation model that aims to maximize a reward function. To achieve the seamlessness in

evacuation, they have constructed an evacuation tree with two improving policies of contraflow and divergence, where vehicles are evacuated using a single path to safety at each intersection. Purba et al. [25] extended the concept of evacuation tree for the evacuation routing problem of alternative fuel vehicles.

By including maximum flow, earliest arrival flow, quickest path and quickest flow problems, Hamacher and Tjandra [9] published a survey for evacuation optimization models. They also distinguished between microscopic and macroscopic approach to the evacuation modeling. The mathematical programming application in the field of evacuation under disaster can be found in Liu et al. [20]. Lu et al. [21] proposed a new capacity constrained routing algorithm for evacuation planning problem and presented a heuristic algorithm which produces a sub-optimal solution for evacuation planning problem without using time-expanded networks. Bretschneider and Kimms [3] presented a basic evacuation model that restructures the traffic routing for the case of an evacuation with regard to a safe evacuation process and a short evacuation time, and developed heuristics to solve the evacuation model. To tackle the large-scale evacuation, Vogiatzis et al. [32] proposed the clustering technique to divide problem into smaller sub-problems. By including the danger factor of each nodes, their model restricts the flow from low danger factor to high danger factor node. Similarly, Hassan and Van [10, 11] presented models and algorithms for large-scale zone-based evacuation planning.

In this paper, we introduce partial switching property on an abstract network to eliminate the crossing effect, which indeed minimizes the delay on flow transmission due to intersection of roads. For example, consider a network given in Fig. 1, where s and t are source and sink elements and rest of the elements (i.e., a, b, c, d and e) are intermediate elements. The numbers between the elements represent the movement capacity of path segments from an element to its adjacent one. By using max-flow-min-cut, maximum static flow in classical network is 17 units which can be transshipped from s to t using either of four paths $P_1 = s - a - e - d - t$, $P_2 = s - b - e - c - t$, $P_3 = s - a - e - c - t$, and $P_4 = s - b - e - d - t$. Here, two paths P_1 and P_2 are crossing at element e . For the complete switching property on abstract network, flow from s reaching at crossing element e along P_1 is switched to P_2 after e and flow reaching at e along P_2 is switched to P_1 after e . Thus, we have two completely switched paths P_3 and P_4 that can transship only 14 units of flow from source element to sink element. Now, by the help of residual network, we can find a residual path P_2 in abstract



Classical maximum flow = 17

Abstract maximum flow with complete switching = 14

Abstract maximum flow with partial switching = 17

Fig. 1 Comparison between classical and abstract flows

network with residual capacity of 3 units so that abstract flow with partial switching can transship 17 units of flow from s to t .

If more than two paths have crossing effect at an element, then we first use the partial switching in two paths. If still there exists the crossing effect on the resulting flow of switched paths with any other paths, then partial switching is successively applied unless the crossing effect is eliminated.

The road traffic at disasters can be more chaotic if proper route plans are not applied. To reduce the congestion due to the crossing effect, abstract flow with partial switching can be a milestone, which eliminates the crossing effect but allows the merging within the residual capacity. It helps to smooth the flow in allocated path system with optimal flow value. Not only in the disasters, this flow model can be applicable to mitigate the rush-hour traffic congestion. Since the delay on the flow transmission due to crossing is not considered in classical network, which is a major and uncertain factor for the flow transmission, the flow obtained from an abstract network with partial switching of paths is more practical than in a classical network.

Our contribution in this paper is to introduce the concept of partial switching on abstract network topology, where crossing effect of the paths is eliminated by using the residual path together with completely switched paths which may violate the complete switching property. We discuss the maximum static and maximum dynamic flow problems and the flow models with partial switching of paths in abstract network. We also present the polynomial time solution strategy of abstract maximum flow problem with partial switching in brief whose detailed illustration is presented in [15]. We present polynomial time algorithms for abstract quickest flow and quickest contraflow problems with partial switching. To the best of our knowledge, abstract quickest flow and contraflow problems with partial switching are introduced for the first time.

We organize the paper as follows. Section 2 provides the basic definitions and mathematical formulations of the flow models whereas in Sect. 3, we discuss polynomial time solution procedures to solve the abstract maximum static and maximum dynamic flow problems with partial switching. Abstract quickest flow problem and its polynomial time algorithm is presented in Sect. 4. Similarly, in Sect. 5, we present the abstract quickest contraflow problem and its polynomial time solution strategy. The paper is concluded in Sect. 6.

2 Mathematical Formulation of Flow Models

In this section, we present the basic mathematical notations. We also present abstract static as well as abstract dynamic flow models with partial switching property.

2.1 Basic Notations

Let $\mathcal{N} = (E, \mathcal{P})$ be a network topology with finite set of elements E and the collection of paths

$$\mathcal{P} = \{P \subseteq E : P \text{ has a linear order } <_P \text{ of elements in } P\} \subseteq 2^E.$$

Here, \mathcal{P} represents the set of all source-sink (s - t) paths P . Each element $e \in E$ has the non-negative integral movement capacity $u_e : E \times E \rightarrow \mathbb{Z}^+$ which is used to send the flow from the element e to its adjacent element. The order of elements in the path $P \in \mathcal{P}$ is denoted by $<_P$. We say that a is left of e on P if $a <_P e$ and right of e if $a >_P e$. Similarly, $e \in P$ is said to be leftmost or first (rightmost or last) element of P if there does not exist a in P such that $a <_P e$ ($a >_P e$). For s - t path P , source s is the leftmost element and sink t is the rightmost element. The set of intermediate elements is denoted by $E_I = E \setminus \{s, t\}$.

Network $\mathcal{N} = (E, \mathcal{P})$ is an abstract network if it satisfies the switching property: $\forall P, Q \in \mathcal{P}$ and intermediate element $e \in P \cap Q$, $\exists R \in \mathcal{P}$ such that $R \subseteq P \times_e Q$ where,

$$P \times_e Q = \{a \in P : s \leq_P a \leq_P e\} \cup \{a \in Q : e \leq_Q a \leq_Q t\}.$$

Similarly, we can define $R \subseteq Q \times_e P$. We use the notations

$$P_{[s \rightarrow e]} = \{a \in P : s \leq_P a \leq_P e\} \quad \text{and} \quad P_{[e \rightarrow t]} = \{a \in P : e \leq_P a \leq_P t\}$$

to represent the elements on path P from source s up to e and that begins from e up to sink t , respectively. Similarly, the elements on path P that are left of e and right of e are represented by

$$P_{[s \rightarrow e)} = \{a \in P : s \leq_P a <_P e\} \quad \text{and} \quad P_{(e \rightarrow t]} = \{a \in P : e <_P a \leq_P t\},$$

respectively. If P and Q are two paths both containing e_1 and e_2 , then it is possible to have $e_1 <_P e_2$ but $e_1 >_Q e_2$.

Due to the switching effect, abstract flow may not be equal the classical (general) network flow (see the comparison of flow values in Fig. 1). In this case, we construct a residual network to find an additional s - t path in original network which eliminates the crossing effect but can violate the complete switching property. Let $x_P : P \rightarrow \mathbb{R}^+$ be a static abstract path flow such that $x_P = \min\{x_e : e \in P\}$. Then the residual movement capacity of an element $e \in E$ is $\tilde{u}_e = u_e - x_e$. An element e is said to be saturated with respect to x if $x_e = u_e$. The partial switching property is defined as follows: $\forall P, Q \in \mathcal{P}$ and intermediate element $e \in P \cap Q$, $\exists R_i \in \mathcal{P}$, $i = 1, 2, 3$ such that

$$R_1 \subseteq P \times_e Q = \{a \in P : s \leq_P a \leq_P e\} \cup \{a \in Q : e \leq_Q a \leq_Q t\}.$$

$$R_2 \subseteq Q \times_e P = \{a \in Q : s \leq_Q a \leq_Q e\} \cup \{a \in P : e \leq_P a \leq_P t\}.$$

and if saturated elements of R_1 and R_2 lie in different sides of the crossing element e , then $\exists R_3$, a residual path, such that

$$R_3 \subseteq P : \tilde{u}_P > 0 \quad \text{or} \quad R_3 \subseteq Q : \tilde{u}_Q > 0,$$

where, $\tilde{u}_P = \min\{\tilde{u}_e : e \in P\}$. Here, R_1, R_2 and R_3 represent three modes of paths with $R = \{UR_i\}$. Clearly, $R \subseteq \mathcal{P}$.

In Fig. 1, paths P_3, P_4 and P_2 stands for the partially switched paths of mode R_1, R_2 and R_3 , respectively. If P be a s - t path without crossing effect, then $P = P \times_e P = R_i, \forall e \in P$. We consider that the movement capacity (i.e., outgoing movement capacity) of source and intermediate elements are finite and that of sink is zero (i.e., $u_t = 0$). Moreover, the incoming and outgoing movement capacities of source and sink, respectively, are zero except for contraflow network.

2.2 Abstract Static Flow Model with Partial Switching

Let $\mathcal{N} = (E, \mathcal{P})$ be an abstract network with path flow $x_{R_i} : R_i \rightarrow \mathbb{R}^+, R_i \in R \subseteq \mathcal{P}, i = 1, 2, 3$, satisfying the partial switching property. Every path flow x_{R_i} with partial switching induces a flow through each element, denoted by $\sum_{R_i \in \mathcal{P}: e \in R_i} x_{R_i} = x_e$. A path flow x is feasible if and only if $0 \leq x_e \leq u_e$ for all $e \in E$. Let A_e and B_e be the sets of outgoing paths from e and incoming paths into e , respectively. We denote $x_e^{out} = \sum_{R_i \in A_e} x_{R_i}$ and $x_e^{in} = \sum_{R_i \in B_e} x_{R_i}$ as the total outflow from e and the total inflow into e , respectively. Let $c_e : E \times E \rightarrow \mathbb{Z}^+$ be the cost of transmission of flow per unit from e to its right element so that $c_{R_i}(x) = \sum_{e \in R_i} c_e x_e$.

The linear program for abstract static network flow with partial switching is

$$\max \sum_{i=1}^3 \sum_{R_i \in R} x_{R_i} \tag{1}$$

$$\text{s.t. } \sum_{i=1}^3 \sum_{R_i \in R: e \in R_i} x_{R_i} \leq u_e \quad \forall e \in E \tag{2}$$

$$x_e^{in} - x_e^{out} = 0 \quad \forall e \in E_I \tag{3}$$

$$x_{R_i} \geq 0 \quad \forall R_i \in R \subseteq \mathcal{P}, \quad i = 1, 2, 3 \tag{4}$$

Equation (1) is an objective function that maximizes the total flow reaching at sink t through paths $R_i \in R$ of all three modes. The capacity constraint of each element is represented in Eq. (2), where sum of flows on paths of all three modes through e is bounded by its movement capacity and the flow conservation constraint is represented by Eq. (3). Similarly, Eq. (4) represents the non-negativity of the flow on each path.

2.3 Abstract Dynamic Flow Model with Partial Switching

Consider a network topology $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ as the abstract dynamic network with temporal dimensions, where each path obey partial switching property. Let $\tau : E \times E \rightarrow \mathbb{Z}^+$ be a non-negative transit time of element $e \in E$ that is necessary to transship the flow from e to its right element and $T \in \mathbb{T}$ be a time horizon. If e and a are two consecutive elements on path R_i with $e <_{R_i} a$, then flow traveling through e at time θ reaches a at time $\theta + \tau_e$. In discrete time setting, time horizon is discretized as $\mathbb{T} = \{0, 1, \dots, T\}$. For each s - t path $R_i \in R \subseteq \mathcal{P}$, $\tau_{R_i} = \sum_{a \in R_i[s \rightarrow t]} \tau_a$

denotes the traversal time of the path flow from s to t .

Let $\psi_{R_i}(\theta) : R_i \times T \rightarrow \mathbb{R}^+$ be the dynamic s - t path flow with partial switching in each discrete time steps $\theta \in \mathbb{T}$. Let $\psi_e^{out} = \sum_{R_i \in A_e} \psi_{R_i}$ and $\psi_e^{in} = \sum_{R_i \in B_e} \psi_{R_i}$ be the total outflow from e and inflow into e , respectively. The linear programming model for abstract dynamic flow with partial switching is

$$\max \sum_{i=1}^3 \sum_{\theta=\tau_{R_i}}^T \sum_{R_i \in R} \psi_{R_i}(\theta) \quad (5)$$

$$\text{s.t.} \quad \sum_{i=1}^3 \sum_{R_i \in R: e \in R_i} \psi_{R_i}(\theta) \leq u_e \quad \forall e \in E, \theta \in \mathbb{T} \quad (6)$$

$$\psi_e^{in}(\theta) - \psi_e^{out}(\theta) \geq 0 \quad \forall e \in E_I, \theta \in \mathbb{T} \quad (7)$$

$$\psi_{R_i} \geq 0 \quad \forall R_i \in R \subseteq \mathcal{P}, i = 1, 2, 3 \quad (8)$$

Equation (5) represents an objective function which aims to maximize the total flow reaching to the sink t through the paths of all three modes within given time horizon T . Similarly, Eq. (6) represents the capacity constraint of each element at $\theta \in \mathbb{T}$, where total flows on paths of all three modes through e is bounded by its movement capacity. Equation (8) represents the non-negativity of the flow on each path. Non-conservation of the flow at intermediate elements in each time step θ is presented in Eq. 7, where the equality condition holds for $\theta = T$.

3 Abstract Maximum Flow with Partial Switching

In this section, we discuss on the maximum static and maximum dynamic flow problems in abstract network topology with partial switching of paths and present their solution strategies in brief. The detailed solution strategies and the proof of correctness are presented in [15].

3.1 Maximum Static Flow

Here, we formulate the problem and its solution strategy for an abstract maximum static flow with partial switching property on paths.

Problem 1 For a given abstract static network $\mathcal{N} = (E, \mathcal{P})$, an abstract maximum static flow problem with partial switching deals with maximization of flow leaving the source element that is to be sent to the sink element via s - t paths $R_i \in R \subseteq \mathcal{P}$ by allowing the partial switching property at each crossing element.

Now, we present a polynomial time solution procedure to solve Problem 1 by using algorithm of McCormick [24] for a single source single sink network \mathcal{N} .

Solution Strategy 1 For a given abstract static network $\mathcal{N} = (E, \mathcal{P})$ with partial switching, we first initialize $x = x_0$ if initial flow is given, otherwise set initial flow at zero. By using augmenting structure of [24], we compute an abstract maximum static flow with complete switching. After that, the residual network is constructed and residual flow is obtained if the saturated elements of complete switched paths lie in different sides of the crossing element. These residual paths are decomposed into source-sink paths with positive residual capacity. Thus the flow obtained from complete switched paths together with residual path provides the abstract maximum static flow with partial switching on \mathcal{N} .

Theorem 3.1 *Solution strategy 1 provides an optimal abstract maximum static flow with partial switching in polynomial time.*

3.2 Maximum Dynamic Flow

In this section, we discuss the abstract maximum dynamic flow problem with partial switching. By defining temporal paths, as in Kappmeier [14], we use temporally repeated flow to solve the problem in polynomial time.

Let u and τ be non-negative capacity and transit time functions, respectively, that are assigned to transship the flow from one element to its adjacent element via some path $R_i \in R$. Let $T \in \mathbb{T} = \{0, 1, \dots, T\}$ be given time horizon. As in classical network, time expanded elements are obtained by creating $T + 1$ copies of the elements for each time step $\theta \in \mathbb{T}$. The set of time expanded elements is

$$E_T = \{e^\theta : e \in E, \theta \in \mathbb{T}\},$$

where e^0 represents the original set of elements in the given network. Flow starting from source element s at time θ reaches to e along path R_i at time $\theta + \sum_{a \in R_i[s \rightarrow e]} \tau_a$. For each path $R_i \in R \subseteq \mathcal{P}$, the temporal path R_i^θ for each $\theta \in \{0, 1, \dots, T\}$ is the copy

of elements of R_i in which flow starts on it at time θ and travels through path R_i . That is,

$$R_i^\theta = \left\{ e^\beta \in E_T : e \in R_i, \beta = \theta + \sum_{a \in R_{i[s \rightarrow e]}} \tau_a \right\}.$$

If we replace arbitrary element e by sink element t , then it represents source-sink temporal path. The order of elements in temporal path R_i^θ is same as in R_i . The set of all temporal paths that reaches to the sink element t within time horizon T is

$$\mathcal{P}_T^\theta = \left\{ R_i^\theta : R_i \in R, \theta \in \mathbb{T}, \theta + \sum_{e \in R_i} \tau_e \leq T \right\}.$$

The abstract path system $(E_T, \mathcal{P}_T^\theta)$ may not be abstract network because it may not satisfy (partial) switching property [14]. To handle this problem, paths with delay in each element are obtained by defining the delay function $\delta : R_i \rightarrow \{0, 1, \dots, T\}$ for each $e \in R_i$. Every flow traveling from s along path R_i with delay pattern δ reaches to $e \in R_i$ at time $\sum_{a \in R_{i[s \rightarrow e]}} (\tau_a + \delta_a) + \delta_e$. The temporal path with delay pattern is

$$R_i^\delta = \left\{ e^\beta \in E_T : e \in R_i, \beta = \sum_{a \in R_{i[s \rightarrow e]}} (\tau_a + \delta_a) + \delta_e \right\}.$$

The order of elements in R_i^δ are same as in R_i . It is to be noted that due to delay on elements, transit times of paths may change and so we use the paths with delay pattern that can transship the flow to the sink within the given time horizon. The set of temporal paths with delay pattern δ arriving the sink t within time T is

$$\mathcal{P}_T^\delta = \left\{ R_i^\delta : R_i \in R, \delta \in \{0, 1, \dots, T\}^{R_i}, \sum_{e \in R_i} (\tau_e + \delta_e) \leq T \right\}.$$

Here, we denote the T -time expanded network of temporal paths with delay pattern by $\mathcal{N}_T^\delta = (E_T, \mathcal{P}_T^\delta)$. Now, we present the abstract maximum dynamic flow problem with partial switching as follows.

Problem 2 Let $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ be a given abstract dynamic network. An abstract maximum dynamic flow problem with partial switching is to find the maximum flow leaving the source element that is to be sent to the sink via s - t paths $R_i \in R \subseteq \mathcal{P}$ by allowing the partial switching property at each crossing element within given time horizon T .

Here, we present a polynomial time solution procedure to solve Problem 2.

Solution Strategy 2 We have given an abstract dynamic network $\mathcal{N} = (E, \mathcal{P}, \tau, T)$. We first use the static flow computation to find the partially switched path flows on R_i as described in Solution Strategy 1. For each R_i , we construct temporal paths R_i^θ in each time step θ . As these temporal paths may not satisfy the partial switching property, abstract paths with delay pattern R_i^δ are essential. Now as in [13], abstract maximum dynamic flow with partial switching is obtained by using temporally repeated flow on R_i^δ .

Theorem 3.2 *Solution strategy 2 computes an abstract maximum dynamic flow with partial switching in polynomial time.*

4 Abstract Quickest Flow with Partial Switching

Quickest flow problem is a dynamic network flow problem, in which smallest possible time is estimated to send the given amount of flow $|\psi'|$ from the source to the destination. As in Burkard et al. [4], the maximum dynamic flow is a non-decreasing function of time T and finding a solution to the quickest flow problem satisfying given amount of flow $|\psi'|$ is equivalent to finding the minimum time T' such that $|\psi(T')| \geq |\psi'|$. For classical network topology, a polynomial time algorithm by using parametric search is presented in [4]. Here, we introduce the abstract quickest flow problem with partial switching and present a polynomial time solution strategy to solve the problem.

Problem 3 Let $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ be a given abstract dynamic network. An abstract quickest flow problem with partial switching is to find the minimum possible time T' to transship the given amount of flow $|\psi'|$ from the source element that is to be sent to the sink element via s - t paths $R_i \in R \subseteq \mathcal{P}$ by allowing the partial switching property at each crossing element.

To solve the problem, we first construct the temporal paths with delay pattern. We send the maximum amount of flow in each temporal path unless the given demand is not satisfied. To obtain the solution in polynomial time, we use binary search method starting with interval $[T_{min}, T_{max}]$ such that $|\psi(T_{min})| \leq |\psi'| \leq |\psi(T_{max})|$. Thus, the quickest time T' lies between T_{min} and T_{max} . Initially, we set $T_{min} = \tau_{\bar{R}_i}$ and $T_{max} = \tau_{\bar{R}_i} + \left\lceil \frac{|\psi'|}{x_{\bar{R}_i}} \right\rceil$, $x_{\bar{R}_i} > 0$, where $\bar{R}_i \in R$ is the shortest path with partial switching and $x_{\bar{R}_i}$ is the static flow on \bar{R}_i . In each iteration, the searched interval is halved unless the point of convergence is obtained and the flow values at extreme points of interval are obtained by using polynomial time algorithm of Kappmeier et al. [13]. Depending on this technique, we now present an algorithm for the abstract quickest flow with partial switching.

Algorithm 1: Abstract quickest flow algorithm with partial switching

Input : Given abstract dynamic network $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ and flow value $|\psi'|$.

Output: Abstract quickest flow with partial switching on \mathcal{N} .

- (1) Using static flow computation, find the partially switched paths R_i .
 - (2) From R_i , construct temporal paths R_i^θ .
 - (3) Construct abstract paths R_i^δ with delay pattern δ .
 - (4) Use parametric search on $[T_{min}, T_{max}]$ unless $|\psi(T')|$ converges to $|\psi'|$.
 - (5) T' = quickest time to satisfy demand $|\psi'|$.
-

Theorem 4.1 *The solution provided by Algorithm 1 is optimal.*

Proof Before proving optimality, we first prove feasibility of Algorithm 1. As described in Solution Strategy 1, Step 1 provides the static flow in R_i which is feasible. In Steps 2 and 3, construction of temporal paths and paths with delay pattern, as in [13], are feasible. As the maximum dynamic flow function $\psi(T)$ is non-decreasing function of T , inquiry of quickest time satisfying the given demand using parametric search is also feasible. Hence, Algorithm 1 provides feasible solution for abstract quickest flow problem with partial switching.

Next, optimality of the algorithm is dominated by Step 4. Due to non-decreasing function $\psi(T)$, searching of the quickest time T' to satisfy the given demand $|\psi'|$ in the interval $[T_{min}, T_{max}]$ can be obtained in polynomial time by using parametric search. As the sequence of flow values obtained from the iterative search converges to the given demand $|\psi'|$, the solution provided by Algorithm 1 is optimal.

Theorem 4.2 *Algorithm 1 solves an abstract quickest flow problem with partial switching in polynomial time.*

Proof The time complexity of maximum dynamic flow is polynomial by using oracle [13]. Similarly, time complexity of quickest time using parametric search on $[T_{min}, T_{max}]$ is also polynomial [4]. So Algorithm 1 solves an abstract quickest flow problem with partial switching in polynomial time.

5 Abstract Quickest Contraflow with Partial Switching

In a two-way abstract network, contraflow configuration means the reversal of oppositely directed paths towards the destination element to increase the outbound capacity of paths and reduce the overall transmission time of the flow.

Consider a two-way dynamic network $\mathcal{N} = (E, \vec{\mathcal{P}}, \tau, T)$, where $\vec{\mathcal{P}} = \vec{P} \cup \bar{P}$ represents the set of two-way paths in which \vec{P} and \bar{P} denote the forward $P_{[s \rightarrow t]}$ and the backward $\bar{P}_{[t \rightarrow s]}$ source-sink paths, respectively. Let $\tau : E \times E \rightarrow \mathbb{Z}^+$ be a symmetric

transit time between pair of consecutive elements along a path so that $\tau_{\vec{P}_{[e \rightarrow a]}} = \tau_{\overleftarrow{P}_{[a \rightarrow e]}}$ with $e <_{\vec{P}} a$ and $a <_{\overleftarrow{P}} e$. Clearly, $\tau_{\vec{P}} = \tau_{\overleftarrow{P}}$. Contrary to the general abstract network, contraflow network contains the incoming movement capacity to the source and outgoing movement capacity from the sink. Using the concept of contraflow, we introduce the quickest flow problem in abstract network with partial switching herein.

Problem 4 For a given abstract dynamic network $\mathcal{N} = (E, \vec{P}, \tau, T)$, an abstract quickest contraflow problem with partial switching is to find the minimum possible time to transship the given amount of flow from the source element that is to be sent to the sink via s - t paths $\vec{P} \cup \overleftarrow{P}$ by reverting the direction of paths \overleftarrow{P} at time zero and allowing the partial switching property to obtain the paths $R_i \in R \subseteq \mathcal{P}$.

As a solution procedure, an auxiliary network $\bar{\mathcal{N}} = (E, \bar{P}, \bar{\tau}, T)$ is constructed by adding two-way movement capacities between two consecutive elements. The set of paths in an auxiliary network \bar{P} is obtained by reverting the direction of paths \overleftarrow{P} at time zero. The movement capacity \bar{u}_e in auxiliary network is obtained by adding two way capacities i.e., for any two consecutive elements e and a with $e <_{\vec{P}} a$ and $a <_{\overleftarrow{P}} e$

$$\bar{u}_e = u_{e:e \in \vec{P}} + u_{a:a \in \overleftarrow{P}}$$

where $u_{a:a \in \overleftarrow{P}} = 0$ if $a \notin \overleftarrow{P}$. The transit time $\bar{\tau}_e$ is obtained as follows.

$$\bar{\tau}_e = \begin{cases} \tau_{e:e \in \vec{P}} & \text{if } e <_{\vec{P}} a \\ \tau_{a:a \in \overleftarrow{P}} & \text{otherwise.} \end{cases}$$

Now, in the auxiliary network, partial switching property is used to obtain s - t paths R_i^δ with delay pattern and the quickest flow with partial switching along paths R_i^δ is calculated by using the Algorithm 2.

Algorithm 2: Abstract quickest contraflow algorithm with partial switching

Input : Given abstract two-way network $\mathcal{N} = (E, \overleftrightarrow{P}, \tau, T)$.

Output: Abstract maximum dynamic contraflow with partial switching.

- (1) Construct an auxiliary network $\bar{\mathcal{N}} = (E, \bar{P}, \bar{\tau}, T)$.
 - (2) Construct cycle free s - t paths R_i^δ with delay pattern on $\bar{\mathcal{N}}$ satisfying the partial switching property.
 - (3) Compute abstract quickest flow in $\bar{\mathcal{N}}$ with partial switching by using Algorithm 1.
 - (4) A path \overleftarrow{P} is reversed if and only if the flow along path \overrightarrow{P} is greater than its capacity or if there is a non-negative flow along path $\overrightarrow{P} \notin \overleftrightarrow{P}$.
-

Theorem 5.1 *The feasible solution to an abstract quickest contraflow problem with partial switching can be obtained by using Algorithm 2 in polynomial time.*

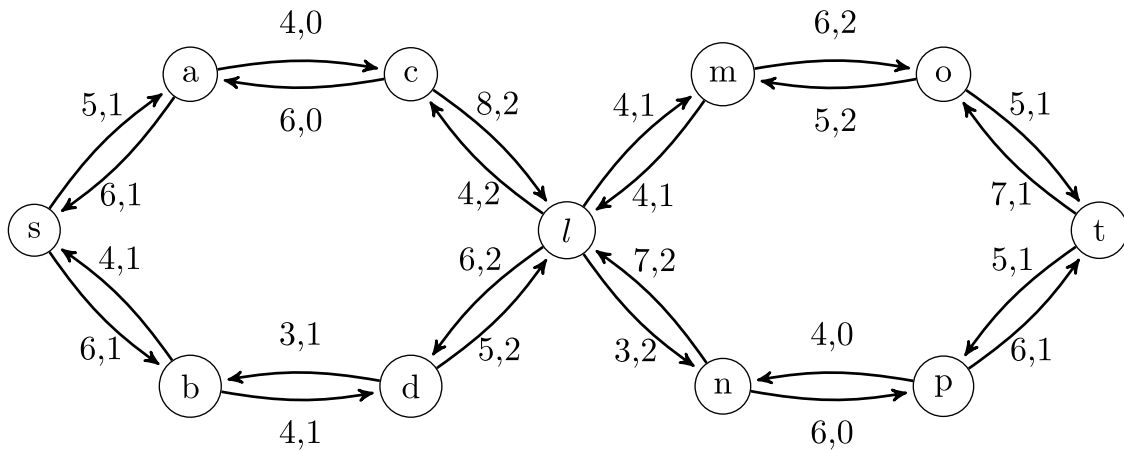


Fig. 2 Two way network \mathcal{N} with capacity and transit times from each element to its adjacent element

Proof First we show feasibility of Algorithm 2 and then prove its time complexity. Construction of auxiliary network by adding two capacities of paths between consecutive elements and symmetric transit time is feasible. In the auxiliary network, we construct partially switched paths R_i^δ with delay pattern δ which is feasible as in Sect. 3. Similarly, computation of the quickest flow on auxiliary network using Algorithm 1 is also feasible. So Algorithm 2 provides feasible solution for abstract quickest contraflow problem with partial switching. The proof of optimality is as similar to Theorem 4.1. Next, Steps 1 and 2 can be computed in $O(E)$ times and Step 3 can be computed in polynomial time using Algorithm 1. So, the time complexity of Algorithm 2 is polynomial.

Example 1 Consider a two way abstract network with movement capacity and transit time between the elements, which is represented in Fig. 2. It consists of 4 forward paths (i.e., $s - t$ paths: $P_1 = s - a - c - l - n - p - t$, $P_2 = s - b - d - l - m - o - t$, $P_3 = s - a - c - l - m - o - t$, $P_4 = s - b - d - l - n - p - t$) and 4 backward paths (i.e., $t - s$ paths: P_5, P_6, P_7, P_8 with reverse direction of P_1, P_2, P_3, P_4 , respectively). Here, forward paths P_1 and P_2 (similarly backward paths P_5 and P_6) crosses at element l .

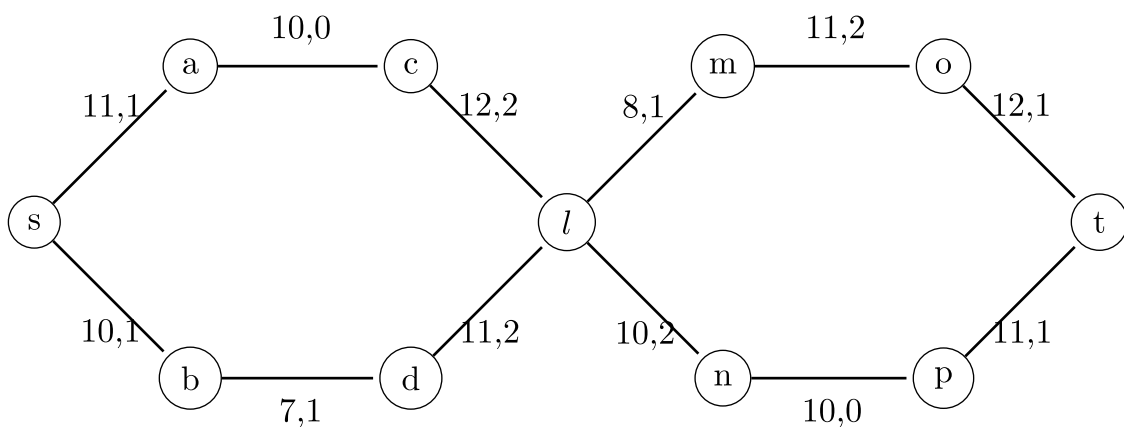


Fig. 3 Auxiliary network $\tilde{\mathcal{N}}$ of network \mathcal{N} in Fig. 2

We construct an auxiliary network $\bar{\mathcal{N}}$ (see Fig. 3) from the given network. In the auxiliary network $\bar{\mathcal{N}}$, we construct three partially switched paths $P_3 = s - a - c - l - m - o - t$, $P_4 = s - b - d - l - n - p - t$ and $P_1 = s - a - c - l - n - p - t$, representing three modes of paths R_1 , R_2 and R_3 , respectively. The flow values and transit times on these paths are $|\psi_{P_3}| = 8$, $|\psi_{P_4}| = 7$, $|\psi_{P_1}| = 2$, $\tau_{P_3} = 7$, $\tau_{P_4} = 7$ and $\tau_{P_1} = 6$.

Let the given flow value at the source element be $|\psi'| = 65$ units. We have to find the minimum possible time to sent it with contraflow configuration by using Algorithm 1 in $\bar{\mathcal{N}}$. Here, binary search is applied with $T_{min} = 6$, $T_{max} = 6 + \left\lceil \frac{65}{2} \right\rceil = 39$. The minimum possible time to satisfy given demand is $T = 10$ units. It is to be noted that, if we do not apply contraflow configuration then it takes $T = 16$ units of time to transship the given flow which is 60% more than the time with contraflow.

This paper is mainly focused on the task of disaster management with three aspects: use of crossing free paths, minimization of the clearance time and use of contraflow configuration. At any kind of disasters, movement of people increases rapidly within very short period of time which creates the unexpected congestions. Movement of flows without crossing effect can be a key factor to smooth the traffic efficiently. Quickest flow problem with partial switching helps to minimize the clearance time by using the paths without congestion. Similarly, quickest contraflow technique with partial switching supports even more to apply quickest transmission of flow by reversing the empty paths towards the destination. So, we hope that these solution strategies will be highly applicable for the disaster management.

6 Conclusions

Abstract network flows have been well-studied in literature. By using complimentary slackness on the augmenting path structure, a polynomial time procedure was obtained for the abstract static flow problem. The lexicographically maximum flow, maximum flow over time and earliest arrival flow problems have been solved efficiently in abstract networks with complete switching of paths.

In this paper, we have presented the abstract flow models with partial switching in static and dynamic networks. We have presented polynomial time solution procedures to solve the abstract maximum static and dynamic flow problems by allowing partial switching property at the crossing elements. As the quickest flow problems are very relevant for disaster management, we have introduced the abstract quickest flow and contraflow problems and presented polynomial time algorithms to solve these problems by partial switching of paths. To the best of our knowledge, these problems and the solution strategies for the abstract network topology are introduced for the first time.

Acknowledgements The first author (Durga Prasad Khanal) thanks to the German Academic Exchange Service - DAAD for Research Grants - Bi-nationally Supervised Doctoral Degrees/Cotutelle, 2021/22 and University Grants Commission Nepal for PhD Research Fellowship, 2020/21. Similarly, the second author (Urmila Pyakurel) thanks to the Alexander von Humboldt Foundation for Digital Cooperation

Fellowship (August 1, 2021 - January 31, 2022) and fellowship for Remote Cooperation Abroad Project (March 1 - August 31, 2022).

Data Availability The authors have not used any additional data in this article.

Declarations

Conflict of Interest The authors declare no competing interests.

References

1. Achrekar O, Vogiatzis C (2018) Evacuation trees with contraflow and divergence considerations. In: Kotsireas I, Nagurney A, Pardalos P. (eds) Dynamics of Disasters. DOD 2017. Springer Optimization and Its Applications vol 140. Springer, Cham. https://doi.org/10.1007/978-3-319-97442-2_1
2. Andreas AK, Smith JC (2009) Decomposition algorithms for the design of a nonsimultaneous capacitated evacuation tree network. *Networks: An International Journal* 53(2):91–103
3. Bretschneider S, Kimms A (2011) A basic mathematical model for evacuation problems in urban areas. *Transp Res A Policy Pract* 45(6):523–539
4. Burkard RE, Dlaska K, Klinz B (1993) The quickest flow problem. *ZOR- Methods and Models of Operational Research* 37:31–58
5. Dhamala TN, Gupta SP, Khanal DP, Pyakurel U (2020) Quickest multi-commodity flow over time with partial lane reversals. *J Math Stat* 16:198–211
6. Ford LR, Fulkerson DR (1956) Maximal flow through a network. *Can J Math* 8:399–404
7. Ford LR, Fulkerson DR (1962) *Flows in networks*. Princeton University Press Princeton New Jersey
8. Goldberg AV, Tarjan RE (1990) Finding minimum-cost circulations by successive approximation. *Math Oper Res* 15(3):430–466
9. Hamacher H, Tjandra S (2001) Mathematical modeling of evacuation problems: a state of art. *Berichte des Fraunhofer ITWM, Nr, p 24*
10. Hasan MH, Van Hentenryck P (2020) Large-scale zone-based evacuation planning. *Models and algorithms, Part I*. <https://doi.org/10.1002/net.21981>
11. Hasan MH, Van Hentenryck P (2021) Large-scale zone-based evacuation planning, Part II: Macroscopic and microscopic evaluations. *Networks* 77:341–358. <https://doi.org/10.1002/NET.21980>
12. Hoffman AJ (1974) A generalization of max flow - min cut. *Math. Prog.* 6:352–359
13. Kappmeier J-PW, Matuschke J, Peis B (2014) Abstract flow over time: A first step towards solving dynamic packing problems. *Theor Comput Sci. Algorithms Combin* 544:74–83
14. Kappmeier PW (2015) Generalizations of flows over time with application in evacuation optimization. PhD Thesis, Technical University, Berlin, Germany
15. Khanal DP, Pyakurel U, Dhamala TN. Abstract flow with partial switching for evacuation planning, Under Review
16. Khanal DP, Pyakurel U, Dhamala TN (2021) Maximum multi-commodity flow with intermediate storage. *Mathematical Problems in Engineering*, 2021, Article ID 5063207. <https://doi.org/10.1155/2021/5063207>
17. Khanal DP, Pyakurel U, Dempe S (2021) Dynamic contraflow with orientation dependent transit times allowing intermediate storage. *Nepali Math Sci Rep* 38(2):1–12. <https://doi.org/10.3126/nmsr.v38i2.42700>
18. Kim S, Shekhar S (2005) Contraflow network reconfiguration for evacuation planning: a summary of results, In: *Proceedings of 13th ACM Symposium on Advances in Geographic Information Systems GIS vol. 05*, 250–259
19. Lin M, Jaillet P (2015) On the quickest flow problem in dynamic networks: a parametric min-cost flow approach. In: *Proceeding of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia: Society for Industrial and Applied Mathematics 1343–1356
20. Liu Y, Lai X, Chang GL (2006) Cell-based network optimization model for staged evacuation planning under emergencies. *Transportation Res. Rec. J Transportation Res Board* 1:127–135

21. Lu Q, George B, Shekhar S (2005) Capacity constrained routing algorithms for evacuation planning: a Summary of results. In: Bauzer Medeiros, E. M. B. E., C., ed., *Lecture Notes in Computer Science*, vol. 3633, Springer, Berlin, 291-307
22. Martens M (2007) *Path-Constrained Network Flows*. PhD Thesis, Technical University, Berlin, Germany
23. Martens M, McCormick ST (2008) A polynomial algorithm for weighted abstract flow. In *Integer Programming and Combinatorial Optimization*, *Lecture Notes in Computer Sciences* 5035:97–111
24. McCormick ST (1996) A polynomial algorithm for abstract maximum flow. In *Proceeding of the 7th annual ACM-SIAM symposium on discrete algorithms*, 490-497
25. Purba DSD, Kontou E, Vogiatzis C (2021) Evacuation network modeling for alternative fuel vehicles. arXiv preprint. [arXiv:2109.01578](https://arxiv.org/abs/2109.01578)
26. Pyakurel U, Dempe S (2020) Network flow with intermediate storage: models and algorithms. *SN Operations Research Forum*. <https://doi.org/10.1007/s43069-020-00033-0>
27. Pyakurel U, Dempe S (2021) Universal maximum flow with intermediate storage for evacuation planning. In: Kotsireas I.S., Nagurney A., Pardalos P.M., Tsokas A. (eds) *Dynamics of Disasters*. Springer Optimization and Its Applications vol. 169, Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-64973-9_14
28. Pyakurel U, Dhamala TN, Dempe S (2017) Efficient continuous contraflow algorithms for evacuation planning problems. *Annals of Operations Research (ANOR)* 254:335–364
29. Pyakurel U, Khanal DP, Dhamala TN (2022) Abstract network flow with intermediate storage for evacuation planning. *Eur J Oper Res*. <https://doi.org/10.1016/j.ejor.2022.06.054>
30. Pyakurel U, Nath HN, Dhamala TN (2018) Efficient contraflow algorithms for quickest evacuation planning. *SCIENCE CHINA Math* 61:2079–2100
31. Pyakurel U, Nath HN, Dhamala TN (2019) Partial contraflow with path reversals for evacuation planning. *Ann Oper Res*. <https://doi.org/10.1007/s10479-018-3031-8>
32. Vogiatzis C, Walteros JL, Pardalos PM (2013) Evacuation through clustering techniques. In: Goldengorin B., Kalyagin V., Pardalos P. (eds) *Models, Algorithms, and Technologies for Network Analysis*. Springer Proceedings in Mathematics & Statistics, vol 32. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-5574-5_10
33. United Nations International Strategy for Disaster Reduction (UNISDR) Geneva, Switzerland, (2009). *Terminology on Disaster Risk Reduction*. https://unisdr.org/files/7817_UNISDRTerminologyEnglish.pdf

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Proceeding Paper

Maximum Multi-Commodity Flow with Proportional and Flow-Dependent Capacity Sharing [†]

Durga Prasad Khanal ¹, Urmila Pyakurel ^{2,*}, Tanka Nath Dhamala ² and Stephan Dempe ³

¹ Saraswati Multiple Campus, Tribhuvan University, Kathmandu 44618, Nepal; durgapsdkhanal@gmail.com

² Central Department of Mathematics, Tribhuvan University, Kathmandu 44618, Nepal; tanka.nath.dhamala@gmail.com

³ Faculty of Mathematics and Computer Science, TU Bergakademie Freiberg, D-09599 Freiberg, Germany; dempe@math.tu-freiberg.de

* Correspondence: urmilapyakurel@gmail.com

† Presented at the 1st International Electronic Conference on Algorithms, 27 September–10 October 2021; Available online: <https://ioca2021.sciforum.net/>.

Abstract: Multi-commodity flow problems concerned with the transshipment of more than one commodity from respective sources to the corresponding sinks without violating the capacity constraints on the arcs. If the objective of the problem is to send the maximum amount of flow within a given time horizon, then it becomes the maximum flow problem. In multi-commodity flow problems, the flow of different commodities departing from their sources arriving at the common intermediate node have to share the capacity through the arc. The sharing of the capacity in the common arc (bundle arc) is one of the major issues in the multi-commodity flow problems. In this paper, we introduce the maximum static and maximum dynamic multi-commodity flow problems with proportional capacity sharing and present polynomial time algorithms to solve the problems. Similarly, we investigate the maximum dynamic multi-commodity flow problems with flow-dependent capacity sharing and present a pseudo-polynomial time solution strategy.

Keywords: multi-commodity; maximum flow; proportional capacity sharing; flow-dependent capacity sharing



Citation: Khanal, D.P.; Pyakurel, U.; Dhamala, T.N.; Dempe, S. Maximum Multi-Commodity Flow with Proportional and Flow-Dependent Capacity Sharing. *Comput. Sci. Math. Forum* **2022**, *2*, 5. <https://doi.org/10.3390/IOCA2021-10904>

Academic Editor: Frank Werner

Published: 26 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

A topological structure with links and crossings, known as arcs and nodes, respectively, is a network in which entities are transshipped from one point to another. The initial and the final points are termed as source and sink nodes, respectively. In a multi-terminal network, the transshipment of more than one commodity from the respective sources to the corresponding sinks satisfying the capacity constraints on the arcs is a multi-commodity flow (MCF) problem. Supply chain networks, message routine in telecommunication, and transportation networks are some examples of multi-commodity network topology.

Ford and Fulkerson [1] introduced the concept of the static multi-commodity flow problem, and thereafter many researchers have contributed to the different aspects of the multi-commodity flow problems [2–5]. If the demand and supply of each commodity is to be maximized in the given time horizon, then the problem becomes a maximum dynamic multi-commodity flow problem. The static multi-commodity flow problem is polynomial time solvable by using the ellipsoid or interior point method, whereas the dynamic multi-commodity flow problem is NP-hard [6]. Kappmeier [7] provided the solution to the maximum dynamic multi-commodity flow problem using a time-expanded network in a pseudo-polynomial time complexity. Pyakurel et al. [8] presented a polynomial time algorithm for the maximum static flow problem and pseudo-polynomial algorithms for the earliest arrival transshipment and maximum dynamic flow problems with partial contraflow. A priority based multi-commodity flow problem can be found in Khanal

et al. [9]. Using the concept of intermediate storage introduced by Pyakurel and Dempe [10], Khanal et al. [11] presented a polynomial time algorithm for the maximum static—and a pseudo-polynomial time algorithm for the maximum dynamic—multi-commodity flow problems with intermediate storage.

The sharing of the bundle arc capacity is one of the major issues in the multi-commodity flow problems. For each commodity, if the sharing of the capacity of the bundle arc is set in proportion to the bottleneck capacity of path from their respective sources to the tail node of the bundle arc, then it is known as proportional capacity sharing. In this case, the shared capacity of the bundle arc for each commodity is fixed and the multi-commodity flow problem is reduced to an independent single commodity flow problem. To avoid the fractional flow, we can use ceiling and floor functions with an appropriate manner. Similarly, if the sharing of the capacity of the bundle arc is made according to the inflow rate of the flow of each commodity, then it is termed as flow-dependent capacity sharing. In this method, the shared capacity of the bundle arc may not always be the same as the flow on the arc may vary over the time. We investigate these two sharing techniques hereafter in Sections 2.1 and 2.2.

In this paper, we introduce the maximum multi-commodity flow problem using proportional as well as flow-dependent capacity sharing on the bundle arcs. We present the polynomial time algorithms for the static as well as the dynamic multi-commodity flow problems, using proportional capacity sharing in Section 3. Similarly, in Section 4 a pseudo-polynomial time algorithm for the dynamic multi-commodity flow problem with flow-dependent capacity sharing is presented. The paper is concluded in Section 5.

2. Basic Terminologies

Consider a network topology $G = (N, A, K, u, \tau, d_i, S, D, T)$ with commodity $i \in K = \{1, 2, \dots, k\}$, set of nodes N and set of arcs A . Here, d_i represents the demand/supply of each commodity $i \in K$ which is routed through a unique source–sink pair s_i – t_i , where $s_i \in S \subseteq N$ and $t_i \in D \subseteq N$. Each arc $e = (v, w) \in A$ with $head(e) = w$ and $tail(e) = v$ is equipped with a capacity function $u: A \rightarrow \mathcal{R}^+$ that restricts the flow of the commodity and a non-negative transit time function $\tau: A \rightarrow \mathcal{R}^+$ that measures the time to transship the flow from node v to node w . Let $\delta^+(v)$ and $\delta^-(v)$ be the set of outgoing arcs from node v and the incoming arcs to node v , respectively. We denote P_i as the set of all paths of the commodity i such that $P \in P_i$ is a s_i – t_i path and $P_{[s_i, v]} \in P_i$ represents the path from s_i to the intermediate node v . The time horizon is denoted by $T = \{0, 1, \dots, T\}$ in discrete time settings and $T = [0, T + 1)$ in continuous time settings. In case of static flow, the time parameters T and τ are absent.

2.1. Proportional Capacity Sharing

The multi-commodity flow problem differs from the single commodity flow problem due to the bundle constraints and the unique source–sink flow for each commodity. Our assumption is that the nature of flows inside the same commodity group are homogeneous and between the commodity groups are heterogeneous yet uniform in the occupancy rate of the arc capacity. To share the capacity of the bundle arc, we propose a proportional capacity sharing technique depending on the minimum of the arc capacity of paths $P_{[s_i, v]}$, (that is, bottleneck capacity of path $P_{[s_i, v]}$) for each commodity i from their respective sources s_i to the tail v of bundle arc $e = (v, w)$ as follows: Let u_e be the capacity of a bundle arc e , then proportional sharing of capacity u_e for each commodity $i \in K$ is,

$$u_e^i = \frac{u_a^i}{\sum_{a \in P_{[s_i, v]}: i \in K} u_a^i} u_e \tag{1}$$

where $P_{[s_i, v]}$ is the path from s_i to the tail v of bundle arc e , for all $i \in K$ and a is an arc in $P_{[s_i, v]}$ with minimum capacity. Here, u_e^i represents the portion of the capacity of the arc e allocated for the commodity i . Clearly, the sum of the shared capacities over each commodity is equal to the original arc capacity, i.e., $\sum_{i \in K} u_e^i = u_e$.

The shared capacity may be in fraction, i.e., $u_e^i = \text{int}(u_e^i) + \text{fra}(u_e^i)$, the sum of the integral part and the fractional part, respectively. The fractional capacities can be converted into the integral capacities as follows:

- Find the sum $\sum \text{fra}(u_e^i)$. If $\sum \text{fra}(u_e^i) = p$, then the first p fractional capacities with the greatest fractional part (with descending order of the fractional part) are rounded up using the ceiling function $\lceil \cdot \rceil$ and the remaining capacities are rounded below by the floor function $\lfloor \cdot \rfloor$.
- If the same fractional part occurs in more than one commodity then priority is given to the capacity with the greatest integral part among them.
- In case of equal integral parts, priority goes to the commodity with the higher demand among them. If the demand values are also the same, then either of them can be rounded up.

It is to be noted that if $u_e^i < 1$ and has no alternative path for commodity i , then it may block the transshipment of the flow. In such a case, the fractional capacity is to be accepted.

2.2. Flow-Dependent Capacity Sharing

In the proportional capacity sharing technique the shared capacity of each commodity remains fixed at each time step θ . In this subsection, we present the flow-dependent capacity sharing technique, where the share of the capacity for each commodity depends on the inflow rate of the flow f in the predecessor arcs. At any instance of time θ , if a bundle arc $e = (v, w)$ with the capacity u_e holds more than one commodity $i \in K$, then the flow-dependent capacity sharing of u_e for each commodity $i \in K$ is,

$$u_e^i(\theta) = \frac{f_a^i(\theta - \tau_a)}{\sum_{a \in \alpha(e): i \in K} f_a^i(\theta - \tau_a)} u_e \tag{2}$$

where $\alpha(e)$ is the set of the predecessor arcs of bundle arc e so that $a \in \alpha(e) \Rightarrow \text{head}(a) = \text{tail}(e)$ and $u_e^i(\theta)$ is the portion of the capacity of arc e for the commodity i at time θ . For each time θ , the sum of the portion of the shared capacities $u_e^i(\theta)$ over all the commodities $i \in K$ is equal to the original arc capacity, i.e., $\sum_{i \in K} u_e^i(\theta) = u_e$. If the shared capacities are in fraction, we can convert them into integer values as described in Section 2.1.

3. Maximum MCF with Proportional Capacity Sharing

3.1. Maximum Static Multi-Commodity Flow

In the static network $G = (N, A, K, u, d_i, S, D)$ the multi-commodity flow φ with proportional capacity sharing is the sum of the non-negative flows $\varphi^i : A \rightarrow \mathcal{R}^+$ for each i with demand d_i satisfying the proportional capacity sharing Equation (1) together with the conditions (3) and (4).

$$\sum_{e \in \vec{\delta}(v)} \varphi_e^i - \sum_{e \in \overleftarrow{\delta}(v)} \varphi_e^i = \begin{cases} d_i & \text{for } v = s_i \\ -d_i & \text{for } v = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in N, i \in K \tag{3}$$

$$0 \leq \varphi_e^i \leq u_e^i \quad \forall e \in A, i \in K \tag{4}$$

The constraints in (3) represent the supply/demand at the source/sink nodes and the flow conservation constraints at the intermediate nodes, whereas the constraints in (4) represent the boundedness of the flow on the arcs by their capacities. By taking the sum over each commodity in the later equation, we get the bundle constraints $0 \leq \sum_{i \in K} \varphi_e^i \leq \sum_{i \in K} u_e^i = u_e$ for all $e \in A$. For a maximum static multi-commodity flow problem with proportional capacity sharing the objective is to maximize the total flow value $\sum_{i \in K} d_i = |\varphi|$ subject to the constraints (1), (3) and (4).

We now introduce the maximum static multi-commodity flow problem with proportional capacity sharing as follows:

Problem 1. For the given static multi-commodity network $G = (N, A, K, u, d_i, S, D)$ the maximum static multi-commodity flow problem with proportional capacity sharing is to transship the maximum flow from s_i to t_i , where the shared capacity for each commodity $i \in K$ on the bundle arc is depending on the minimum capacity of paths from the respective source to the tail node of the bundle arc.

To solve the problem, we first reduce the multi-commodity flow problem into k independent single commodity flow problems by sharing the capacity of the bundle arc using Equation (1). For each commodity i maximum static flow φ^i is obtained and the sum of the flows for the commodities is the maximum static flow value $|\varphi|$. We now present the algorithm to solve Problem 1.

Theorem 1. Algorithm 1 solves the maximum static MCF problem correctly in polynomial time complexity.

Algorithm 1: Maximum static MCF algorithm with proportional capacity sharing

Input: Given static multi-commodity flow network $G = (N, A, K, u, d_i, S, D)$.

1. Construct k independent sub-problems by proportional capacity sharing (1) on bundle arcs for all $i \in K$.
2. Compute the solution φ^i to the static maximum flow problem for all i .
3. Maximum flow $|\varphi| = \sum_{i \in K} \varphi^i$.

Output: Maximum static MCF on G with proportional capacity sharing.

3.2. Maximum Dynamic Multi-Commodity Flow

For a given dynamic network G with constant transit times τ on arc e , the MCF over time function f with proportional capacity sharing is the sum of the flows $f^i : A \times T \rightarrow \mathcal{R}^+$, satisfying the proportional capacity sharing Equation (1) together with the constraints (5) and (7).

$$\sum_{e \in \vec{\delta}(v)} \sum_{\theta=0}^T f_e^i(\theta) - \sum_{e \in \overleftarrow{\delta}(v)} \sum_{\theta=0}^T f_e^i(\theta) = \begin{cases} d_i & \text{for } v = s_i \\ -d_i & \text{for } v = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in N, i \in K \quad (5)$$

$$\sum_{e \in \vec{\delta}(v)} \sum_{\theta=0}^{\beta} f_e^i(\theta) - \sum_{e \in \overleftarrow{\delta}(v)} \sum_{\theta=0}^{\beta} f_e^i(\theta) \leq 0 \quad \forall v \notin \{s_i, t_i\}, i \in K, \beta \in T \quad (6)$$

$$0 \leq f_e^i(\theta) \leq u_e^i \quad \forall e \in A, i \in K \text{ and } \theta \in T \quad (7)$$

Here, the constraints in (5) represent the supply/demand at the sources/sinks and the flow conservation at the intermediate nodes on time horizon T . The non-conservation of the flow at the intermediate nodes in any time step β in $T = \{0, 1, \dots, T\}$ are represented by the constraints in (6). Similarly, (7) represents that the flows on the arcs are bounded above by their capacities. With these constraints, together with Equation (1), we introduce the maximum dynamic MCF problem with proportional capacity sharing, which maximizes the total flow value $\sum_{i \in K} d_i = |f|$ within the given time horizon T as follows:

Problem 2. For given dynamic multi-commodity network $G = (N, A, K, u, \tau, d_i, S, D, T)$, the maximum multi-commodity flow problem with proportional capacity sharing is to transship the maximum amount of flow from s_i to t_i within the given time horizon T , where the shared capacity for each $i \in K$ on the bundle arc is depending on the minimum capacity of paths from the respective source to the tail node of the bundle arc.

We now present an algorithm to solve Problem (2).

Theorem 2. Algorithm 2 provides the feasible solution to the maximum dynamic MCF problem with proportional capacity sharing in polynomial time.

Algorithm 2: The maximum dynamic MCF algorithm with proportional capacity sharing

Input: Given static multi-commodity flow network $G = (N, A, K, u, d_i, S, D)$.

1. Construct k independent sub-problems by proportional capacity sharing (1) on the bundle arcs for all $i \in K$.
2. Compute the maximum static flow φ^i for all i using Algorithm 1.
3. Decompose the flow φ^i into path flows φ_p^i .
4. Determine the maximum dynamic flow for each $i \in K$ using temporally repeated flow such that $f^i = \sum_{p \in P_i} (T + 1 - \tau_p) \varphi_p^i$.
5. Maximum flow $|f| = \sum_{i \in K} f^i$.

Output: Maximum dynamic MCF on G with proportional capacity sharing.

4. Maximum MCF with Flow-Dependent Capacity Sharing

For a given dynamic network G with constant transit times τ on arc e , the multi-commodity flow over time function f with flow-dependent capacity sharing is the sum of flows $f^i : A \times T \rightarrow \mathcal{R}^+$, satisfying the constraints (8)–(12).

$$\sum_{e \in \vec{\delta}(v)} \sum_{\theta=0}^T f_e^i(\theta) - \sum_{e \in \overleftarrow{\delta}(v)} \sum_{\theta=0}^T f_e^i(\theta) = \begin{cases} d_i & \text{for } v = s_i \\ -d_i & \text{for } v = t_i \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in N, i \in K \quad (8)$$

$$\sum_{e \in \vec{\delta}(v)} \sum_{\theta=0}^{\beta} f_e^i(\theta) - \sum_{e \in \overleftarrow{\delta}(v)} \sum_{\theta=0}^{\beta} f_e^i(\theta) \leq 0 \quad \forall v \notin \{s_i, t_i\}, i \in K, \beta \in T \quad (9)$$

$$\sum_{i \in K} f_e^i(\theta) \leq u_e \quad \forall e \in A \quad (10)$$

$$u_e^i(\theta) = \frac{f_a^i(\theta - \tau_a)}{\sum_{a \in \alpha(e): i \in K} f_a^i(\theta - \tau_a)} u_e \quad \forall e \in A \quad (11)$$

$$f_e^i(\theta) \geq 0 \quad \forall e \in A, i \in K \text{ and } \theta \in T \quad (12)$$

Here, the constraints in (8) and (9) have the usual meanings as represented in Section 3.2. The bundle constraints bounded by the arc capacities are presented by (10). The constraints in (11) represent the flow-dependent capacity sharing and the non-negativity of flows are represented by the constraints in (12). We now present the maximum dynamic MCF problem with flow-dependent capacity sharing satisfying the above constraints as follows:

Problem 3. For a given multi-commodity network $G = (N, A, K, u, \tau, d_i, S, D, T)$, the maximum multi-commodity flow problem with flow-dependent capacity sharing is to transship the maximum amount of flow from s_i to t_i within the given time horizon T , where shared capacity for each $i \in K$ on the bundle arc is depending on the inflow of incoming arcs of the bundle arc.

To solve the problem, we use a time-expanded layer graph.

Multi-Commodity Time-Expanded Layer Graph

The multi-commodity time-expanded layer graph is a three-dimensional graph that contains the copy of nodes from the underlying static network for every discrete time step and for each commodity. It is applicable to solve the variety of flow over time problems by applying the algorithms and techniques developed for the static network flows. For a given network G with integral transit time on the arcs and the time horizon T , the T -time-expanded layer graph G^T is obtained by creating $T + 1$ copies of node set N , which are

labeled as $N(0), N(1), \dots, N(T)$, together with an θ^{th} copy of node v labeled as $v(\theta), \theta \in T$ and the commodities $i \in K$. For every arc $e = (v, w) \in A$ and $\theta \in \{0, 1, \dots, T - \tau_e\}$, there is an arc $e_i(\theta)$ from $v_i(\theta)$ to $w_i(\theta + \tau_e)$ with the same capacity of arc e for a single commodity arc and the sharing capacity for bundle arc e . If intermediate storage is allowed at node v , then the arc from $v_i(\theta)$ to $v_i(\theta + 1)$ represents the holdover arc with infinite capacity that is used to hold the flow for the unit time interval $[\theta, \theta + 1)$ for all $\theta \in \{0, 1, \dots, T\}$.

For the graphical representation, we present a three-dimensional layer graph G^T with the set of node N , time T , and commodity K as the coordinate axes (see Figure 1). Each commodity $i \in K$ preforms the layers of the graphs in a vertical line. In Figure 1, network (a) represents a two-commodity network in which commodity-1 is transshipped from s_1 to t_1 and commodity-2 from s_2 to t_2 . Arc (x, y) is the bundle arc, which carries both commodities. Figure 1b represents the time-expanded layer graph of Figure 1a with the time horizon $T = 6$, where parallel arcs on (x, y) share the capacity for each commodity with the flow-dependent capacity sharing technique. At time step $\theta = 0$ and $\theta = 1$, no flow of commodity-1 reaches arc (x, y) , so only commodity-2 is transshipped on it; however, the capacity is shared after among the commodities. Similarly, the bundle arc transships only commodity-1 at time $\theta = 4$ due to the absence of commodity-2.

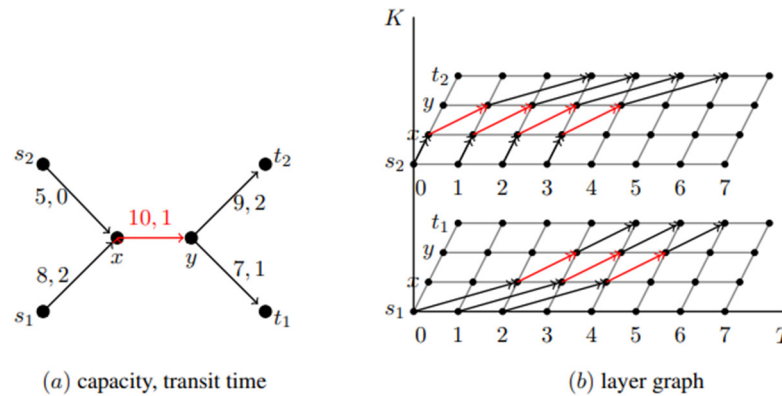


Figure 1. (b) represents the time-expanded layer graph G^T of given network (a).

Depending on the time-expanded layer graph, we now present the algorithm to solve Problem 3.

Theorem 3. A feasible solution to the maximum dynamic MCF problem with flow-dependent capacity sharing can be obtained by using Algorithm 3 in pseudo-polynomial time.

Algorithm 3: Maximum dynamic MCF algorithm with flow-dependent capacity sharing

Input: Given dynamic multi-commodity flow network $G = (N, A, K, u, \tau, d_i, S, D, T)$.

1. Construct a multi-commodity time-expanded layer graph G^T .
2. Share the capacity on the bundle arcs (parallel arcs in G^T) with flow-dependent capacity sharing (2) at each $\theta \in T$.
3. Decompose the static flow φ^i into path flows $\varphi_p^i(\theta)$ in G^T at each time step θ .
4. Maximum flow $|f| = \sum_{i \in K} \varphi_p^i$.

Output: Maximum dynamic MCF on G with proportional capacity sharing.

5. Conclusions

The maximum MCF problem deals with the transshipment of the maximum amount of flow of more than one different commodity from respective sources to the corresponding sinks within the given time horizon. Allocation of the capacity of the bundle arc to each commodity is one of the major issues in the multi-commodity flow problem. To deal with this problem we have proposed proportional capacity sharing and flow-dependent

capacity sharing. We have presented polynomial time solutions for the static—as well as the dynamic—maximum MCF problems with proportional capacity sharing and a pseudo-polynomial time algorithm with flow-dependent capacity sharing. To the best of our knowledge these solution strategies for the maximum MCF problems are introduced for the first time.

Author Contributions: D.P.K.—conceptualization, investigation and documentation, U.P., T.N.D. and S.D.—formal analysis, editing and supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research work received no specific grants from any funding in the public, commercial or non-profit organizations.

Data Availability Statement: The authors have not used any additional data in this article.

Acknowledgments: The first author (Durga Prasad Khanal) thanks to the German Academic Exchange Service—DAAD for research grants—Bi-nationally Supervised Doctoral Degree/Cotutelle, 2021/2022 and the second author (Urmila Pyakurel) thanks the Alexander von Humboldt Foundation for Digital Cooperation Fellowship (1 August 2021–31 January 2022).

Conflicts of Interest: Authors have no any conflict of interest regarding the publication of the paper.

References

1. Ford, L.R.; Fulkerson, D.R. *Flows in Networks*; Princeton University Press: Princeton, NJ, USA, 1962.
2. Ahuja, R.K.; Magnanti, T.L.; Orlin, J.B. *Network Flows: Theory, Algorithm and Applications*; Englewood Cliffs: Bergen, NJ, USA, 1993.
3. Ali, A.; Helgason, R.; Kennington, J.; Lall, H. Computational comparison among three multi-commodity network flow algorithms. *Oper. Res.* **1980**, *28*, 995–1000. [[CrossRef](#)]
4. Assad, A. Multi-commodity network flows—A survey. *Networks* **1978**, *8*, 37–91. [[CrossRef](#)]
5. Kennington, J. A survey of linear cost multi-commodity network flows. *Oper. Res.* **1978**, *26*, 209–236. [[CrossRef](#)]
6. Hall, A.; Hippler, S.; Skutella, M. Multi-commodity flows over time: Efficient algorithms and complexity. *Theor. Comput. Sci.* **2007**, *379*, 387–404. [[CrossRef](#)]
7. Kappmeier, P.W. Generalizations of Flows Over Time with Application in Evacuation Optimization. Ph.D. Thesis, Technical University, Berlin, Germany, 2015.
8. Pyakurel, U.; Gupta, S.P.; Khanal, D.P.; Dhamala, T.N. Efficient algorithms on multicommodity flow over time problems with partial lane reversals. *Int. J. Math. Math. Sci. Hindawi* **2020**, *2020*, 2676378. [[CrossRef](#)]
9. Khanal, D.P.; Pyakurel, U.; Dhamala, T.N. Prioritized multi-commodity flow model and algorithm. In Proceedings of the International Symposium on Analytic Hierarchy Process 2020, ISAHP 2020, Web Conference, 3–6 December 2020. [[CrossRef](#)]
10. Pyakurel, U.; Dempe, S. Network flow with intermediate storage: Models and algorithms. *SN Oper. Res. Forum* **2020**, *2020*, 37. [[CrossRef](#)]
11. Khanal, D.P.; Pyakurel, U.; Dhamala, T.N. Maximum multicommodity flow with intermediate storage. *Math. Probl. Eng. Hindawi* **2021**, *2021*, 5063207. [[CrossRef](#)]

Contents lists available at [ScienceDirect](https://www.sciencedirect.com)

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

Abstract network flow with intermediate storage for evacuation planning

Urmila Pyakurel^a, Durga Prasad Khanal^b, Tanka Nath Dhamala^{a,*}^a Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal^b Tribhuvan University, Saraswati Multiple Campus, Kathmandu, Nepal

ARTICLE INFO

Article history:
Received 27 March 2021
Accepted 25 June 2022
Available online xxx

Keywords:
Abstract flow
Switching property
Intermediate storage
Evacuation planning
Asymmetric transit time

ABSTRACT

The abstract network is a generalization of the classical network which is associated with the set of elements and linearly ordered subset of elements, known as paths, that satisfy switching property. Each element is equipped with an integral capacity of two types, movement capacity which transships the flow from the element and storage capacity which stores the flow at the element. In the evacuation process, every individual may not be sent to the final destination due to capacity or time constraints. So, the storage of evacuees at intermediate elements can be a milestone to save the life of people. In this paper, we present polynomial time algorithms to solve maximum static, lexicographic maximum static and maximum dynamic flow problems in an abstract network. We also present the solution procedure to these problems in the abstract contraflow networks having symmetric as well as asymmetric transit times between the pair of adjacent elements.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

Motivation. A flow problem in an abstract network is considered, where elements are capacitated and paths are linearly ordered subset of elements. In this network flow model, paths must satisfy the switching property: when two paths cross at an element then there must be a path that is a subset of the first path up to the crossing element and a subset of the second path after the crossing element. More precisely, flows on crossing paths of an abstract network are diverged to the different directions by switching the flows on non-crossing sides. (e.g., by means of traffic diversion).

A disaster is an uncertain disruption that causes massive loss of human, infrastructure and overall economy. As an example, Gorkha earthquake 2015 in Nepal causes 8020 death, 375 missing and 16,033 injuries together with 416,359 damages of houses and infrastructures, [Government of Nepal \(2015\)](#). Evacuation planning, may be pre and/or post disaster, is not only essential to save the lives and property but also for the quick recovery of the post disaster economy. After any disaster, may be natural or man-made, movement of large number of vehicles towards the safe place causes high congestion on the roads. Similarly, the placement of evacuees at comparatively safer places in an efficient way, which

can not reach to the destination due to some constraints, is another important issue. In this paper, we deal with the transshipment of the evacuees by reducing the crossing effect of the paths and incorporating the settlement of excess flow at comparatively safer intermediate shelters, termed as an abstract flow with intermediate storage.

A classical network deals with the transmission of flow on arcs. The major impact of this condition is that the flow reaching to the head of an arc is allowed to move towards either of its adjacent arcs without considering the crossing effect. Traffic during and evacuation is a special case of flow transmission in which extensive movement of vehicles causes a high congestion. If vehicles are allowed to traverse the nodes (elements) where different paths intersect, the vehicles using one of the intersecting paths may have to wait for a long time unless the flow of the other path is interrupted or stopped. Thus, during usual circumstances, the vehicles' waiting time at intersections may be low, but during a disaster, these waiting times could dramatically increase. Therefore, proper switching of the flow towards the non-crossing sides is essential.

In an abstract network, flow is transmitted along the given path system with switching property and it is not necessary to be used all elements, even the crossing element, in the switched path. The crossing effect is reduced by switching of the paths and the augmentation of flow on feasible paths as in [McCormick \(1996\)](#). Due to abstract paths with switching, direction of paths from the common element are switched to the next path (for example, using traffic control mechanisms like proactive presence of traffic police,

* Corresponding author.

E-mail address: tanka.nath.dhamala@gmail.com (T.N. Dhamala).

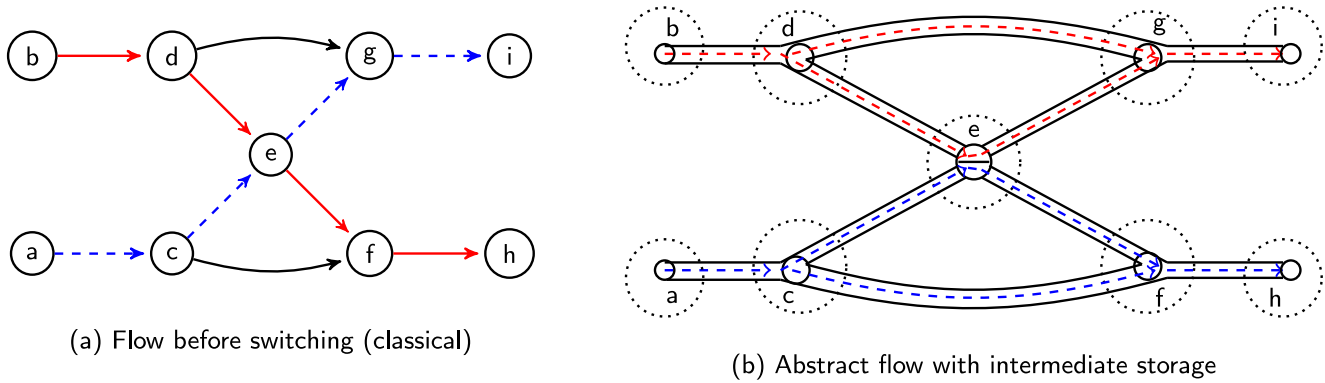


Fig. 1. Flow pattern on classical and abstract networks. Dotted circles represent the appropriate shelters for the storage of flow nearby the intermediate elements.

use of traffic signals, barricades, diversion, traffic lights or signal agents etc.) instead of crossing it. Together with this, the excess flows that can not move forward due to the capacity or time constraints are stored at appropriate shelters, known as intermediate storage, which are the beneficiary task for congestion reduction on the paths and the increment of flow out from the source. Thus an abstract flow with intermediate storage can be an appropriate mathematical model for evacuation planning. Similarly, calculating the flow value in an abstract network is more realistic than in a classical network because delay on the flow transmission due to congestion is not considered in classical networks, which is a major and uncertain factor for flow transmission. This problem is addressed in the abstract network with intermediate storage through switching of the paths and holding the excess flows at the shelters, which also helps to smooth the flow on paths.

Our assumption is that intermediate elements have holding capacity at least the sum of capacity of incoming paths from the predecessor elements. If the number of evacuees out from the source are more than the minimum cut capacity, then the excess of evacuees are to be hold at intermediate shelters (dotted circles in Figure 1(b)) which are comparatively safer than the danger zone (source). Similarly, the flows unable to reach to sink or any other successor element due to insufficient time have to hold at the current intermediate element. In Fig. 1(b), flows reaching to element e are diverged to different paths due to the switching property. Not only in the evacuations, traffic management bureau or any other authorized institutions can transform the classical flow pattern to abstract flow pattern by switching the paths as per the need to manage the peak hour traffic as well. The flow pattern in classical and abstract networks are presented in Fig. 1 with given six paths $P_1 = (a, c, e, g, i)$, $P_2 = (b, d, e, f, h)$, $P_3 = (a, c, e, f, h)$, $P_4 = (b, d, e, g, i)$, $P_5 = (a, c, f, h)$ and $P_6 = (b, d, g, i)$, where flows are transhipped along the crossing of paths P_1 and P_2 in the classical network (Fig. 1(a)) but are switched to P_3 and P_4 together with the flow augmentation in P_5 and P_6 to reduce the crossing effect in the abstract network (Fig. 1(b)). Dotted circles at the intermediate elements represent the desired shelters nearby the intersections and diversion signal at e is represented by the diameter of entire circle.

Literature Review. A network is considered as a graphical representation of a physical scenario that ease to solve different aspects of the real-world problems. With the aspect of objectives, the flow model can be the minimization of time or maximization of the flow, whereas on the basis of the number of objectives, it can be classified as single or multi-objective. With respect to the nature of the time parameter, it can be a discrete or a continuous flow model. Similarly, an evacuation model on the basis of the movement of the vehicles can be a single or a multi-model. If only an homogeneous type of vehicles like cars or buses are used on the evacuation process then it is single model (i.e., uni-model)

whereas the use of two or more non-homogeneous vehicle types together is a multi-model. The classification of the flow model on the behavioral aspect can be classified as microscopic and macroscopic model. The consideration of homogeneous behavior of the group of objects to treat as a single object is the macroscopic model whereas dealing with the individual behavior of each object separately is the microscopic model. We investigate the macroscopic flow model in the abstract network. We refer survey papers and references therein such as Akbari and Salman (2017), Akter and Wamba (2019), Dhamala (2015), Dhamala and Adhikari (2018), Anaya-Arenas, Renaud, and Ruiz (2014), Aronson (1989), Cova and Johnson (2003), Hamacher, Heller, and Rupp (2013), Pascoal, Captivo, and Clímaco (2006) and Kotsireas, Nagurney, and Pardalos (2015) for more detailed descriptions of flow models and algorithms in various aspects.

Network flows are the prominent techniques to solve evacuation planning problems using mathematical models such as maximum flow (Ford & Fulkerson, 1956), quickest flow (Burkard, Dlaska, & Klinz, 1993) and earliest arrival flow (Minieka, 1973) problems. Evacuation with auto base models can be found in Altay and Green (2006), Chen and Miller-Hooks (2008), Hamacher and Tjandra (2002), Moriarty, Ni, and Collura (2007), Schadschneider et al. (2008), Yusoff, Ariffin, and Mohamed (2008) whereas bi-modal formulation of bus and car can be found in Pardalos and Arulselvan (2009). Similarly, Hua, Ren, Cheng, and Ran (2014) studied multi-modal evacuation planning problem and Purba, Kontou, and Vogiatzis (2021) studied the evacuation network modeling for alternative fuel vehicles.

Modeling on network flows with intermediate storage, introduced by Pyakurel and Dempe (2020), address the important problem of evacuation planning. The corresponding solutions are very relevant in large scale disaster management. In evacuation models, one wishes to shift evacuees from danger zones (sources) to safety places (sinks) as quickly and efficiently as possible. Most of the flow over time models used in evacuation planning are based on the flow conservation constraints, that is, the inflow must be equal to the outflow at each element (node), except at the source and the sink. In contrast to this, for an evacuation planning problem with intermediate storage, the inflow may be greater than the outflow at intermediate elements. By holding the excess flow not reaching to the destination at comparatively safer intermediate elements, this problem type maximizes the number of evacuees leaving the danger zone.

Not only in evacuation planning, intermediate storage is highly applicable for different demand-supply chains like commodity supply, electricity distribution and water supply. Production houses used to store their products in major cities to make the supply system smooth. Electricity is stored at substations to regulate the flow. The storage of hazards associated liquids in intermedi-

ate bulk containers with practical measures on the design, construction and operation of storage areas is very essential to regulate the supply with safety from unexpected accidents. Similarly, for uniform distribution of water supply, it is stored at reservation tanks and supplied to the customers, [Kurian, Chinnusamy, Natarajan, Narasimhan, and Narasimhan \(2018\)](#).

[Hoffman \(1974\)](#) introduced the concept of abstract flow by reviewing the first proof of the max-flow-min-cut theorem of [Ford and Fulkerson \(1956\)](#) with flows in terms of paths rather than on arcs. [McCormick \(1996\)](#) provided a polynomial time algorithm by using an oracle where the input is an arbitrary subset of elements whose output is either a path contained in that subset or states that no such path exists. He used the augmenting path structure satisfying the complementary slackness condition: every positive path meets the cut set exactly at one common element and every element of the cut is saturated. [Martens and McCormick \(2008\)](#) extended the result of [McCormick \(1996\)](#) in more general case with additional attribute of weight on paths. Unsplittable and k -splittable abstract network flows can be found in the thesis of [Martens \(2007\)](#) for single as well as multi-commodity flows. Similarly, [Kappmeier \(2015\)](#) presented a polynomial time algorithm for lexicographic abstract maximum flow and used it to prove the existence of abstract earliest arrival flow in his PhD thesis.

At the time of evacuation, paths towards the danger zones are almost empty. So, the optimal use of empty paths by reversing their direction is a prominent way to maximize the flow and minimize the evacuation time. Reflecting this idea in the network optimization, contraflow is one of the most useful and widely used techniques in evacuation planning problems, where flipping of the orientation of arcs towards the destination node is used to increase the outbound capacity of arcs and reduce the time horizon. Heuristic, analytic and simulation techniques are used by different authors at different time to solve the problems with contraflow configuration. [Kim, Shekhar, and Min \(2008\)](#) presented macroscopic models by incorporating multiple sources, road capacity constraints, congestion and scalability. Different evacuation models with and/or without contraflow and their solution strategies can be found in [Amideo, Scaparra, and Kotiadis \(2019\)](#), [Arulselvan \(2009\)](#), [Dhamala, Pyakurel, and Dempe \(2018\)](#), [Pyakurel and Dhamala \(2016\)](#), [Pyakurel, Nath, Dempe, and Dhamala \(2019a\)](#), [Rebennack, Arulselvan, Elefteriadou, and Pardalos \(2010\)](#) and references therein.

By integrating the concept of abstract flow and contraflow, [Pyakurel, Dhamala, and Dempe \(2017\)](#) introduced the concept of the continuous maximum abstract contraflow problem and presented polynomial time algorithms for the static as well as the dynamic cases. [Pyakurel, Nath, and Dhamala \(2019b\)](#) introduced the partial contraflow approach in abstract network by saving unused capacities of elements and presented efficient algorithms for static, lexicographically maximum static, maximum dynamic and earliest arrival partial contraflow problems in an abstract network. Similarly, incorporating the contraflow configuration in network flow with intermediate storage, [Pyakurel and Dempe \(2020\)](#) introduced the maximum dynamic contraflow problem with intermediate storage and presented a polynomial time algorithm to solve the problem. [Weitzel and Glock \(2018\)](#) presented a systematic review of the literature on electric energy storage systems for balancing energy demand and supply. [Pyakurel and Dempe \(2021\)](#) investigated universal maximum dynamic flow and contraflow problems with intermediate storage and presented efficient algorithms in two-terminal series parallel networks. The solution of maximum multi-commodity flow problem with intermediate storage and the contraflow solutions with symmetric as well as asymmetric transit times can be found in [Khanal, Pyakurel, and Dhamala \(2021\)](#).

Congestion minimization is one of the important issues for the evacuation management. [Choi, Hamacher, and Tufekci \(1988\)](#) pre-

sented building evacuation models by network flow with side constraints. The flow dependent capacity constraints on arcs are used to avoid the congestion on paths which are considered as side constraints. A review of optimization models for pedestrian behaviors and crowd dynamics can be found in [Vermuyten, Beliën, De Boeck, Reniers, and Wauters \(2016\)](#). They have described the characteristics of different models like continuum, network-based, cellular automata, agent-based, social-force and game-theoretic models relating to the congestion minimization for the pedestrian evacuation. [Zambrano, Huertas, Segura-Durán, and Medaglia \(2020\)](#) presented a mesoscopic model that combines a network-oriented macroscopic optimization (to determine the evacuation flow) with a microscopic discrete-event simulation model (to evaluate the congestion effect on the dynamics of the evacuation flows). Recently, [Huertas and Van Hentenryck \(2022\)](#) presented zone based evacuation by assigning single evacuation path to the corresponding zone. They used convergent paths at intersections and non-preemptive schedules to ensure the evacuation process without interruptions. Our models are macroscopic model based on the network optimization with constant capacity on arcs and do not include constraints to explicitly model the effect of congestion. However, switching of paths at crossings, intermediate storage of excess flow and contraflow configuration are key tools to manage the congestion.

Research Gap. At the time of evacuation, transmission of maximum evacuees from the danger zone to the comparatively safer places through congestion free paths is very essential. As abstract network flow eliminates the crossing and merging effects of the paths, flow with intermediate storage is the major research gap in the literature of abstract flow. To fulfill the research gap, this paper mainly emphasizes the flow transmission in abstract network with intermediate storage. The focus is also given to fulfill the gap of an abstract contraflow problem with intermediate storage for the optimal use of empty paths.

Our Contribution. In this paper, we introduce the concept of intermediate storage on an abstract network topology for the first time, where nodes are taken as elements. We introduce the maximum static, lexicographic maximum static, maximum dynamic flow problems, and present the flow models and the solution strategy with intermediate storage for an abstract network. As per our knowledge, all of these works are introduced for the first time. For the uniqueness of the solution, we assume that the storage capacity of each intermediate element is at least the sum of incoming capacities from its left elements through the paths. Capacities of the source and the sink are considered as sufficiently large. As the solution strategy, we fix the first priority to the sink and the priority of intermediate elements are set according to the maximum of shortest distance from the source. Flows are stored according to their priority order. To satisfy the flow conservation constraints, dummy ports are created to absorb the excess flow of intermediate elements with respective storage capacity and zero cost (transit time). Denoting the set of dummy ports and a sink as a super sink, the problem is transformed to a single source multi-sink abstract flow problem. In the case of a lexicographic abstract flow problem, the priority of given multiple sinks is followed by the priority of intermediate elements.

To obtain the maximum static, lexicographic maximum static and maximum dynamic flow with intermediate storage in an abstract network, we use the lexicographic static flow algorithm of [Kappmeier \(2015\)](#) without intermediate storage. We present polynomial time algorithms to solve these problems. Another contribution of the paper is that we present temporally repeated flow to obtain the maximum dynamic abstract flow with intermediate storage if the storage capacity of each intermediate element is sufficient (i.e., T times the sum of incoming capacities from its left elements through the paths). We also provide the generic algo-

rithm which solves all these problems for a contraflow network with symmetric as well as asymmetric transit times. By using natural transformation, we present the solution procedure to solve dynamic flow and contraflow problems in continuous time setting also.

Organization of the Paper. We organize the paper as follows. Section 2 provides basic notations that we use throughout the paper. In Section 3, we formulate an abstract maximum static flow problem, its mathematical model and present a polynomial time algorithm to solve it. We also present a lexicographic abstract maximum static flow problem and its solution strategy. Abstract maximum dynamic flow problem, its mathematical model and a polynomial time algorithm are presented in Section 4. For a contraflow configuration, a generic algorithm to solve these problems is presented in Section 5. We use natural transformation to transform the discrete time solution to continuous one in Section 6. The paper is concluded in Section 7.

2. Basic notations

In this section, we give basic mathematical notations that are used throughout the paper. Let $\mathcal{N} = (E, \mathcal{P})$ be a network topology with finite set of elements E and the collection of paths

$$\mathcal{P} = \{P \subseteq E : P \text{ has a linear order } <_P \text{ of elements in } P\} \subseteq 2^E.$$

Here, \mathcal{P} represents the set of all source-sink, i.e. $s - t$, paths P as well as intermediate paths $P_{[s \rightarrow e]}$ from $s \in E$ to $e \in E$. Each element $e \in E$ has the non-negative integral movement capacity $u_e : E \rightarrow \mathbb{Z}^+$ which is used to send flow from the element e to its adjacent elements and the storage capacity $v_e : E \rightarrow \mathbb{Z}^+$ which is used to hold flow at e . The order of elements in the path $P \in \mathcal{P}$ is denoted by $<_P$. We say that a is the left of e on P if $a <_P e$ and right of e if $a >_P e$. Similarly, $e \in P$ is said to be the leftmost or first (rightmost or last) element of P if there does not exist a in P such that $a <_P e$ ($a >_P e$). For $s - t$ path P , source node s is the leftmost element and sink node t is the rightmost element. We denote the set of intermediate elements by $E_I = E \setminus \{s, t\}$.

Network $\mathcal{N} = (E, \mathcal{P})$ is an abstract network if it satisfies the switching property: $\forall P, Q \in \mathcal{P}$ and an intermediate element $e \in P \cap Q$, $\exists R \in \mathcal{P}$ such that $R \subseteq P \times_e Q$, where

$$P \times_e Q = \{a \in P : s \leq_P a \leq_P e\} \cup \{a \in Q : e \leq_Q a \leq_Q t\}.$$

A similar definition for switched path $R \subseteq Q \times_e P$ can be obtained. For simplicity, we use the notations

$$P_{[s \rightarrow e]} = \{a \in P : s \leq_P a \leq_P e\} \text{ and } P_{[e \rightarrow t]} = \{a \in P : e \leq_P a \leq_P t\}$$

to represent the elements on path P from source s up to e and that begin from e up to sink t , respectively. Similarly,

$$P_{[s \rightarrow e]} = \{a \in P : s \leq_P a <_P e\}, \text{ and } P_{[e \rightarrow t]} = \{a \in P : e <_P a \leq_P t\}$$

represent the elements on path P that are left of e and right of e , respectively. If P and Q are two paths both containing e_1 and e_2 , then it is possible to have $e_1 <_P e_2$ but $e_1 >_Q e_2$.

Throughout the paper, we assume that the source and sink have sufficiently large storage capacity, i.e., $v_s = v_t = \infty$ and that of intermediate elements are finite. The source and intermediate elements have finite movement capacities (i.e., $u_s < \infty$) and that of the sink is zero (i.e., $u_t = 0$). If the incoming movement capacity of an intermediate element $e \in E_I$ is more than the outgoing movement capacity, then the excess flow is used to store at e . Moreover, the storage capacity of $e \in E_I$ should be $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$. Furthermore, the incoming and outgoing movement capacities of source and sink elements are zero, respectively, except for the contraflow network.

In this paper, our assumption is that the elements being far from the source element are safer than nearer one. Also, the evacuation scenario is of single source single sink, except in the lexicographic abstract static flow problem, with possibility of intermediate storage. In case, if there are some unsafe intermediate elements, we consider them as the virtual source elements which turns the problem to the multiple sources. To produce it into single source, we create a super source with storage capacity, movement capacity and transit time (cost) as zero. Due to the zero transit time of unsafe intermediate elements from the super source, flow can not be stored at these intermediate elements.

3. Abstract static flow with intermediate storage

This section deals with the formulation of problems, mathematical models and their solution strategies for abstract maximum static and lexicographic abstract maximum static flows with intermediate storage. We discuss the abstract maximum static flow problem with intermediate storage for single source single sink network in Section 3.1 and solve it by creating dummy ports for each intermediate element. The prioritized dummy ports with sink is considered as a set of multiple sinks which reduce the problem into single source multi-sink problem. Similarly in Section 3.2, we introduce the lexicographic abstract maximum static flow problem with intermediate storage for a given single source multi-sink network. For the solution procedure, priority of given sinks are fixed in the first stage and then priority of intermediate elements in the second stage. In the third stage, we merge first and second stages to form a super sink.

As both problems are transformed into single source multi-sink structure, we use the lexicographic abstract maximum flow algorithm of Kappmeier (2015) to solve them. The quick review of this algorithm is as follows: For a given network $\mathcal{N}(E, \mathcal{P})$ with single source s multi-sink $\{t_i\} i = 1, \dots, r$, first keep the elements of sink in compatible sequence and construct a modified network (E, \mathcal{P}_i) with the increasing subset of paths $\mathcal{P}_i \subseteq \mathcal{P}$. For each iteration $i = 1, \dots, r$, flow x_{t_i} is computed by using the augmenting structure of McCormick (1996), which is used to augment the feasible set of path flows to a set with strictly large total flow value by polynomial number of calls to the oracle.

3.1. Abstract maximum static flow

In this subsection, we introduce the abstract maximum static flow problem with intermediate storage, give its mathematical model and present a polynomial time algorithm to solve it. By using abstract network topology, we aim to send the maximum flow from the source element to the sink element and the excess flow is stored at intermediate elements.

Problem 1. For the given abstract static network $\mathcal{N} = (E, \mathcal{P})$, the abstract maximum static flow problem with intermediate storage deals with maximization of flow leaving the source element that is to be sent to the sink element via $s - t$ paths $P \in \mathcal{P}$ by allowing the maximum storage of excess flow at intermediate elements e via paths $P_{[s \rightarrow e]} \forall e \in E_I$ with storage capacity $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$.

Mathematical Model. Let $\mathcal{N} = (E, \mathcal{P})$ be an abstract $s - t$ network with path-flow $x^P : P \rightarrow \mathbb{R}^+$. Every path-flow x^P induces a flow through each element, denoted by $x_e^P = \sum_{P \in \mathcal{P}: e \in P} x^P$. A path-flow x^P is feasible if and only if $x_e^P \leq u_e$ and $x^P \geq 0$ for all $e \in E$. An element e is said to be saturated with respect to x if $x_e^P = u_e$. We denote $x_e^{P, out} = \sum_{P \in A_e} x^P$ and $x_e^{P, in} = \sum_{P \in B_e} x^P$ as the total outflow from e and the total inflow into e , respectively, where A_e and B_e represent the set of outgoing paths from e and incoming paths into e . Let $c_e : E \rightarrow \mathbb{Z}^+$ be the cost of transmission of flow per unit from e to its right element so that $c_P = \sum_{e \in P} c_e$.

Hoffman (1974) generalized the max-flow-min-cut theorem of Ford and Fulkerson (1962) for abstract network flow where the storage of flow at intermediate elements is prohibited. Due to bottleneck flow on each path, pushing the full movement capacity of flow from the source element greater than the minimum cut capacity is impossible. To deal with this problem, Pyakurel and Dempe (2020) introduced the concept of intermediate storage for classical network flow. Using this concept in abstract network flow, we can push maximum flow outward from the source where only flow with minimum cut capacity reaches to the sink and the rest of the flow can be stored at intermediate elements. Similarly, due to the switching property at common element $e \in E_I$ of two paths P and Q , if the inflow at e through $P_{[s \rightarrow e]}$ is greater than outflow from e through $Q_{[e \rightarrow t]}$ then the excess flow can be stored at e .

We define the flow function $\hat{x}_e : E_I \rightarrow \mathbb{R}^+$ as the excess flow stored at element $e \in E_I$ obtained by the difference of inflow and outflow. The linear programming formulation of abstract static network flow with intermediate storage is

$$\max \sum_{P \in \mathcal{P}} x^P + \sum_{e \in E_I} \hat{x}_e \quad (1)$$

$$\text{s.t.} \sum_{P \in \mathcal{P}: e \in P} x^P \leq u_e \quad \forall e \in E \quad (2)$$

$$x_e^{P, \text{in}} - x_e^{P, \text{out}} = \hat{x}_e \quad \forall e \in E_I \quad (3)$$

$$0 \leq \hat{x}_e \leq v_e \quad \forall e \in E_I \quad (4)$$

$$x^P \geq 0 \quad \forall P \in \mathcal{P} \quad (5)$$

Eq. (1) is an objective function that wants to maximize the total flow reaching at sink t and the excess flow stored at intermediate elements. The capacity constraint of each element is represented in Eq. (2) and the excess flow is presented in Eq. (3). The non-conservation of flow is represented by the left inequality of Eq. (4) and its right inequality represents that the excess flow is bounded by the storage capacity of e . Similarly, Eq. (5) represents the non-negativity of the flow on each path. As a precondition for the uniqueness of the solution, the lower bound of storage capacity is considered as $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a, \forall e \in E_I$.

Due to different nature of the disasters, range of their affected regions may vary. We consider the affected area as a danger zone (source) and rest of the places as comparatively safer. The priority order of intermediate elements (shelters) can be considered with respect to different aspects like security, accessibility, availability of physical infrastructure and basic needs, possibility to deliver medical and other services etc. In some disasters like tsunami, industrial accidents, fires, nuclear explosions etc., the regions much farther away from the source of disaster are considered more safe. In this paper, we consider the priority of intermediate elements with respect to the distance from the source. Though elements far from the source are considered more safer than nearer one because of which flows try to move as far as possible, but due to different barriers (constraints) flows may have to be hold nearer as well.

As in Pyakurel and Dempe (2020), due to the uniqueness of the sink in a given network, first priority is given to the sink to transship as much flow as possible. To store the excess flow at intermediate elements, the priority ordering is set as follows: For each $e \in E_I$ with storage capacity $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$, calculate the shortest distance $d_{P_{[s \rightarrow e]}}$ from s by using the algorithm of Dijkstra (1959). Path with minimum cost is considered as the shortest path and the priority is given to the farthest element. That is, $\forall e_1, e_2 \in E_I$ if $d_{P_{[s \rightarrow e_1]}} > d_{P_{[s \rightarrow e_2]}}$, then e_1 has higher priority than e_2 and this is denoted by $e_1 > e_2$.

We create dummy port e' for each prioritized element $e \in E_I$ with cost $c_{P_{[e \rightarrow e']}} = d_{P_{[e \rightarrow e']}} = 0$ and capacities $u_{[e \rightarrow e']} = v_e = v_{e'}$, where $u_{P_{[e \rightarrow e'']}}$ and $c_{P_{[e \rightarrow e'']}}$ are the movement capacity and cost from e to e' , respectively. Every dummy port e' has the same priority order as e has. The collection of dummy ports $D' = \cup \{e'\}$ together with sink t forms a modified network $\mathcal{N}' = (E', \mathcal{P}')$ with single source s and multiple sinks $D = D' \cup \{t\}$, where $E' = E \cup D$ and $\mathcal{P}' = \mathcal{P} \cup \{P_{[s \rightarrow e']}\}$.

Let e_1, \dots, e_r be r intermediate elements and $t > e_1 > e_2 > \dots > e_r$ be priority order of elements in E . By denoting t as e'_0 , dummy port e'_i of each e_i forms the priority order $e'_0 > e'_1 > \dots > e'_r$ in D . A sequence of elements is said to be compatible if the elements respect their ranks. For the priority ordered set of sinks $D = \{e'_0 > e'_1 > \dots > e'_r\}$, more priority is given to the one in left than in right and satisfies the condition

$$P \in \mathcal{P}', e'_i \neq e'_j \in P : i < j \Rightarrow e'_i \leq_P e'_j,$$

so that D forms a compatible sequence of sinks.

For this compatible sequence of sinks, we define the collection of paths as

$$\begin{aligned} \mathcal{P}'_0 &= \{P_{[s \rightarrow e'_0]}\} \\ \mathcal{P}'_i &= \mathcal{P}'_{i-1} \cup \{P_{[s \rightarrow e'_i]}\} \quad \text{for } i = 1, \dots, r. \end{aligned}$$

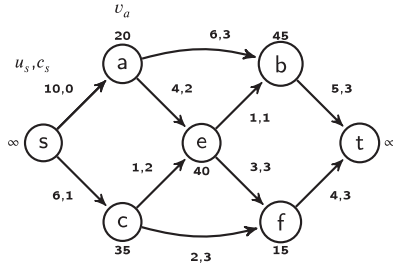
Here, each abstract path-system (E', \mathcal{P}'_i) for $i = 0, 1, \dots, r$ contains the paths starting from s and ending at e'_i . The network topology $\mathcal{N}'_i = (E', \mathcal{P}'_i)$ for all such path-system forms the abstract network satisfying the switching property. The solution obtained in this network is transformed to the original network by removing dummy ports and dummy paths. Flows to dummy ports are shifted to their corresponding intermediate elements.

Example 1. Consider a network with element set $E = \{s, a, b, c, e, f, t\}$ and set of paths $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ where, $P_1 = (s, a, e, f, t)$, $P_2 = (s, c, e, b, t)$, $P_3 = (s, a, b, t)$, $P_4 = (s, c, f, t)$, $P_5 = (s, a, e, b, t)$ and $P_6 = (s, c, e, f, t)$. Fig. 2 represents a network with storage capacity at each element together with movement capacity and cost in between pair of elements. Let P_1, P_2, P_3 and P_4 be four positive paths. As e is common element in paths P_1 and P_2 , the switched paths are $P_5 = P_1 \times_e P_2$ and $P_6 = P_2 \times_e P_1$. The minimum distance of each intermediate elements are $d_{P_{[s \rightarrow f]}} = 4$, $d_{P_{[s \rightarrow b]}} = 3$, $d_{P_{[s \rightarrow e]}} = 2$, $d_{P_{[s \rightarrow c]}} = 1$ and $d_{P_{[s \rightarrow a]}} = 0$ so that the priority ordering is $t > f > b > e > c > a$. For each intermediate element, dummy port is created with dummy path of zero cost and the movement capacity equal to the storage capacity of original element. Also, the storage capacity of dummy port is taken as the storage capacity of original element. The set of dummy ports $D' = \{f', b', e', c', a'\}$ together with sink element forms a compatible sequence denoted by $D = \{t, f', b', e', c', a'\}$ (See in Fig. 3).

Lemma 3.1 (Kappmeier (2015)). For an abstract network $\mathcal{N} = (E, \mathcal{P})$ and a compatible sequence of sinks e'_0, e'_1, \dots, e'_r , the abstract path-system $\mathcal{N}'_i = (E', \mathcal{P}'_i)$ for each $i = 0, 1, \dots, r$ is an abstract network.

Now, we present Algorithm 1 to solve Problem 1 by using the lexicographically maximum flow algorithm of Kappmeier (2015) (cf. Page 167, Algorithm 6.2) for a single source multi-sink abstract network \mathcal{N}' , which solves the abstract maximum static flow problem with intermediate storage in polynomial time.

For any two flows x and y , we say that x is lexicographically greater than y , and denote $x \geq^L y$, if either $(x_{e'_i}^{\text{in}})_l > (y_{e'_i}^{\text{in}})_l$ and $(x_{e'_j}^{\text{in}})_{j-1} = (y_{e'_j}^{\text{in}})_{j-1}$ holds for some $l \in \{0, 1, \dots, r\}$ and $j = 1, \dots, l$, or $(x_{e'_j}^{\text{in}})_j = (y_{e'_j}^{\text{in}})_j$ holds for all $j = 0, 1, \dots, r$. The maximum flow \bar{x} with lexicographic order \geq^L among all feasible abstract flows x is a lexicographic abstract maximum flow and we denote it as $\bar{x} \geq^L x$ for all x .



- $s - t$ paths:
- $P_1 = (s, a, e, f, t)$ $u_{P_1} = 3, c_{P_1} = 8$
 - $P_2 = (s, c, e, b, t)$ $u_{P_2} = 1, c_{P_2} = 7$
 - $P_3 = (s, a, b, t)$ $u_{P_3} = 5, c_{P_3} = 6$
 - $P_4 = (s, c, f, t)$ $u_{P_4} = 2, c_{P_4} = 7$
 - $P_5 = (s, a, e, b, t)$ $u_{P_5} = 1, c_{P_5} = 6$
 - $P_6 = (s, c, e, f, t)$ $u_{P_6} = 1, c_{P_6} = 9.$

Fig. 2. Network with movement capacity, cost between elements and storage capacity at elements.

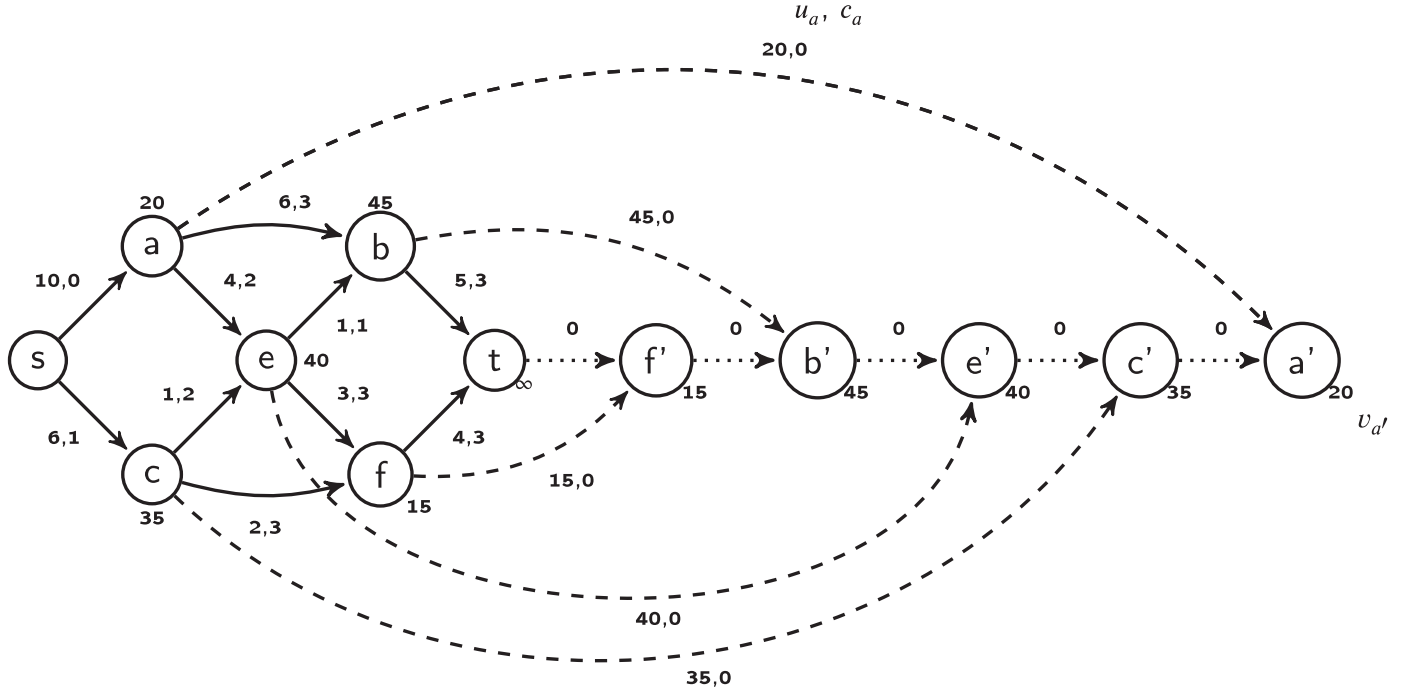


Fig. 3. Reconfigured network with a compatible sequence of sinks (dummy ports) after priority ordering on Fig. 2.

Algorithm 1: Abstract maximum static flow algorithm.

- Input :** Given abstract static network $\mathcal{N} = (E, \mathcal{P})$.
Output: Abstract maximum static flow with intermediate storage on \mathcal{N} .
1. For each $e \in E_I$ with $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$, compute the shortest distance $d_{P[s \rightarrow e]}$ by using Dijkstra's algorithm.
 2. Fix the priority order as $t > e_1 > \dots > e_r$ with first priority to the sink t and priority for intermediate elements as $d_{P[s \rightarrow e_i]} > d_{P[s \rightarrow e_{i+1}]} \implies e_i > e_{i+1}$, for $i = 1, \dots, r - 1$.
 3. Construct the modified network $\mathcal{N}' = (E', \mathcal{P}')$ with single source s and compatible sequence of multiple sinks with dummy ports $D = \{t = e'_0, e'_1, \dots, e'_r\}$, where $E' = E \cup D$ and $\mathcal{P}' = \mathcal{P} \cup \{P[s \rightarrow e'_1]\}$.
 4. Compute the lexicographic abstract maximum static flow with priority ordering in Step~(2) according to Kappmeier (2015).
 5. Transform the solution to the original network \mathcal{N} by removing dummy ports and dummy paths.

Theorem 3.2. There exists an abstract flow x_i in $\mathcal{N}'_i = (E', \mathcal{P}'_i)$ for each $i = 0, 1, \dots, r$ that is a lexicographically maximum.

Proof. When $i = 0$, $\mathcal{N}'_0 = \mathcal{N}$ is a single source single sink abstract network. By taking the initial flow as zero flow and using an aug-

menting structure of McCormick's algorithm, it provides an abstract maximum flow. For further proof we use induction. We assume that for some $i < r$, flow x_i obtained by taking initial flow x_{i-1} and using augmenting structure of McCormick's algorithm is lexicographic abstract maximum flow (see McCormick, 1996). We aim to show that x_{i+1} is also lexicographic abstract maximum flow. As the sequence e'_0, e'_1, \dots, e'_r of sinks is compatible, augmenting structure of McCormick assures that the inflow to the sink element is not reduced and so x_{i+1} is maximum. If possible, let us assume that x_{i+1} is not lexicographic abstract maximum flow in the abstract network $\mathcal{N}'_{i+1} = (E', \mathcal{P}'_{i+1})$. Then there exists a flow x^* which sends more flow to the sink e'_k for some $k \in \{0, 1, \dots, i\}$. For each $P \in \mathcal{P}'_i$, define the restricted flow \tilde{x} by $\tilde{x}^P = x^{*P}$. For \tilde{x} and x^* , incoming flow at sink e'_k is the same and \tilde{x} is a feasible abstract flow in $\mathcal{N}'_i = (E', \mathcal{P}'_i)$ which sends more flow to sink e'_k than x_i . This contradicts to x_i being a lexicographically maximum. \square

Theorem 3.3. Algorithm 1 computes an abstract maximum static flow with intermediate storage in $\mathcal{N} = (E, \mathcal{P})$.

Proof. The algorithm starts with determining the shortest distance of intermediate elements having storage capacity at least the sum of movement capacity from left elements through $s - t$ paths. We fix the priority order of elements as follows: The first priority is given to the sink element. The second priority is for the farthest element from the source, the third priority for the second farthest element and so on. Prioritized elements are shifted to the dummy

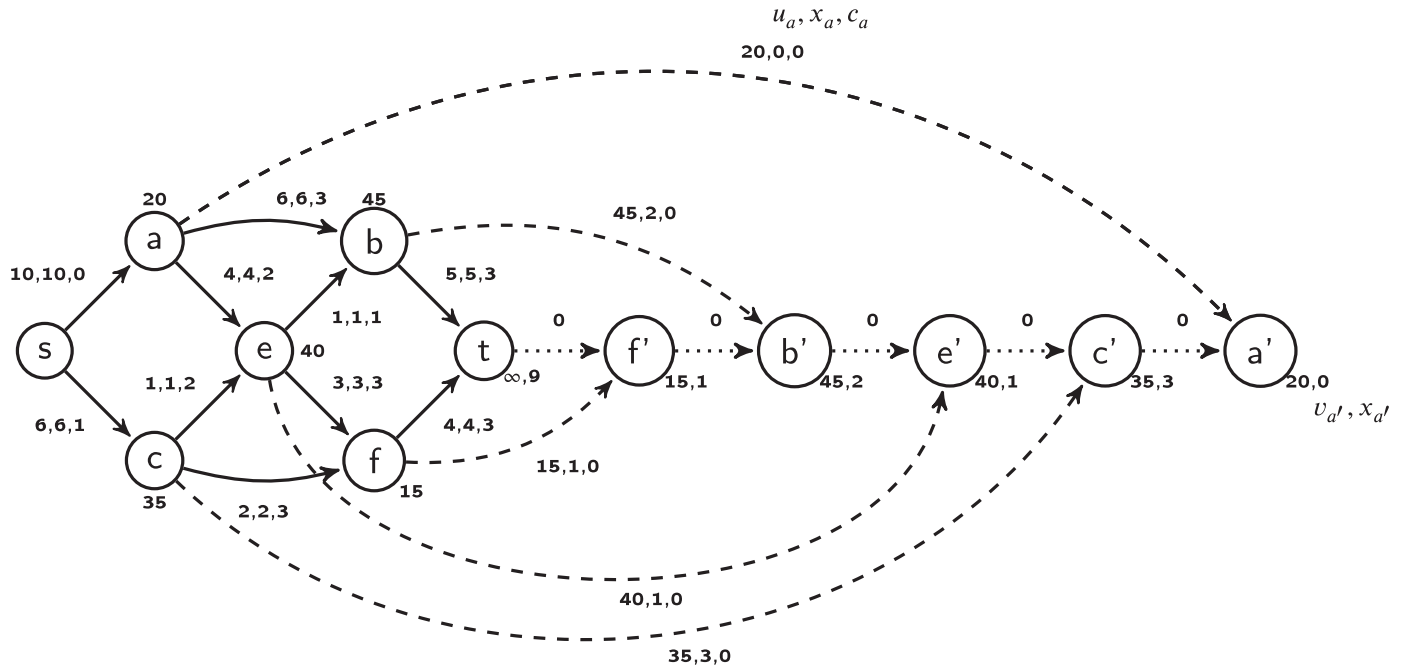


Fig. 4. Storage of flow at dummy ports and sink before switching the paths (classical).

ports and form a compatible sequence with the same priority ordering.

For the network with abstract path-system $\mathcal{N}'_i = (E', \mathcal{P}'_i)$, the set inclusion holds i.e., $\mathcal{N}'_i \subseteq \mathcal{N}'_{i+1}$, for $i = 0, 1, \dots, r-1$. Theorem 3.2 assures the existence of a lexicographic abstract maximum flow in \mathcal{N}'_i which is computed using Kappmeier's algorithm. At last, we transform the solution obtained in \mathcal{N}' to the original network \mathcal{N} by removing dummy ports and sending back the flows to their respective intermediate elements. Thus the solution obtained is an abstract maximum static flow with intermediate storage in $\mathcal{N} = (E, \mathcal{P})$. \square

It is to be noted that the maximum flow entering at sink t is an abstract maximum static flow without intermediate storage.

Corollary 3.4. Abstract maximum static flow problem with intermediate storage can be solved in polynomial time by using Algorithm 1.

Proof. Since the shortest distance of each intermediate element can be obtained in $O(|E|^2)$ times and priority ordering with respect to the distance can be calculated in linear time, Steps 1 and 2 can be solved in polynomial time. After fixing the priority order of intermediate elements, the problem is transformed to a single source and multi-sink problem and by using Kappmeier (2015), Step 4 can be obtained in polynomial time. Similarly, a transformation of the solution to the original network can be obtained in linear time. So, Algorithm 1 solves an abstract maximum static flow problem with intermediate storage in polynomial time complexity. \square

Example 2. Here, we continue Example 1 to find the static solution with intermediate storage before and after the switching of paths. In Example 1, we created a set of dummy ports $D' = \{f', b', e', c', a'\}$ and a compatible sequence $D = D' \cup \{t\}$. Now, we obtain the solution by using lexicographically maximum static flow and restore it at sink and dummy ports with priority order. Finally, dummy ports and dummy paths are removed to get a maximum static flow with intermediate storage.

Fig. 4 represents the flow with intermediate storage in reconfigured classical network (without switching of paths) where flow is

transmitted through six paths P_1, P_2, P_3, P_4, P_5 and P_6 by extending to respective compatible dummy ports. The numbers in between the elements represent the movement capacity, flow and cost of the path segment. Similarly in reconfigured abstract network, flow is transmitted through four paths P_3, P_4, P_5 and P_6 by extending to respective compatible dummy ports after switching of paths P_1 and P_2 which is shown in Fig. 5 and also in Table 1. Being minimum cost path, flow is first send to the path $P_3 = (s, a, b, t)$. As the compatible sequence of sink and intermediate elements is $\{t, b', a'\}$, the flow of 5 units is transmitted to the sink t through this path. Now, possible maximum excess flow is send to b' through two possible paths (s, a, b, t, b') and (s, a, b, b') . As the first path (s, a, b, t, b') is already saturated, only 1 unit of excess flow can be reached to b' . Similarly, the 4 units flow at a is diverged towards e , no excess flow is stored at a' . This process is simultaneously used for all possible paths. At last, the flow at each dummy port is transformed to the respective element.

It is to be noted in Fig. 5 that the 5 units of flow reaching e through two paths (4 from a and 1 from c) are not been merged at e but diverged to non-crossing sides by some traffic signal mechanism; and excess flows are stored at appropriate shelter. Otherwise, it becomes a classical network flow. Out of the 4 units of flow reaching e from a , 1 unit is switched to b due to the switching property, and the remaining 3 units are stored as excess flow. On the other hand, the 1 unit of flow reaching e from c is switched to f with no excess flow. Thus, two flows from different paths are not crossing at e but diverging from the intersection.

The total amount of flow pushed from the source element before switching the paths (i.e. classical network) without intermediate storage is 9 units whereas with intermediate storage is 16 units (Fig. 6(a)). Fig. 6(b) shows the flow at sink and excess flow at intermediate elements after switching the paths. Total amount of flow pushed from source element without intermediate storage is 8 units whereas with intermediate storage is 16 units, which emphasize the importance of intermediate storage in the abstract network. Table 1 represents the flow pattern in each path with intermediate storage.

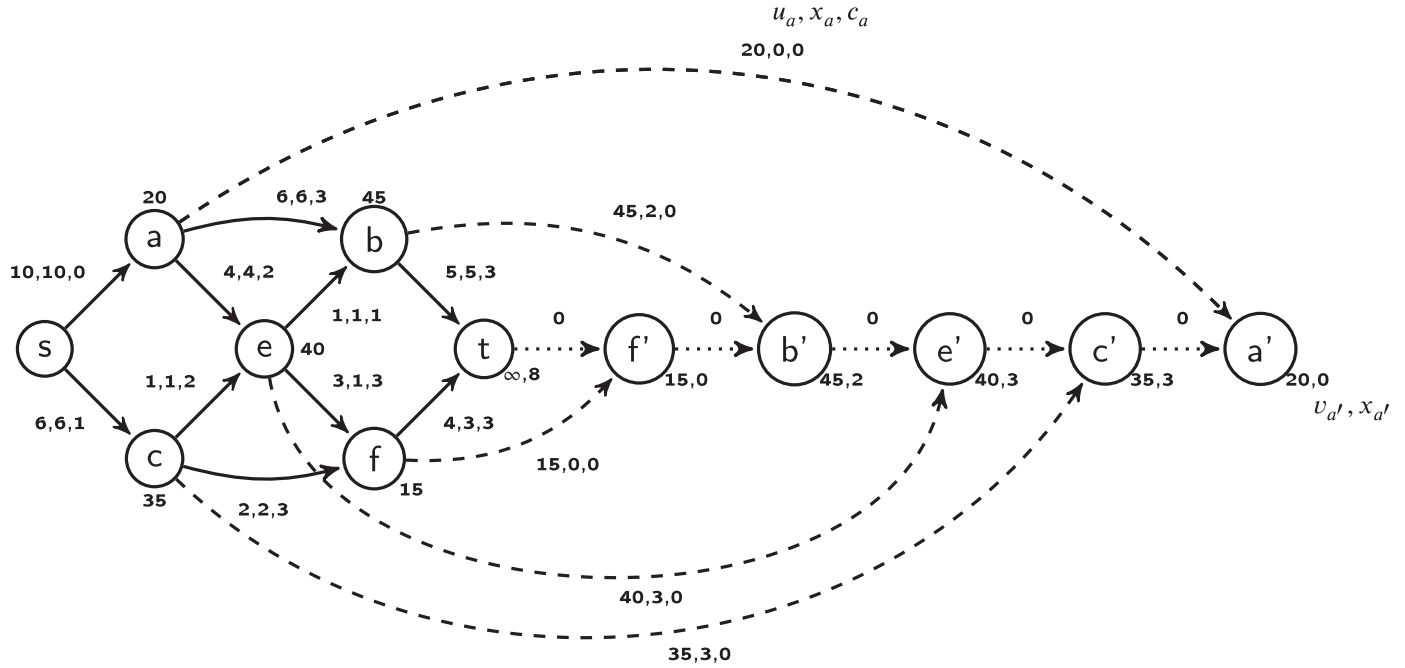


Fig. 5. Storage of flow at dummy ports and sink after switching the paths (abstract).

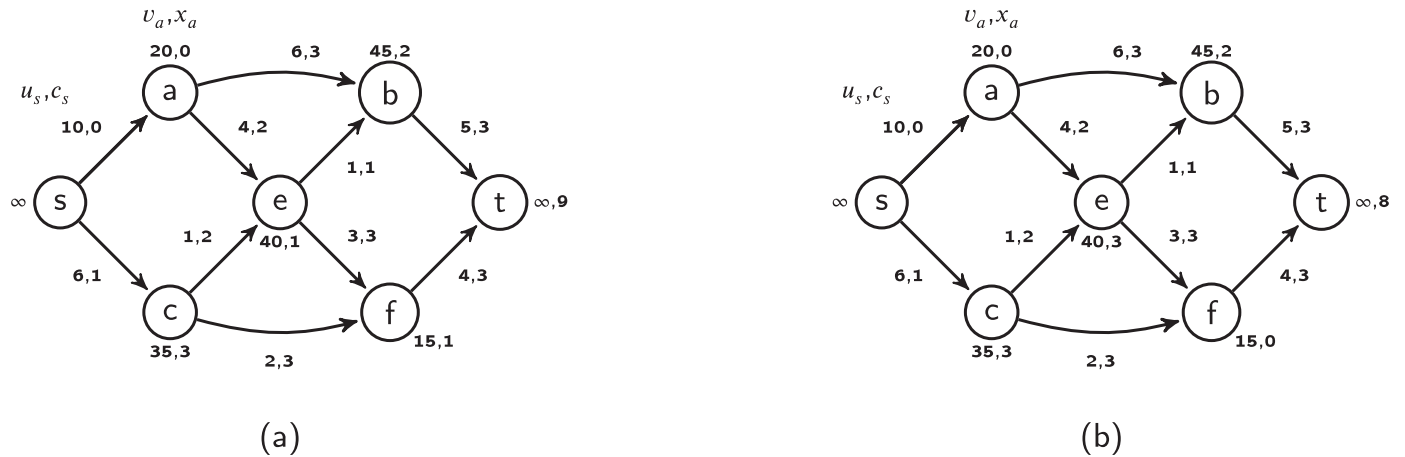


Fig. 6. (a) Solution before switching the paths (b) Solution after switching the paths.

Table 1
Classical and abstract path-flows with intermediate storage.

Intermediate storage: Classical						Intermediate storage: Abstract							
Path	a	c	e	b	f	t	Path	a	c	e	b	f	t
P_3	0	×	×	1	×	5	P_3	0	×	×	1	×	5
P_5	0	×	0	1	×	0	P_5	0	×	3	1	×	0
P_2	×	3	1	0	×	0	P_4	×	3	×	×	0	2
P_4	×	0	×	×	0	2	P_6	×	0	0	×	0	1
P_1	0	×	0	×	1	2							
P_6	×	0	0	×	0	0							
Total flow	0	3	1	2	1	9	Total flow	0	3	3	2	0	8

×= element not used.

3.2. Lexicographic abstract maximum static flow

In the previous Section 3.1, we derived the polynomial time solution procedure for an abstract maximum static flow problem with intermediate storage by using the concept of lexicographic abstract maximum flow without intermediate storage. In this section, we introduce the lexicographic abstract maximum flow prob-

lem with intermediate storage in a single source multi-sink network and present a polynomial time algorithm to solve it. Let $\mathcal{N} = (E, \mathcal{P})$ be an abstract $s - D$ network with $E = \{s\} \cup E_I \cup D$, where $D = \{t_1, \dots, t_k\}$ be a set of multiple sinks and other components have their usual meaning as defined above.

Problem 2. Let $\mathcal{N} = (E, \mathcal{P})$ be a given abstract static $s - D$ network. A lexicographic abstract maximum static flow problem with inter-

mediate storage is to find the maximum flow leaving the source element that is to be sent to sinks via $s - D$ paths $P \in \mathcal{P}$ by allowing the maximum storage of excess flow at intermediate elements e via paths $P_{[s \rightarrow e]} \forall e \in E_I$ with storage capacity $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$.

To solve the problem, we fix the priority of sinks with a compatible sequence in the first stage. In the second stage, we do the same for intermediate elements and finally, in the third stage, we merge the first two stages with a priority of sinks followed by priority of intermediate elements.

Stage 1: Let $D = \{t_1, t_2, \dots, t_k\}$ be a given set of multiple sinks with $d_{P_{[s \rightarrow t_1]}} > d_{P_{[s \rightarrow t_2]}} > \dots > d_{P_{[s \rightarrow t_k]}}$. Then the priority ordering of sinks is $t_1 > t_2 > \dots > t_k$ which respect their rankings. In the case of equal distance, priority is given arbitrarily.

Stage 2: Let $\{e_1, e_2, \dots, e_r\}$ be the set of intermediate elements in E_I with storage capacity more than the sum of movement capacity of incoming paths. Let $d_{P_{[s \rightarrow e_1]}} > d_{P_{[s \rightarrow e_2]}} > \dots > d_{P_{[s \rightarrow e_r]}}$. Then the priority ordering of intermediate elements are $e_1 > e_2 > \dots > e_r$ which respect their rankings. In case of equal distance, priority is given arbitrarily.

Stage 3: In this stage, we first merge Stage 1 and Stage 2 with element priority order $t_1 > \dots > t_k > e_1 > \dots > e_r$. We create dummy ports $\{e'_{k+1}, \dots, e'_{k+r}\}$ corresponding to $\{e_1, \dots, e_r\}$ with the same priority ordering as in E_I of Stage 2. Let $D' = \{t_1, \dots, t_k, e'_{k+1}, \dots, e'_{k+r}\}$ be super sink with a compatible sequence of elements with priority order $t_1 > \dots > t_k > e'_{k+1} > \dots > e'_{k+r}$.

As in previous section, we define the collection of paths as

$$\mathcal{P}'_0 = \emptyset$$

$$\mathcal{P}'_i = \mathcal{P}'_{i-1} \cup \{P_{[s \rightarrow t_i]}\} \text{ for } i = 1, \dots, k$$

$$\mathcal{P}'_i = \mathcal{P}'_{i-1} \cup \{P_{[s \rightarrow e'_i]}\} \text{ for } i = k+1, \dots, k+r.$$

Here, each abstract path-system $\mathcal{N}'_i = (E', \mathcal{P}'_i)$ for $i = 0, 1, \dots, k+r$ forms the abstract network satisfying the switching property where $E' = E \cup \{e'_{k+1}, \dots, e'_{k+r}\}$. We now present [Algorithm 2](#) to solve [Problem 2](#).

Algorithm 2: Lexicographic abstract maximum static flow algorithm.

Input : Given abstract static network $\mathcal{N} = (E, \mathcal{P})$.

Output: Lexicographic abstract maximum static flow with intermediate storage.

1. Fix the priority order $t_1 > \dots > t_k > e'_{k+1} > \dots > e'_{k+r}$ as described in three stages.
 2. Construct the modified network $\mathcal{N}' = (E', \mathcal{P}')$ with single source s and compatible sequence of super sinks $D' = \{t_1, \dots, t_k, e'_{k+1}, \dots, e'_{k+r}\}$.
 3. Compute the lexicographic abstract maximum static flow with priority ordering in Step~1 according to Kappmeier (2015).
 4. Transform the solution to the original network \mathcal{N} by removing dummy ports and dummy paths.
-

The existence of lexicographic abstract maximum flow in \mathcal{N}'_i is as shown in [Theorem 3.2](#). We fix the priority of sink elements followed by the priority of intermediate elements and create super sink set D' by including dummy ports to the given sink D . As in [Theorem 3.3](#), computation of lexicographic abstract maximum static flow with new path-system \mathcal{N}'_i and intermediate storage can be obtained by [Algorithm 2](#). Thus the polynomial time complexity of [Algorithm 2](#) can be proved as in [Corollary 3.4](#).

Theorem 3.5. *Algorithm 2 computes the lexicographic abstract maximum static flow with intermediate storage in $\mathcal{N} = (E, \mathcal{P})$ in polynomial time complexity.*

4. Abstract maximum dynamic flow with intermediate storage

In this section, we investigate the abstract maximum dynamic flow by allowing the storage of excess flow at intermediate elements. We introduce the abstract maximum dynamic flow problem and give its mathematical model with intermediate storage. As a solution procedure, we define the temporal paths and use lexicographic property on a compatible sequence of dummy ports to solve the problem in polynomial time.

Now, we introduce the abstract maximum dynamic flow problem with intermediate storage and mathematical model as follows.

Problem 3. Let $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ be a given abstract dynamic network. An abstract maximum dynamic flow problem with intermediate storage is to find the maximum flow leaving the source element that is to be sent to the sink via $s - t$ paths $P \in \mathcal{P}$ by allowing the maximum storage of excess flow at intermediate elements e via paths $P_{[s \rightarrow e]} \forall e \in E_I$ with storage capacity $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$ within given time horizon T .

Mathematical Model. Let us consider $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ as the abstract dynamic network with temporal dimensions. Let $\tau : E \rightarrow \mathbb{Z}^+$ be a non-negative transit time of element $e \in E$ that is necessary to transship flow from e to its right element and $T \in \mathbb{T}$ be a time horizon. If e and a are two consecutive elements on path P with $e <_P a$, then flow traveling through e at time θ reaches a at time $\theta + \tau_e$. In discrete time setting, time horizon is discretized as $\mathbb{T} = \{0, 1, \dots, T\}$. For each $s - t$ path $P \in \mathcal{P}$, $\tau_P = \sum_{a \in P_{[s \rightarrow t]}} \tau_a$ denotes the traversal time of flow from s to t and $\tau_{P_{[s \rightarrow e]}} = \sum_{a \in P_{[s \rightarrow e]}} \tau_a$ denotes the traversal time of flow from s to intermediate element e through the path $P_{[s \rightarrow e]}$.

Let $\psi^P(\theta) : P \times T \rightarrow \mathbb{R}^+$ be the dynamic $s - t$ path-flow reaching t at discrete time $\theta \in \mathbb{T}$ and $\hat{\psi}_e(\theta) : E_I \times T \rightarrow \mathbb{R}^+$ be the amount of excess flow stored at intermediate element $e \in E_I$ within time $\theta \in \mathbb{T}$. Let $\psi_e^{P,out} = \sum_{P \in \mathcal{A}_e} \psi^P$ and $\psi_e^{P,in} = \sum_{P \in \mathcal{B}_e} \psi^P$ denote the total outflow from e and inflow into e , respectively, whose difference gives the excess flow. The linear programming formulation of abstract dynamic flow with intermediate storage is

$$\max \sum_{P \in \mathcal{P}} \sum_{\theta = \tau_P}^T \psi^P(\theta) + \sum_{e \in E_I} \sum_{\theta = \tau_{P_{[s \rightarrow e]}}}^T \hat{\psi}_e(\theta) \quad (6)$$

$$\text{s.t. } \sum_{P \in \mathcal{P}: e \in P} \psi^P(\theta) \leq u_e \quad \forall e \in E, \theta \in \mathbb{T} \quad (7)$$

$$\psi_e^{P,in}(\theta) - \psi_e^{P,out}(\theta) = \hat{\psi}_e(\theta) \quad \forall e \in E_I, \theta \in \mathbb{T} \quad (8)$$

$$0 \leq \hat{\psi}_e(\theta) \leq v_e \quad \forall e \in E_I, \theta \in \mathbb{T} \quad (9)$$

$$\psi^P \geq 0 \quad \forall P \in \mathcal{P} \quad (10)$$

The objective function in [Eq. \(6\)](#) is to maximize the total flow reaching at t and the excess flow stored at intermediate elements within time horizon T . [Eq. \(7\)](#) represents the capacity constraint of each element at $\theta \in \mathbb{T}$ and the excess flow is represented by [Eq. \(8\)](#). Similarly, non-conservation of the flow at each time step θ is presented by the left inequality of [Eq. \(9\)](#) where the right inequality shows that the excess flow is bounded by the storage capacity of e at each time $\theta \in \mathbb{T}$. [Eq. \(10\)](#) represents the non-negativity of the flow on each path. For the existence of unique solution, the upper and lower bounds of the storage capacity for intermediate elements are taken as $(1 + \delta_e) \sum_{P \in \mathcal{P}: a <_P e} u_a \leq v_e \leq T \sum_{P \in \mathcal{P}: a <_P e} u_a \quad \forall e \in E_I, 0 \leq \delta_e < T$ where, the lower bound is the necessary storage capacity and the upper bound is the sufficient storage capacity. Here, δ_e is a non-negative integer taken as the waiting time or delay of the flow at an element e .

At first, we describe the formation of temporal paths in time expanded form and then present the solution procedure of Problem 3. As in a classical network, time expanded elements are obtained by creating $T + 1$ copies of the elements for each time step $\theta \in \mathbb{T}$. The set of time expanded elements is

$$E_T = \{e^\theta : e \in E, \theta \in \mathbb{T}\},$$

where e^0 represents the original set of elements in the given network. Flow starting from source element s at the time θ reaches to e along with a path P at time $\theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a$. For each path $P \in \mathcal{P}$, the temporal path P^θ for each $\theta \in \{0, 1, \dots, T\}$ is the copy of elements of P in which flow starts on it at the time θ and travels through path P . That is,

$$P^\theta_{[s \rightarrow e]} = \left\{ e^\beta \in E_T : e \in P, \beta = \theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a \right\}.$$

If we replace arbitrary element e with sink element t , then it represents a source-sink temporal path and it is simply denoted by P^θ . The order of elements in temporal path P^θ is the same as in P . The set of all temporal paths that reach to the intermediate element e and sink element t within time horizon T are

$$\mathcal{P}^\theta_{[s \rightarrow e], T} = \left\{ P^\theta_{[s \rightarrow e]} : P_{[s \rightarrow e]} \subset P \in \mathcal{P}, \theta \in \mathbb{T}, \theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a \leq T \right\}$$

and

$$\mathcal{P}_T^\theta = \left\{ P^\theta : P \in \mathcal{P}, \theta \in \mathbb{T}, \theta + \sum_{e \in P} \tau_e \leq T \right\},$$

respectively.

The abstract path system $(E_T, \mathcal{P}_T^\theta)$ may not be an abstract network because it may not satisfy the switching property, Kappmeier (2015). To handle this problem, paths with delay in elements are essential. We define the delay function $\delta : P \rightarrow \{0, 1, \dots, T\}$ for each element in path P . Every flow traveling from s along with path P with delay pattern δ reaches to $e \in P$ at time $\sum_{a \in P_{[s \rightarrow e]}} (\tau_a + \delta_a) + \delta_e$. The temporal path with delay pattern is

$$P^\delta_{[s \rightarrow e]} = \left\{ e^\beta \in E_T : e \in P, \beta = \sum_{a \in P_{[s \rightarrow e]}} (\tau_a + \delta_a) + \delta_e \right\}.$$

The order of elements in P^δ is the same as in P . The set of temporal paths with delay pattern δ arriving at the intermediate element e and destination sink t within time T are

$$\mathcal{P}^\delta_{[s \rightarrow e], T} = \left\{ P^\delta_{[s \rightarrow e]} : P_{[s \rightarrow e]} \subset P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{a \in P_{[s \rightarrow e]}} (\tau_a + \delta_a) \leq T \right\}$$

and

$$\mathcal{P}_T^\delta = \left\{ P^\delta : P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{e \in P} (\tau_e + \delta_e) \leq T \right\},$$

respectively.

Example 3. To clarify the delay pattern, we consider an example with given transit times between the elements as presented in Fig. 7. Here, $\mathcal{P} = \{P_I, P_{II}, P_{III}, P_{IV}, P_V\}$ is the set of paths from s to t . Let $T = 4$ be given time horizon. Then, the temporal $s - t$ paths P^θ with starting time θ from s are as follows:

$$P_I^0 = s^0 - x^1 - t^2, P_I^1 = s^1 - x^2 - t^3, P_I^2 = s^2 - x^3 - t^4$$

$$P_{II}^0 = s^0 - y^2 - t^3, P_{II}^1 = s^1 - y^3 - t^4$$

$$P_{III}^0 = s^0 - x^1 - y^2 - t^3, P_{III}^1 = s^1 - x^2 - y^3 - t^4$$

$$P_{IV}^0 = s^0 - y^2 - x^3 - t^4$$

$$P_V^0 = s^0 - t^4$$

Here, the temporal paths P_{III}^0 and P_{IV}^0 are crossing at y^2 at $\theta = 2$. Due to switching property, paths are switched along $s^0 - y^2 - t^3$ and $s^0 - x^1 - y^2 - x^3 - t^4$. The second switched path does not belong to the path set and also forms a cycle $x^1 - y^2 - x^3$. The cycle can be removed by taking path as $s^0 - x^1 - t^4$ but still it is not an abstract path as transit time from x to t is not 3 units but 1 unit. Thus the waiting time of $\delta_x = 2$ is essential to form an abstract path.

Lemma 4.1 (Kappmeier (2015)). *If an abstract network $\mathcal{N} = (E, \mathcal{P})$ preserves the order on paths, then the network $\mathcal{N}_T = (E_T, \mathcal{P}_T^\delta)$ with path system \mathcal{P}_T^δ is an abstract network.*

For the solution procedure of Problem 3, we first fix the priority of sink and intermediate elements as in Section 3.1, i.e., $t > e_1 > e_2 > \dots > e_r$. By creating dummy ports e' for all $e \in E_I$ and denoting t as e'_0 , the compatible set of dummy ports $D = \{t = e'_0, e'_1, \dots, e'_r\}$ with same priority ordering $e'_0 > e'_1 > \dots > e'_r$ is obtained. Now, the problem is transformed to a single source multi-sink abstract maximum static flow problem. We solve the problem by using a lexicographic maximum flow of Kappmeier (2015) for each time step $\theta \in \mathbb{T}$ in the reconfigured network $\mathcal{N}_{i,T}^\delta = (E_i, \mathcal{P}_{i,T}^\delta)$ with $E_i = E \cup D_i$, $D_i = \{e'_0, e'_1, \dots, e'_r\}$ and $\mathcal{P}_{i,T}^\delta = \mathcal{P}_{i,T}^\delta \cup \mathcal{P}_{[s \rightarrow e'_i], T}^\delta$ for all $i = 0, 1, \dots, r$. Finally, dummy ports are removed to get the solution with intermediate storage. Here, we present a polynomial time algorithm to solve Problem 3.

Algorithm 3: Abstract maximum dynamic flow algorithm.

Input : Given abstract dynamic network $\mathcal{N} = (E, \mathcal{P}, \tau, T)$.

Output: Abstract maximum dynamic flow with intermediate storage on \mathcal{N} .

1. For each $e \in E_I$ with $v_e \geq \sum_{P \in \mathcal{P}: a < p_e} u_a$, compute the shortest distance $d_{P_{[s \rightarrow e]}}$ by taking transit time as cost and using Dijkstra's algorithm.
 2. Fix the priority order as $t = e_0 > e_1 > \dots > e_r$ with first priority to the sink $t = e_0$ and priority for intermediate elements as $d_{P_{[s \rightarrow e_i]}} > d_{P_{[s \rightarrow e_{i+1}]}} \implies e_i > e_{i+1}$, for $i = 1, \dots, r - 1$.
 3. Construct the modified network $\mathcal{N}' = (E', \mathcal{P}')$ with single source s and compatible sequence of multiple sinks with dummy ports $D = \{e'_0, e'_1, \dots, e'_r\}$, where $E' = E \cup D$ and $\mathcal{P}' = \mathcal{P} \cup \{P_{[s \rightarrow e'_i]}\}$.
 4. Set $D_i = \{e'_0, e'_1, \dots, e'_r\}$ for all $i = 0, 1, \dots, r$ so that $D_0 \subseteq \dots \subseteq D_r$.
 5. Construct the reconfigured network $\mathcal{N}_{i,T}^\delta = (E_i, \mathcal{P}_{i,T}^\delta)$ with $E_i = E \cup D_i$ and $\mathcal{P}_{i,T}^\delta = \mathcal{P}_{i,T}^\delta \cup \mathcal{P}_{[s \rightarrow e'_i], T}^\delta$ for all $i = 0, 1, \dots, r$.
 6. For $\theta = 0, 1, \dots, T$:
 For $i = 0, 1, \dots, r$:
 Compute the lexicographic abstract maximum static flow with priority ordering of Step-2 in $\mathcal{N}_{i,T}^\delta = (E_i, \mathcal{P}_{i,T}^\delta)$ using Kappmeier (2015).
 7. Transform the solution to the original network \mathcal{N} by removing dummy ports and dummy paths.
-

Theorem 4.2. *An abstract maximum dynamic flow with intermediate storage obtained from Algorithm 3 is optimal.*

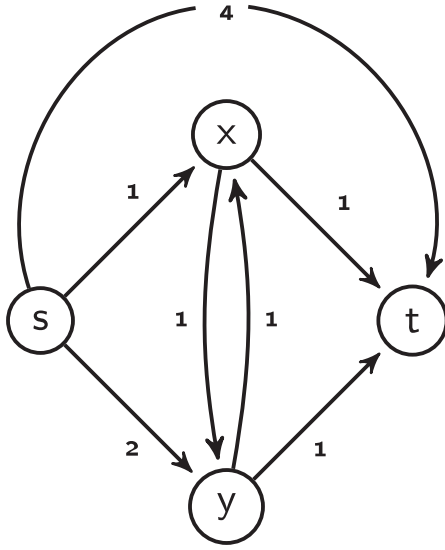


Fig. 7. Dynamic network with transit times between the elements.

Paths:

$$\begin{aligned}
 P_I &= s - x - t \\
 P_{II} &= s - y - t \\
 P_{III} &= s - x - y - t \\
 P_{IV} &= s - y - x - t \\
 P_V &= s - t
 \end{aligned}$$

Proof. In the first three steps of the algorithm, we fix the priority of intermediate elements and keep them in compatible sequence as described in the previous section. To use the lexicographic algorithm of Kappmeier, we obtain set $D_i = \{e'_0, e'_1, \dots, e'_i\}$ for all $i = 0, 1, \dots, r$ so that $D_0 \subseteq \dots \subseteq D_r$. For this set inclusion in multiple sinks, we construct T time expanded temporal paths with delay pattern δ and obtain the reconfigured network $\mathcal{N}_{i,T}^\delta = (E_i, \mathcal{P}_{i,T}^\delta)$ where $E_i = E \cup D_i$ and $\mathcal{P}_{i,T}^\delta = \mathcal{P}_T^\delta \cup \mathcal{P}_{[s \rightarrow e'_i], T}^\delta$ for $i = 0, 1, \dots, r$ such that $\mathcal{N}_{i,T}^\delta \subseteq \mathcal{N}_{i+1,T}^\delta$. Flow at sink $t = e'_0$ is obtained through temporal paths in temporally repeated fashion within time horizon T . At the mean time, excess flows are stored at intermediate elements with respect to their priority order. After this, priority is given to element e'_1 and flow is sent through path $\mathcal{P}_{[s \rightarrow e'_1], T}^\delta$ satisfying the storage capacity constraints within time T together with storing excess flow on the rest of the prioritized intermediate elements along the path. This process continues as long as all prioritized elements have storage capacity and sufficient time to reach the flow. Each iteration of Step 6 executes the flow sent from s at time $\theta = 0, \dots, T$ and reaches lexicographically in each e'_i within time horizon T . So Step 6 provides a feasible solution. At last, flow stored at dummy ports e'_i are replaced at corresponding intermediate elements e_i to get feasible abstract dynamic flow with intermediate storage.

Feasible flow obtained by Algorithm 3 is optimal because lexicographic abstract maximum flow obtained in Step 6 is optimal. Due to dummy ports, every element $e \in E_i$ satisfies flow conservation at every time step θ . As prioritized abstract maximum flow at each dummy port is shifted to the corresponding intermediate element without changing flow at sink t , it gives the optimal abstract maximum dynamic flow with intermediate storage. \square

Observation 4.3. The abstract maximum dynamic flow at the sink is obtained by using temporally repeated flow along paths in \mathcal{P}_T^δ . To calculate the flow at intermediate elements, temporally repeated paths are used but not temporally repeated flow because flow value on paths $\mathcal{P}_{[s \rightarrow e], T}^\delta$ may change over time due to insufficient storage capacity.

Lemma 4.4. Algorithm 3 computes an abstract maximum dynamic flow with intermediate storage in polynomial time complexity.

Proof. Step 1 of Algorithm 3 can be computed in $O(|E|^2)$ time. Steps 2-5 and Step 7 are used to fix the priority of elements,

arranging in a compatible sequence and reformation of a network, which can be done in linear time. As in Kappmeier, Matuschke, and Peis (2014), maximum abstract flow over time at sink t can be computed in time $\mu(|E|, \log(u_{\max}), \log(T)) \cdot \mathcal{O}(P)$, where μ is a polynomial, $u_{\max} = \max_{e \in E} u_e$, and $\mathcal{O}(P)$ denotes the time needed by a call of the oracle \mathcal{O} for the abstract network path $P \in \mathcal{P}_{i,T}^\delta$. It is to be noted that, if $Q \subseteq E$ be a set of elements, then for all $P \in \mathcal{P}_{i,T}^\delta$ the oracle \mathcal{O} returns a violating path P together with the order $<_P$ of its elements such that $P \subseteq Q$ or verifies that there is no path contained in Q . Again, flows at r dummy ports can be obtained in time $r[\mu(|E|, \log(u_{\max}), \log(T)) \cdot \mathcal{O}(P)]$ and so Algorithm 3 solves an abstract maximum dynamic flow problem with intermediate storage within the time complexity of $r[\mu(|E|, \log(u_{\max}), \log(T)) \cdot \mathcal{O}(P)]$. \square

Example 4. Consider an abstract network presented in Fig. 2 of Example 1 by taking cost c as the transit time τ and time horizon $T = 10$. We aim to find an abstract maximum dynamic flow with intermediate storage. As in Algorithm 3, we push the flow up to the sink as the first priority for each time step θ and successively in intermediate elements along the path with respect to priority order which is illustrated in detail in Table 2.

The total amount of flow reached at sink t in time $T = 10$ is 35 units whereas intermediate elements f, b and e store 9, 31 and 31 units, respectively. Flow leaving through path $P_{[s \rightarrow c]}$ at time $\theta = 9$ can store only 3 units at c because of insufficient storage capacity. Similarly, only 4 units of flow through $P_{[s \rightarrow a]}$ can store a at $\theta = 10$ due to insufficient storage capacity. Total amount of flow sent from s within time $T = 10$ is 161 units which are stored at different elements as follows: $\hat{\psi}_t = 35, \hat{\psi}_f = 9, \hat{\psi}_b = 31, \hat{\psi}_e = 31, \hat{\psi}_c = 35$ and $\hat{\psi}_a = 20$. The detailed information (regarding paths and flow values) is given in Table 2 below.

Abstract $s - t$ paths after switching:

Abstract intermediate paths after switching:

$$P_5 = (s, a, e, b, t), \tau_{P_5} = 6, \psi_{P_5} = 1 \quad P_{[s \rightarrow b]}, P_{[s \rightarrow e]}, P_{[s \rightarrow a]}$$

$$P_3 = (s, a, b, t), \tau_{P_3} = 6, \psi_{P_3} = 4 \quad P_{[s \rightarrow b]}, P_{[s \rightarrow a]}$$

$$P_6 = (s, c, e, f, t), \tau_{P_6} = 9, \psi_{P_6} = 1 \quad P_{[s \rightarrow f]}, P_{[s \rightarrow e]}, P_{[s \rightarrow c]}$$

$$P_4 = (s, c, f, t), \tau_{P_4} = 7, \psi_{P_4} = 2 \quad P_{[s \rightarrow f]}, P_{[s \rightarrow c]}$$

Time horizon $T = 10$

Table 2
Abstract flow with intermediate storage in each time θ .

Path	Start time at s	a	c	e	b	f	t	Reaching time at last element
P_5	$\theta = 0$	0	×	3	0	×	1	$\theta = 6$
P_5	$\theta = 1$	0	×	3	0	×	1	$\theta = 7$
P_5	$\theta = 2$	0	×	3	0	×	1	$\theta = 8$
P_5	$\theta = 3$	0	×	3	0	×	1	$\theta = 9$
P_5	$\theta = 4$	0	×	3	0	×	1	$\theta = 10$
$P_{[s \rightarrow b]}$	$\theta = 5$	0	×	3	1	×	×	$\theta = 8$
$P_{[s \rightarrow b]}$	$\theta = 6$	0	×	3	1	×	×	$\theta = 9$
$P_{[s \rightarrow b]}$	$\theta = 7$	0	×	3	1	×	×	$\theta = 10$
$P_{[s \rightarrow e]}$	$\theta = 8$	0	×	4	×	×	×	$\theta = 10$
$P_{[s \rightarrow a]}$	$\theta = 9$	4	×	×	×	×	×	$\theta = 9$
$P_{[s \rightarrow a]}$	$\theta = 10$	4	×	×	×	×	×	$\theta = 10$
P_3	$\theta = 0$	0	×	×	2	×	4	$\theta = 6$
P_3	$\theta = 1$	0	×	×	2	×	4	$\theta = 7$
P_3	$\theta = 2$	0	×	×	2	×	4	$\theta = 8$
P_3	$\theta = 3$	0	×	×	2	×	4	$\theta = 9$
P_3	$\theta = 4$	0	×	×	2	×	4	$\theta = 10$
$P_{[s \rightarrow b]}$	$\theta = 5$	0	×	×	6	×	×	$\theta = 8$
$P_{[s \rightarrow b]}$	$\theta = 6$	0	×	×	6	×	×	$\theta = 9$
$P_{[s \rightarrow b]}$	$\theta = 7$	0	×	×	6	×	×	$\theta = 10$
$P_{[s \rightarrow a]}$	$\theta = 8$	6	×	×	×	×	×	$\theta = 8$
$P_{[s \rightarrow a]}$	$\theta = 9$	6	×	×	×	×	×	$\theta = 9$
$P_{[s \rightarrow a]}$	$\theta = 10$	0	×	×	×	×	×	storage full
P_4	$\theta = 0$	×	3	×	×	0	2	$\theta = 7$
P_4	$\theta = 1$	×	3	×	×	0	2	$\theta = 8$
P_4	$\theta = 2$	×	3	×	×	0	2	$\theta = 9$
P_4	$\theta = 3$	×	3	×	×	0	2	$\theta = 10$
$P_{[s \rightarrow f]}$	$\theta = 4$	×	3	×	×	2	×	$\theta = 8$
$P_{[s \rightarrow f]}$	$\theta = 5$	×	3	×	×	2	×	$\theta = 9$
$P_{[s \rightarrow f]}$	$\theta = 6$	×	3	×	×	2	×	$\theta = 10$
$P_{[s \rightarrow c]}$	$\theta = 7$	×	5	×	×	×	×	$\theta = 8$
$P_{[s \rightarrow c]}$	$\theta = 8$	×	5	×	×	×	×	$\theta = 9$
$P_{[s \rightarrow c]}$	$\theta = 9$	×	3	×	×	×	×	$\theta = 10$
P_6	$\theta = 0$	×	0	0	×	0	1	$\theta = 9$
P_6	$\theta = 1$	×	0	0	×	0	1	$\theta = 10$
$P_{[s \rightarrow f]}$	$\theta = 2$	×	0	0	×	1	×	$\theta = 8$
$P_{[s \rightarrow f]}$	$\theta = 3$	×	0	0	×	1	×	$\theta = 9$
$P_{[s \rightarrow f]}$	$\theta = 4$	×	0	0	×	1	×	$\theta = 10$
$P_{[s \rightarrow e]}$	$\theta = 5$	×	0	1	×	×	×	$\theta = 8$
$P_{[s \rightarrow e]}$	$\theta = 6$	×	0	1	×	×	×	$\theta = 9$
$P_{[s \rightarrow e]}$	$\theta = 7$	×	0	1	×	×	×	$\theta = 10$
$P_{[s \rightarrow c]}$	$\theta = 8$	×	1	×	×	×	×	$\theta = 9$
$P_{[s \rightarrow c]}$	$\theta = 9$	×	0	×	×	×	×	storage full
Total flow stored		20	35	31	31	9	35	Total=161

× = element not used

Here, the waiting pattern of each element is 0 because no two successive common elements appeared in crossing of paths.

The storage pattern of each element can be represented by the time expanded network as shown in Fig. 8.

4.1. Temporally repeated abstract flow with upper bound (sufficient) storage capacity

If the storage capacity of each prioritized intermediate element equals its upper bound (i.e. $v_e = T \sum_{P \in \mathcal{P}: a <_P e} u_a \forall e \in E_I$), then the flow value can be obtained by using temporally repeated flow on sink and intermediate elements through paths with waiting pattern δ as follows:

For $e_0 = t$ and $P \in \mathcal{P}_T^\delta$ with $\psi^P = \min\{u_a : a \in P\}$,

$$|\psi|_{e_0, T} = \sum_P (T - \tau_P + 1) \cdot \psi^P$$

For intermediate element $e_i \in E_I$ and $P_{[s \rightarrow e_i]} \in \mathcal{P}_{[s \rightarrow e_i], T}^\delta \subseteq \mathcal{P}_T^\delta$ with $e_i <_P e_j$,

$$|\psi|_{e_i, T} = \sum_{P_{[s \rightarrow e_i]}} \left[(T - \tau_{P_{[s \rightarrow e_j]}} + 1) \cdot \psi_{e_i} + (\tau_{P_{[s \rightarrow e_j]}} - \tau_{P_{[s \rightarrow e_i]}}) \cdot \psi^{P_{[s \rightarrow e_i]}} \right]$$

where, $|\psi|_{e_i, T}$ is the net flow at element e_i within time T , $\psi^{P_{[s \rightarrow e_i]}} = \min\{u_a : a \in P_{[s \rightarrow e_i]}\}$ and $\tau_{P_{[s \rightarrow e_i]}} = \sum_{a \in P_{[s \rightarrow e_i]}} (\tau_a + \delta_a)$.

To use the temporally repeated flow in Example 4, we take sufficient storage capacity on each intermediate element. The path decomposition of flow in prioritized order of elements is presented in Fig. 9. Numbers indicated between the elements are flow and transit time.

The total amount of flow that can be pushed from the source within time $T = 10$ is 170 units. By using temporally repeated flow, amount of flow stored at sink and intermediate elements are as follows: $\hat{\psi}_t = 35$, $\hat{\psi}_f = 9$, $\hat{\psi}_b = 31$, $\hat{\psi}_e = 31$, $\hat{\psi}_c = 38$ and $\hat{\psi}_a = 26$.

Theorem 4.5. The maximum dynamic abstract flow with intermediate storage obtained by using temporally repeated flow is optimal for the sink as well as each intermediate element.

Proof. First, we prove the feasibility of the flow. Since flow on each decomposed path is the bottleneck movement capacity of elements along the path, the flow on each path P and $P_{[s \rightarrow e_i]}$ satisfies the capacity constraints. The flow conservation is due to dummy ports, so each flow on the path is feasible.

Due to priority ordering, the first prioritized flow at sink t is optimally obtained by using a temporally repeated flow of

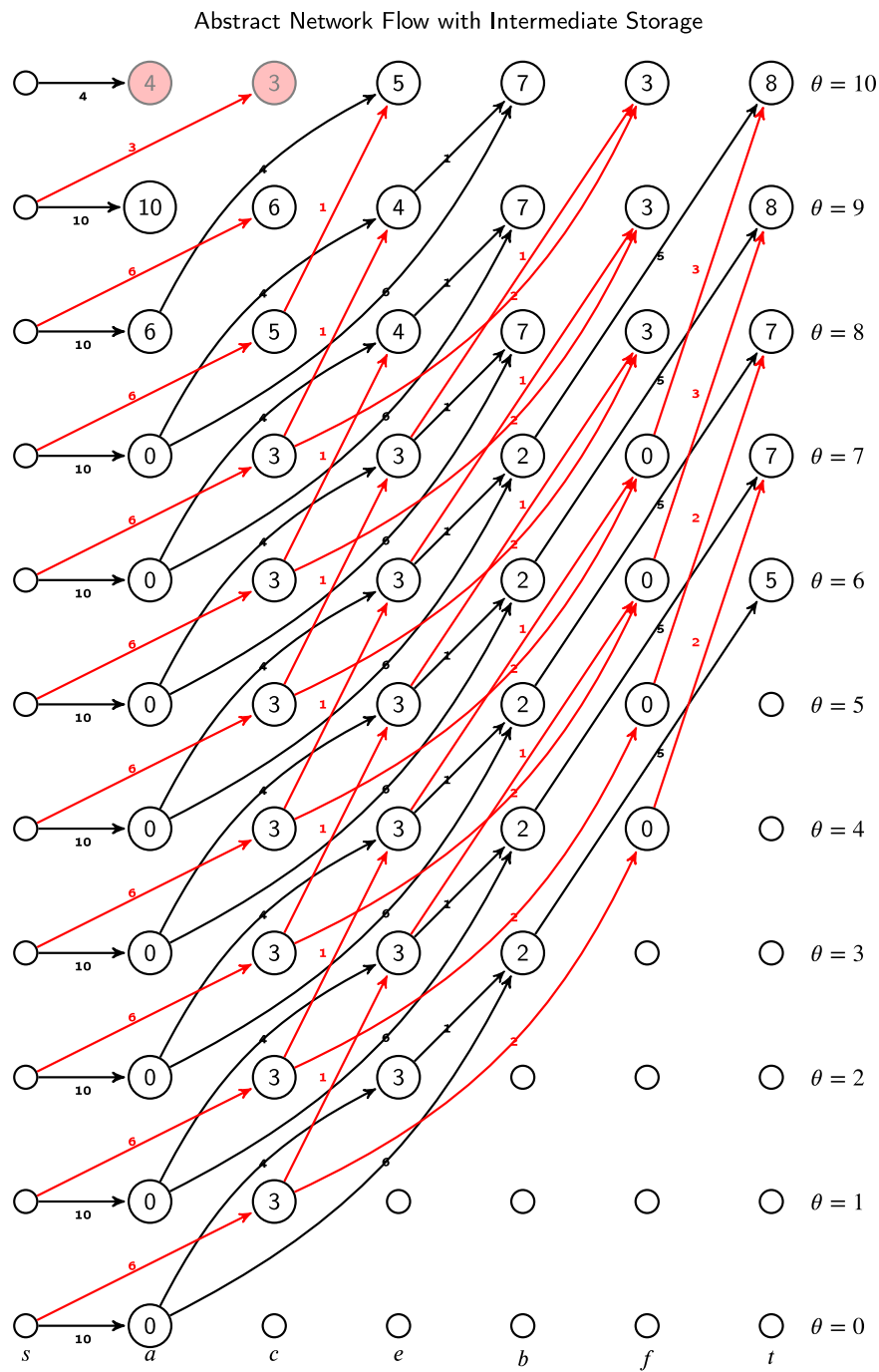


Fig. 8. Time expanded network with intermediate storage where numbers inside the circle represent the storage of flow at each time step. The pink circle shows the element with full of capacity.

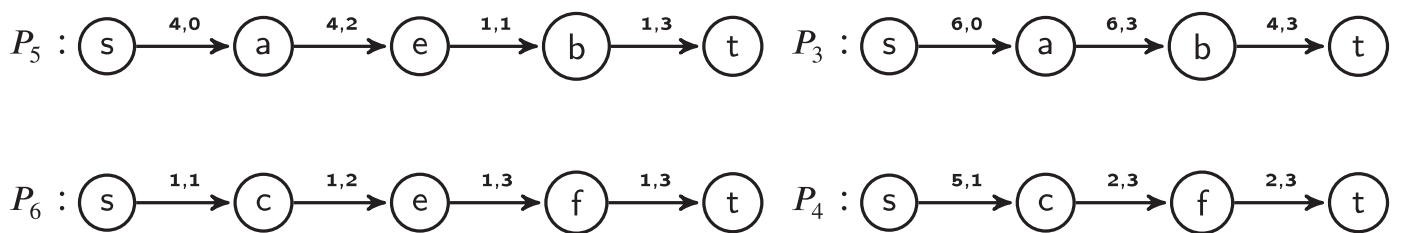


Fig. 9. Path decomposition of abstract flow after switching .

Ford and Fulkerson (1956). In the meantime, excess flows are stored at prioritized intermediate elements. To obtain the flow at each prioritized intermediate element e_i , the bottleneck movement capacity of elements along each decomposed paths reaching the element are temporally repeated within time deviation $(\tau_{P_{[s \rightarrow e_j]}} - \tau_{P_{[s \rightarrow e_i]}})$ for $e_i <_P e_j$. No more flow can reach this intermediate element within time horizon T . Thus, the sum of temporally repeated excess flow within time $(T - \tau_{P_{[s \rightarrow e_j]}} + 1)$ and the temporally repeated path flow reaching e_i within time $(\tau_{P_{[s \rightarrow e_j]}} - \tau_{P_{[s \rightarrow e_i]}})$ is optimal for $e_i <_P e_j$. Again, the total amount of flow at e_i does not exceed $T \sum_{P \in \mathcal{P}: a <_P e} u_a \forall e_i \in E_I$, so every intermediate element holds optimal flow within the time horizon T . \square

If the storage capacity v_e is less than the sufficient storage capacity (i.e., $v_e < T \sum_{P \in \mathcal{P}: a <_P e} u_a$), then the temporally repeated formula may not hold for intermediate elements because storage capacity may be insufficient to store the flow after some iterations. This is clearly shown in Table 2 and Fig. 8 of Example 4.

5. Abstract contraflow with intermediate storage

In this section, we discuss the evacuation problem with contraflow configuration in an abstract network where intermediate storage of the flow is allowed. In a two-way abstract network, contraflow means reversal of oppositely directed paths towards the destination element to improve the flow and reduce the evacuation time. We discuss two different aspects of transit times, symmetric and asymmetric, between pair of elements along the oppositely directed paths.

5.1. Contraflow with symmetric transit times

Let $\mathcal{N} = (E, \overleftrightarrow{P}, \tau, T)$ be an abstract dynamic network, where E represents the set of elements and $\overleftrightarrow{P} = \overrightarrow{P} \cup \overleftarrow{P} \cup \overrightarrow{P}_{[s \rightarrow e]} \cup \overleftarrow{P}_{[e \rightarrow s]}$ represents the set of two-way paths. Here, \overrightarrow{P} and \overleftarrow{P} represent forward ($P_{[s \rightarrow t]}$) and backward ($\overleftarrow{P}_{[t \rightarrow s]}$) source-sink paths, respectively. Similarly, $\overrightarrow{P}_{[s \rightarrow e]}$ and $\overleftarrow{P}_{[e \rightarrow s]}$ represent forward and backward intermediate paths from s to e and e to s , respectively. Let $\tau : E \rightarrow \mathbb{Z}^+$ be a symmetric transit time between pair of consecutive elements along a path so that $\tau_{\overrightarrow{P}_{[e \rightarrow a]}} = \tau_{\overleftarrow{P}_{[a \rightarrow e]}}$ with $e <_{\overrightarrow{P}} a$ and $a <_{\overleftarrow{P}} e$. Clearly, $\tau_{\overrightarrow{P}} = \tau_{\overleftarrow{P}}$ and $\tau_{\overrightarrow{P}_{[s \rightarrow e]}} = \tau_{\overleftarrow{P}_{[e \rightarrow s]}}$. Temporal component T represents the time horizon. For static network, transit time τ is considered as cost c and time horizon T is absent. Contrary to the general abstract network, it is possible to have incoming movement capacity to the source and outgoing movement capacity from the sink as non-negative integer for the two way abstract network.

Problem 4. For a given abstract static (dynamic) network $\mathcal{N} = (E, \overleftrightarrow{P})$ ($\mathcal{N} = (E, \overleftrightarrow{P}, \tau, T)$), the abstract static (dynamic) maximum contraflow problem with intermediate storage is to find the maximum flow leaving the source element that is to be sent to sink via $s - t$ paths $\overrightarrow{P} \cup \overleftarrow{P}$ and allowing the storage of excess flow at intermediate elements e via paths $\overrightarrow{P}_{[s \rightarrow e]} \cup \overleftarrow{P}_{[e \rightarrow s]} \forall e \in E_I$ with storage capacity $v_e \geq \sum_{P \in \mathcal{P}: a <_P e} u_a$; $P' = \overrightarrow{P} \cup \overleftarrow{P}$ (within a given time horizon T) by reversing the direction of paths \overleftarrow{P} and $\overleftarrow{P}_{[e \rightarrow s]}$ at time zero.

To solve the problem, we first construct an auxiliary network by adding two-way movement capacities between two consecutive elements. We denote auxiliary network $\tilde{\mathcal{N}} = (E, \tilde{P}, \tilde{\tau}, T)$ where \tilde{P} is obtained by reverting the direction of paths \overleftarrow{P} and $\overleftarrow{P}_{[e \rightarrow s]}$ at time zero. The movement capacity \tilde{u}_e and transit time $\tilde{\tau}_e$ are defined

as follows: For any two consecutive elements e and a with $e <_{\overrightarrow{P}} a$ and $a <_{\overleftarrow{P}} e$

$$\tilde{u}_e = u_{e:e \in \overrightarrow{P}} + u_{a:a \in \overleftarrow{P}}$$

where $u_{a:a \in \overleftarrow{P}} = 0$ if $a \notin \overleftarrow{P}$ and

$$\tilde{\tau}_e = \begin{cases} \tau_{e:e \in \overrightarrow{P}} & \text{if } e <_{\overrightarrow{P}} a \\ \tau_{a:a \in \overleftarrow{P}} & \text{otherwise.} \end{cases}$$

We now present a generic algorithm to solve abstract contraflow problems. We first transform the given two-way network to an auxiliary network $\tilde{\mathcal{N}} = (E, \tilde{P})$ and construct cycle free paths on $\tilde{\mathcal{N}}$. On these cycle free paths, we solve abstract maximum static and lexicographic abstract maximum static flow problems as described in Section 3 and use Algorithm 4 to obtain an optimal solution to the corresponding contraflow problems with intermediate storage. In these static cases, transit times are considered as costs. Similarly, in Step 2 of Algorithm 4, we solve the abstract maximum dynamic flow problem on the auxiliary network via cycle free path as in Section 4, which gives the optimal solution to the abstract maximum dynamic contraflow problem with intermediate storage.

Algorithm 4: Generic algorithm for abstract maximum contraflow problems.

Input : Given abstract two-way network $\mathcal{N} = (E, \overleftrightarrow{P})$.

Output: Abstract maximum contraflow with intermediate storage.

1. Construct an auxiliary network $\tilde{\mathcal{N}} = (E, \tilde{P})$.
 2. Construct cycle free paths on $\tilde{\mathcal{N}}$ satisfying the switching property.
 3. Compute abstract maximum flow with intermediate storage.
 4. A path $\overleftarrow{P} / \overrightarrow{P}_{[e \rightarrow s]}$ is reversed if and only if the flow along with path $\overrightarrow{P} / \overleftarrow{P}_{[s \rightarrow e]}$ is greater than its capacity or if there is a non-negative flow along path with $\overrightarrow{P} / \overleftarrow{P}_{[s \rightarrow e]} \notin \overleftrightarrow{P}$ and the resulting flow is maximum abstract flow with the path reversals for the network \mathcal{N} .
-

Since Step 1 and Step 2 of Algorithm 4 can be computed in $O(E)$ time and Step 3 can be obtained in polynomial time complexity (using Algorithm 1 for static, Algorithm 2 for lexicographic and Algorithm 3 for dynamic problems), the polynomial time solvability of Algorithm 4 is at hand.

Theorem 5.1. Algorithm 4 solves the abstract maximum contraflow problems with symmetric transit times in polynomial time complexity.

So far we discussed about the contraflow with symmetric transit times on the anti-parallel paths. But it is not always necessary to have the same transit times on the anti-parallel paths between pair of elements. Such transit times are known as asymmetric transit times. In such cases, symmetric reversal (i.e. reversal with same capacity and transit times as before the reversal) of paths can be possible by taking an additional artificial element in the reversed path with capacity and transit times as $u, \frac{\tau}{2}$ in both halves of the reversed path. As the parallel reversal is the symmetric reversal, we can apply the algorithm of Rebennack et al. (2010), Pyakurel et al. (2019b), Pyakurel et al. (2019a) to solve the contraflow problem without intermediate storage and Pyakurel and Dempe (2020) in case of intermediate storage. Hereafter, we present the contraflow technique in which reversal of paths are made with asymmetric transit time by taking direction dependent transit times.

5.2. Contraflow with asymmetric transit times

In two-way network topology, if transit times along the anti-parallel paths between two consecutive elements are not identical then it is known as a network with asymmetric transit times. Nath, Pyakurel, and Dhamala (2021) considered the orientation dependent asymmetric transit times of reversed lanes in general form and presented strongly polynomial time algorithms to solve single source-sink maximum dynamic and quickest contraflow problems. Here, we discuss on abstract contraflow problem with intermediate storage by taking direction dependent transit times.

Let $\mathcal{N} = (E, \vec{P}, \tau, T)$ be an abstract dynamic network with asymmetric transit time $\tau : E \rightarrow \mathbb{Z}^+$ between pair of consecutive elements along a path so that $\tau_{\vec{P}_{[e \rightarrow a]}} \neq \tau_{\overleftarrow{P}_{[a \rightarrow e]}}$ with $e <_{\vec{P}} a$ and $a <_{\overleftarrow{P}} e$. We construct an auxiliary network $\tilde{\mathcal{N}} = (E, \vec{P}, \bar{\tau}, T)$ where \vec{P} is obtained by reversing the direction of paths \vec{P} and $\overleftarrow{P}_{[e \rightarrow s]}$ at time zero. For any two consecutive elements e and a with $e <_{\vec{P}} a$ and $a <_{\overleftarrow{P}} e$, we define movement capacity \bar{u}_e and transit time $\bar{\tau}_e$ as

$$\bar{u}_e = u_{e:e \in \vec{P}} + u_{a:a \in \overleftarrow{P}}$$

where $u_{a:a \in \overleftarrow{P}} = 0$ if $a \notin \overleftarrow{P}$ and

$$\bar{\tau}_e = \begin{cases} \tau_{e:e \in \vec{P}} & \text{if } e <_{\vec{P}} a \\ \tau_{a:a \in \overleftarrow{P}} & \text{if } a <_{\overleftarrow{P}} e \text{ or } \nexists \vec{P} : e <_{\vec{P}} a \end{cases} \quad \forall \vec{P} \in \vec{P}.$$

It is to be noted that if there is no path segment between two adjacent elements along the orientation of auxiliary network, then transit time of the auxiliary path is taken as the symmetric reversal of the opposite path segment.

Here, the transit time $\bar{\tau}_e$ of the auxiliary network depends on the direction of the abstract path after contraflow $\vec{P} \in \vec{P}$. By using Algorithm 4, the optimal solution to abstract maximum static, lexicographic abstract maximum static and abstract maximum dynamic contraflow problems with intermediate storage can be obtained in a two-way network with asymmetric transit times, where transit time is considered as cost in static problems. For the asymmetric contraflow, we fix the priority of elements before the contraflow configuration because the alternative path reversals between the elements in the network grows exponentially with the number of path segments on the network which increases its computational complexity. Though priority order of the elements in our solution strategy is on the basis of distance, our algorithms can be used for any other priority order.

Theorem 5.2. For contraflow network with asymmetric transit times, Algorithm 4 solves the abstract maximum contraflow problems in polynomial time complexity.

In these contraflow configurations mentioned above, we add the capacity of each oppositely directed paths between two adjacent elements to form undirected auxiliary network. While sending flow on abstract path, if the movement capacity of an element along the forward path segment is less then the bottleneck capacity of the path in auxiliary network then the reversal of backward path segment is essential.

6. Continuous time abstract dynamic flow with intermediate storage

In Section 4 and Section 5, we discussed abstract dynamic flow and contraflow problems with intermediate storage in discrete time setting, where discrete dynamic flow function ψ is used to assign the flow from each element at each time step $\theta = 0, \dots, T$ satisfying the capacity constraints. In this section, we discuss continuous time abstract flow with intermediate storage. A continuous dynamic abstract flow function $\check{\psi}$ with intermediate storage is

a Lebesgue-measurable function, which is defined as the flow rate per unit time that leaves each element at each moment of time by allowing the storage of excess flow at intermediate elements without violating the capacity constraints.

Fleischer and Tardos (1998) established the strong relation between discrete and continuous flow models by the notion of natural transformation. This natural transformation defines the continuous dynamic flow for a time interval $[\theta, \theta + 1)$ with $\check{\psi}_{ij}[\theta, \theta + 1) = \psi_{ij}(\theta)$, where $\psi_{ij}(\theta)$ is the amount of discrete dynamic flow entering arc (i, j) , $i, j \in E$ at each time step $\theta = 0, \dots, T$. Here, we use the same logic to transform the discrete time abstract flow to continuous time abstract flow with intermediate storage as follows: any discrete abstract flow over time $\psi^{P_{[e \rightarrow a]}}(\theta)$ with integral time horizon T is equivalent to the continuous abstract flow over time $\check{\psi}^{P_{[e \rightarrow a]}}[\theta, \theta + 1)$ by incorporating the flow $\psi^{P_{[e \rightarrow a]}}$ leaving element e to a along path P with $e <_P a$ at time step $\theta \leq T - \tau_{P_{[e \rightarrow a]}}$ as a constant flow rate from e during the unit time interval $[\theta, \theta + 1)$ by allowing the storage of excess flow at intermediate elements. Mathematically,

$$\int_{\theta}^{\theta+1} \check{\psi}^{P_{[e \rightarrow a]}}(\alpha) d\alpha = \psi^{P_{[e \rightarrow a]}}(\theta) \quad \forall e <_P a.$$

Using this natural transformation, problems defined in previous sections (Sections 4 and 5) with continuous time settings can be solved in polynomial time complexity using their respective algorithms.

7. Conclusion

Abstract flow without intermediate storage has been well studied in literature which permits the flow on path rather than on arc. By using complementary slackness condition on augmenting path structure, a polynomial time algorithm for abstract static flow problem has been obtained. The lexicographically maximum flow, maximum flow over time and earliest arrival flow problems have been solved efficiently in abstract networks. All these problems permit the flow conservation at intermediate elements in their solutions.

In this paper, we have investigated the abstract flow models with intermediate storage in static and dynamic networks. We have introduced the maximum static, lexicographic maximum static and maximum dynamic flow problems in the abstract network and presented polynomial time algorithms to solve them. We have derived temporally repeated flow with intermediate storage if the storage capacity of each element is sufficient, i.e., equals its upper bound.

As contraflow is one of the best techniques to increase the outbound capacity and minimize the transmission time, we have presented a generic algorithm to solve maximum static, lexicographic maximum static and maximum dynamic flow problems in the abstract contraflow network with symmetric as well as asymmetric transit times. By using natural transformation in abstract network, we solved the maximum dynamic flow problem and maximum dynamic contraflow problems in continuous time setting.

To the best of our knowledge, abstract network flow models with intermediate storage, polynomial time solution to the maximum static, lexicographic maximum static and maximum dynamic flow problems and formulation of temporally repeated flow with intermediate storage are introduced for the first time. Together with this, abstract contraflow problems for symmetric as well as asymmetric transit times are the novel work of this research.

Declaration of Competing Interest

None.

Acknowledgement

The first author (Urmila Pyakurel) thanks to the Alexander von Humboldt Foundation for Digital Cooperation Fellowship (August 1, 2021 – January 31, 2022) and for Remote Cooperation Aboard Fellowship (March 1 - August 30, 2022); and the second author (Durga Prasad Khanal) thanks to the German Academic Exchange Service - DAAD for Research Grants - Bi-nationally Supervised Doctoral Degrees/Cotutelle, 2021/22. The authors would also like to thank the anonymous referees and the editor for their valuable suggestions to improve the quality of this paper.

References

- Akbari, V., & Salman, F. S. (2017). Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. *European Journal of Operational Research*, 257, 625–640.
- Akter, S., & Wamba, S. F. (2019). Big data and disaster management: A systematic review and agenda for future research. *Annals of Operations Research*, 283, 939–959.
- Altay, N., & Green III, W. G. (2006). OR/MS research in disaster operations management. *European Journal of Operational Research*, 175, 475–493.
- Amideo, A. E., Scaparra, M. P., & Kotiadis, K. (2019). Optimising shelter location and evacuation routing operations: The critical issues. *European Journal of Operational Research*, 279, 279–295.
- Anaya-Arenas, A. M., Renaud, J., & Ruiz, A. (2014). Relief distribution networks: A systematic review. *Annals of Operations Research*, 223, 53–79.
- Aronson, J. E. (1989). A survey of dynamic network flows. *Annals of Operations Research*, 20, 1–66.
- Arulselvan, A. (2009). *Network model for disaster management*. University of Florida Gainesville Ph.D. thesis.
- Burkard, R. E., Daskalakis, K., & Klinz, B. (1993). The quickest flow problem. *Zeitschrift für Operations Research*, 37, 31–58.
- Chen, L., & Miller-Hooks, E. (2008). The building evacuation problem with shared information. *Naval Research Logistics (NRL)*, 55, 363–376.
- Choi, W., Hamacher, H. W., & Tufekci, S. (1988). Modeling of building evacuation problems by network flows with side constraints. *European Journal of Operational Research*, 35, 98–110. [https://doi.org/10.1016/0377-2217\(88\)90382-7](https://doi.org/10.1016/0377-2217(88)90382-7).
- Cova, T. J., & Johnson, J. P. (2003). A network flow model for lane-based evacuation routing. *Transportation Research Part A: Policy and Practice*, 37, 579–604.
- Dhamala, T. N. (2015). A survey on models and algorithms for discrete evacuation planning network problems. *Journal of Industrial and Management Optimization*, 11, 265–289.
- Dhamala, T. N., & Adhikari, I. M. (2018). On evacuation planning optimization problems from transit-based perspective. *International Journal of Operations Research*, 15, 29–47.
- 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, 101–133.
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269–271.
- Fleischer, L., & Tardos, E. (1998). Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23, 71–80.
- Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8, 399–404.
- Ford, L. R., & Fulkerson, D. R. (1962). *Flows in networks*. Princeton University Press.
- Government of Nepal (2015). Nepal disaster risk reduction portal, <http://drportal.gov.np>.
- Hamacher, H., & Tjandra, S. (2002). Mathematical modeling of evacuation problems: A state of the art. In M. Schreckenberger, & S. D. Sharma (Eds.), *Pedestrian and evacuation dynamics* (pp. 227–266).
- Hamacher, H. W., Heller, S., & Rupp, B. (2013). Flow location (FlowLoc) problems: Dynamic network flows and location models for evacuation planning. *Annals of Operations Research*, 207, 161–180.
- Hoffman, A. (1974). A generalization of max flow - min cut. *Mathematical Programming: Series A and B*, 6, 352–359.
- Hua, J., Ren, G., Cheng, Y., & Ran, B. (2014). An integrated contraflow strategy for multimodal evacuation. *Mathematical Problems in Engineering*, 2014.
- Huertas, J. A., & Van Hentenryck, P. (2022). Large-scale zone-based evacuation planning: Generating convergent and non-preemptive evacuation plans via column generation. In *Proceedings of the 55th Hawaii international conference on system sciences*. <https://doi.org/10.24251/HICSS.2022.307>.
- Kappmeier, J. P. (2015). *Generalizations of flows over time with applications in evacuation optimization*. Technical University, Berlin, Germany Ph.D. Thesis.
- 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.
- Khanal, D. P., Pyakurel, U., & Dhamala, T. N. (2021). Maximum multimodal flow with intermediate storage. *Mathematical Problems in Engineering*, 2021.
- Kim, S., Shekhar, S., & Min, M. (2008). Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, 20, 1115–1129.
- Kotsireas, I. S., Nagurney, A., & Pardalos, P. M. (2015). *Dynamics of disasters: Key concepts, models, algorithms, and insights*. Springer Proceedings in Mathematics and Statistics.
- Kurian, V., Chinnusamy, S., Natarajan, A., Narasimhan, S., & Narasimhan, S. (2018). Optimal operation of water distribution networks with intermediate storage facilities. *Computers and Chemical Engineering*, 119, 215–227.
- Martens, M. (2007). *Path-constrained network flows*. Technical University, Berlin, Germany Ph.D. Thesis.
- 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). Springer.
- McCormick, S. T. (1996). A polynomial algorithm for abstract maximum flow. In *SODA* (pp. 490–497). Citeseer.
- Minieka, E. (1973). Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21, 517–527.
- Moriarty, K. D., Ni, D., & Collura, J. (2007). Modeling traffic flow under emergency evacuation situations: Current practice and future directions. In *Transportation research board 2007 annual meeting*.
- Nath, H. N., Pyakurel, U., & Dhamala, T. N. (2021). Network reconfiguration with orientation-dependent transit times. *International Journal of Mathematics and Mathematical Sciences*, Vol. 2021, 11. <https://doi.org/10.1155/2021/6613622>. Article ID 6613622.
- Pardalos, P. M., & Arulselvan, A. (2009). Multimodal solutions for large scale evacuations. *Technical report*. University of Florida, Center for Multimodal Solutions for Congestion Mitigation.
- Pascoal, M. M., Captivo, M. E. V., & Clímaco, J. C. (2006). A comprehensive survey on the quickest path problem. *Annals of Operations Research*, 147, 5–21.
- Purba, D. S. D., Kontou, E., & Vogiatzis, C. (2021). Evacuation network modeling for alternative fuel vehicles. arXiv:2109.01578.
- Pyakurel, U., & Dempe, S. (2020). Network flow with intermediate storage: Models and algorithms. *SN Operations Research Forum*, 1, 1–23.
- Pyakurel, U., & Dempe, S. (2021). Universal maximum flow with intermediate storage for evacuation planning. In I. S. Kotsireas, A. Nagurney, P. M. Pardalos, & A. Tsokas (Eds.), *Dynamics of disasters, springer optimization and its applications: Vol. 169*. Cham: Springer. https://doi.org/10.1007/978-3-030-64973-9_14.
- Pyakurel, U., & Dhamala, T. N. (2016). Continuous time dynamic contraflow models and algorithms. *Advances in Operations Research*, Vol. 2016.
- Pyakurel, U., Dhamala, T. N., & Dempe, S. (2017). Efficient continuous contraflow algorithms for evacuation planning problems. *Annals of Operations Research*, 254, 335–364.
- 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, 1–29.
- Pyakurel, U., Nath, H. N., & Dhamala, T. N. (2019a). Partial contraflow with path reversals for evacuation planning. *Annals of Operations Research*, 283, 591–612.
- Rebennack, S., Arulselvan, A., Eleftheriadou, L., & Pardalos, P. M. (2010). Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, 19, 200–216.
- Schadschneider, A., Klingsch, W., Klüpfel, H., Kretz, T., Rogsch, C., & Seyfried, A. (2008). Evacuation dynamics: Empirical results, modeling and applications. arXiv:0802.1620.
- Vermuyten, H., Beliën, J., De Boeck, L., Reniers, G., & Wauters, T. (2016). A review of optimisation models for pedestrian evacuation and design problems. *Safety Science*, 87, 167–178. <https://doi.org/10.1016/j.ssci.2016.04.001>.
- Weitzel, T., & Glock, C. H. (2018). Energy management for stationary electric energy storage systems: A systematic literature review. *European Journal of Operational Research*, 264, 582–606.
- Yusoff, M., Ariffin, J., & Mohamed, A. (2008). Optimization approaches for macroscopic emergency evacuation planning: A survey. In *2008 International symposium on information technology* (pp. 1–7). IEEE.
- Zambrano, J. A., Huertas, J. A., Segura-Durán, E., & Medaglia, A. L. (2020). Time estimation and hotspot detection in the evacuation of a complex of buildings: A mesoscopic approach and case study. *IEEE Transactions on Engineering Management*, 67, 641–658. <https://doi.org/10.1109/TEM.2019.2960354>.

ABSTRACT TEMPORALLY REPEATED FLOW WITH INTERMEDIATE STORAGE

DURGA PRASAD KHANAL¹, URMILA PYAKUREL², TANKA NATH DHAMALA³,
STEPHAN DEMPE⁴

¹ *Saraswati Multiple Campus, Tribhuvan University, Kathmandu, Nepal;*

^{2,3} *Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal;*

⁴ *Faculty of Mathematics and Computer Science, TU Bergakademie Freiberg, Freiberg, Germany;*

² *Corresponding author: urmilapyakurel@gmail.com*

Abstract: Network associated with the set of elements and linearly ordered subset of elements, known as paths, satisfying the switching property is an abstract network. Due to the switching property, flows crossing at intersections are diverted to the non-crossing sides. Each element of an abstract network is equipped with two types of integral capacities: one is movement capacity which transships the flow from an element to its adjacent element and another is the storage capacity which holds the flow at the element. Due to insufficient movement capacity of intermediate elements, flow out from the source may not reach at the destination. If the flow out from the source is more than the minimum cut capacity, then the problem associated with the settlement of excess flow at appropriate intermediate elements is termed as network flow with intermediate storage. In this paper, we discuss the static and dynamic flow models with intermediate storage in an abstract network using temporal repetition of flow. We solve abstract maximum dynamic flow and contraflow problems with intermediate storage.

Keywords: Abstract network, maximum flow, switching property, temporally repeated flow, contraflow

1. INTRODUCTION

A disaster is a disruption occurring over a short or long period of time that causes massive loss of human, material, economic or environment. A very efficient post disaster evacuation planning is essential to minimize the losses. At the time of evacuation, evacuees are to be shifted from danger zones (sources) to safety places (sinks) as quickly and efficiently as possible. In contrast to evacuation planning problems based on the flow conservation constraints at intermediate elements, evacuation planning problem with intermediate storage deals with the settlement of excess flow at intermediate elements. By holding the excess flow not reaching to the destination at comparatively safer intermediate shelters, it maximizes the number of evacuees leaving the danger zone. To address such problems in classical network, Pyakurel and Dempe [19] introduced the concept of network flow with intermediate storage and presented polynomial time algorithm to solve the maximum dynamic flow problem. Pyakurel and Dempe [21] investigated universal maximum dynamic flow with intermediate storage and presented efficient algorithms in general as well as two-terminal

series parallel networks. To transship more than one commodity, Khanal et al. [11] introduced the maximum multicommodity flow problem with intermediate storage and presented efficient algorithm to solve it. Not only in evacuation planning, intermediate storage is highly applicable for different demand-supply chains like commodity supply, electricity distribution, water supply, etc.

In two way network topology, an important scenario after disaster is that the paths towards the risk zones are almost empty (i.e., no flow on paths towards the danger zones). The optimal use of empty paths to maximize the flow and minimize the time of evacuation is possible by a widely accepted technique, known as contraflow configuration. Kim et al. [15] presented a greedy heuristic to produce high quality solutions and a bottleneck heuristic to deal with large scale evacuations. The strongly polynomial time algorithms for maximum and quickest contraflow problems in two terminal network with discrete time settings can be found in Rebenack et al. [27]. In continuous-time settings, Pyakurel and Dhamala [20] introduced the dynamic contraflow model. By using the natural transformation of Fleischer and Tardos [3], they have presented efficient algorithms to solve the maximum, quickest, and earliest arrival flow problems with lane reversals. Pyakurel et al. [25] introduced the concept of partial lane reversals in which only necessary arc capacities are reversed to increase the flow value and unused arc capacities are saved for other emergency purposes like logistic supports and facility locations. Models and solution strategies of different problems regarding the evacuation plannings with/without contraflow can be found in [1, 6, 10, 12, 14, 23].

A network with capacitated elements and linearly ordered subset of elements, called paths, is an abstract network. When two paths in abstract network cross at an element then there must be a path that is a subset of the first path up to the crossing element and a subset of second path after the crossing element, known as switching property. Hoffman [7] introduced the concept of abstract flow by reviewing the first proof of max-flow-min-cut theorem of Ford and Fulkerson [4] with flows in term of paths rather than on arcs. McCormick [18] provided a polynomial time algorithm by using an oracle where input is an arbitrary subset of elements whose output is either a path contained in that subset or states that no such path exists. He used the augmenting path structure satisfying the complementary slackness condition: every positive path meets the cut set exactly at one common element and every element of the cut is saturated. Martens and McCormick [17] extended the result of [18] in more general case by using additional attribute of weight on paths. Martens [16] presented unsplitable and k -splitable abstract network flows. Similarly, Kappmeier [9] presented polynomial algorithm for lexicographic abstract maximum flow and used it to prove the existence of abstract earliest arrival flow.

The concept of continuous maximum abstract contraflow problem is introduced by Pyakurel et al. [22]. They have presented polynomial time algorithms to solve the problems. Similarly, Pyakurel et al. [26] introduced the partial contraflow approach in abstract network by saving unused capacities of the elements, and presented efficient algorithms for static, lexicographically maximum static, maximum dynamic and earliest arrival partial contraflow problems. The polynomial time algorithms to solve the maximum static, lexicographic maximum static and maximum dynamic flow problems in abstract network and abstract contraflow network with intermediate storage can be found in Pyakurel et al. [24]. They have introduced the formula of temporally repeated flow to solve maximum dynamic flow with intermediate storage if the holding capacity of nodes is sufficient.

By introducing the partial switching property, Khanal et al. [13] solved an abstract quickest flow problem with partial switching of paths.

For a given set of source-sink paths with fixed transit times, if the flow is sent along the decomposed paths repeatedly over the time with constant rate of flow, is called temporally repeated flow. Ford and Fulkerson [5] introduced the temporally repeated flow in which a stationary maximal dynamic flow can be obtained by solving a transshipment problem associated with the static network. This flow is known to be an optimal dynamic flow. Using the concept of temporally repeated flow in classical network, Kappmeier et al. [8] presented the temporally repeated abstract flow to obtain abstract flow over time from source to the sink. Here, we introduce the temporally repeated flow on an abstract network from the source element to the sink as well as each of the intermediate elements. With the help of intermediate storage, it maximizes the flow out from the source in polynomial time complexity.

In this paper, we present mathematical models for static and dynamic flow problems with intermediate storage. We introduce a temporally repeated flow with intermediate storage to obtain the abstract maximum dynamic flow (MDF) if the storage capacity of each intermediate element is sufficient (i.e., T times the sum of incoming capacities from its left elements through the paths) and present a polynomial time algorithm to solve MDF problem. We also present a maximum dynamic contraflow (MDCF) problem with intermediate storage using the temporally repeated flow and present polynomial time solution strategy. We organize the paper as follows. Section 2 provides the basic definitions and mathematical formulations of the flow models. In Section 3, we introduce abstract temporally repeated flow to solve the maximum dynamic flow problem with intermediate storage. Similarly, we solve abstract temporally repeated MDCF problem with intermediate storage in Section 4. The paper is concluded in Section 5. To the best of our knowledge, problems introduced in Section 3 and Section 4 and their solution strategies with temporally repeated flow are investigated for the first time.

2. MATHEMATICAL FORMULATION OF FLOW MODELS

In this section, we give basic mathematical denotations that are used throughout the paper. We also present static as well as dynamic flow models for abstract network with intermediate storage.

2.1. Basic Denotations. Consider a network $\mathcal{N} = (E, \mathcal{P})$ with finite set of elements E and the collection of paths

$$\mathcal{P} = \{P \subseteq E : P \text{ has a linear order } <_P \text{ of elements in } P\} \subseteq 2^E.$$

The collection \mathcal{P} of paths contains source-sink (s - t) paths P as well as intermediate paths $P_{[s \rightarrow e]}$ from source $s \in E$ to an intermediate element $e \in E$. Each element $e \in E$ has the non-negative integral movement capacity $u_e : E \rightarrow \mathbb{Z}^+$ which is used to send flow from the element e to its adjacent element and the storage capacity $v_e : E \rightarrow \mathbb{Z}^+$ which is used to hold flow at e . The order of elements in the path $P \in \mathcal{P}$ is denoted by $<_P$. An element $a \in P$ is left of e on P if $a <_P e$ and right of e if $a >_P e$. Similarly, $e \in P$ is said to be leftmost or first (rightmost or last) element of P if there does not exist a in P such that $a <_P e$ ($a >_P e$). For s - t path P , source s is the

leftmost element and sink t is the rightmost element. Let $E_I = E \setminus \{s, t\}$ be the set of intermediate elements.

Network $\mathcal{N} = (E, \mathcal{P})$ is an abstract network if it satisfies the switching property: $\forall P, Q \in \mathcal{P}$ and intermediate element $e \in P \cap Q$, $\exists R \in \mathcal{P}$ such that $R \subseteq P \times_e Q$ where,

$$P \times_e Q = \{a \in P : s \leq_P a \leq_P e\} \cup \{a \in Q : e \leq_Q a \leq_Q t\}.$$

Similar definition for switched path $R \subseteq Q \times_e P$ can be obtained. For simplicity, we use the notations

$$P_{[s \rightarrow e]} = \{a \in P : s \leq_P a \leq_P e\} \text{ and } P_{[e \rightarrow t]} = \{a \in P : e \leq_P a \leq_P t\}$$

to represent the elements on path P from source s up to e and that begins from e up to sink t , respectively. Similarly,

$$P_{[s \rightarrow e)} = \{a \in P : s \leq_P a <_P e\}, \text{ and } P_{(e \rightarrow t]} = \{a \in P : e <_P a \leq_P t\}$$

represent the elements on path P that are left of e and right of e , respectively. If P and Q are two paths both containing e_1 and e_2 , then it is possible to have $e_1 <_P e_2$ but $e_1 >_Q e_2$.

Throughout the paper, we consider that the storage capacity at source and sink elements are sufficiently large, i.e., $v_s = v_t \leq \infty$ and that of intermediate elements are finite. The movement capacity of source and intermediate elements are finite (i.e., $u_s < \infty$) and that of sink is zero (i.e., $u_t = 0$). If the incoming movement capacity of an intermediate element $e \in E_I$ is more than the outgoing movement capacity, then the excess flow is used to store at e . Moreover, for the uniqueness of the solution, the storage capacity of $e \in E_I$ should be $v_e \geq \sum_{P \in \mathcal{P}: a \in P} u_a \forall a <_P e$. Furthermore, the incoming and outgoing movement capacities of source and sink are zero, respectively, except for contraflow network.

2.2. Abstract Static Flow Model with Intermediate Storage.

Consider an abstract network $\mathcal{N} = (E, \mathcal{P})$ with path-flow $x_P : P \rightarrow \mathbb{R}^+$. Every path-flow x_P induces a flow through each element, denoted by $x_{P(e)} = \sum_{P \in \mathcal{P}: e \in P} x_P$. For all $e \in E$, if $x_{P(e)} \leq u_e$ and $x_P \geq 0$ then the path-flow x_P is feasible and an element e is said to be saturated with respect to x if $x_{P(e)} = u_e$. We denote $x_e^{out} = \sum_{P \in A_e} x_P$ and $x_e^{in} = \sum_{P \in B_e} x_P$ as the total outflow from e and the total inflow into e , respectively, where A_e and B_e represent the set of outgoing paths from e and incoming paths into e . Let $c_e : E \rightarrow \mathbb{Z}^+$ be the cost of transmission of flow per unit from e to its right element so that $c_P = \sum_{e \in P} c_e$.

Hoffman [7] generalized the max-flow-min-cut theorem of Ford and Fulkerson [4] for abstract network flow without storage of flow at intermediate elements. Due to the bottleneck flow on each paths, sending the flow with full movement capacity from source element greater than the minimum cut capacity is impossible. To deal with this problem, Pyakurel and Dempe [19] introduced the concept of intermediate storage for classical network flow. Using this concept in abstract network flow, we aim to push the maximum flow outward from the source in which flow with minimum cut capacity reaches to the sink and rest of the flow is to be stored at intermediate elements.

We define the flow function $x_e : E_I \rightarrow \mathbb{R}^+$ as the excess flow stored at element $e \in E_I$. The linear program for abstract static network flow with intermediate storage, as presented in [24], is

$$\begin{aligned}
 (2.1) \quad & \max \quad \sum_{P \in \mathcal{P}} x_P + \sum_{e \in E_I} x_e \\
 (2.2) \quad & \text{s.t.} \quad \sum_{P \in \mathcal{P}: e \in P} x_P \leq u_e \quad \forall e \in E \\
 (2.3) \quad & 0 \leq x_e^{in} - x_e^{out} = x_e \leq v_e \quad \forall e \in E_I \\
 (2.4) \quad & x_P \geq 0 \quad \forall P \in \mathcal{P} \\
 (2.5) \quad & v_e \geq \sum_{P \in \mathcal{P}: a \in P} u_a \quad \forall e \in E_I, \quad a <_P e
 \end{aligned}$$

Equation (2.1) is an objective function that refers to maximize the total flow reaching at sink t and the excess flow stored at intermediate elements. Equation (2.2) represents the capacity constraint of each element. The left inequality of Equation (2.3) represents the non-conservation of flow whose right inequality indicates that the excess flow is bounded by the storage capacity of e . Similarly, Equation (2.4) represents the non-negativity of the flow on each path and lower bound of storage capacity is given in Equation (2.5).

2.3. Abstract Dynamic Flow Model with Intermediate Storage. Let us consider $\mathcal{N} = (E, \mathcal{P}, \tau, T)$ as the abstract dynamic network, where $\tau : E \rightarrow \mathbb{Z}^+$ be a non-negative transit time of element $e \in E$ that is necessary to transship the flow from e to its right element and $T \in \mathcal{T}$ be a time horizon. If e and a are two consecutive elements on path P with $e <_P a$, then flow traveling through e at time θ reaches a at time $\theta + \tau_e$. In discrete time setting, time horizon is discretized as $\mathcal{T} = \{0, 1, \dots, T\}$. For each s - t path $P \in \mathcal{P}$, $\tau_P = \sum_{a \in P_{[s \rightarrow t]}} \tau_a$ denotes the traversal time of flow from

s to t and $\tau_{P_{[s \rightarrow e]}} = \sum_{a \in P_{[s \rightarrow e]}} \tau_a$ denotes the traversal time of flow from s to intermediate element e through the path $P_{[s \rightarrow e]}$.

Let $\psi_P(\theta) : P \times T \rightarrow \mathbb{R}^+$ be the dynamic s - t path-flow in discrete time $\theta \in \mathcal{T}$ and $\psi_e(\theta) : E_I \times T \rightarrow \mathbb{R}^+$ be the amount of excess flow stored at intermediate element $e \in E_I$ within time $\theta \in \mathcal{T}$. Let $\psi_e^{out} = \sum_{P \in A_e} \psi_P$ and $\psi_e^{in} = \sum_{P \in B_e} \psi_P$ denote the total outflow from e and inflow into e , respectively. As in [24], the abstract dynamic flow model with intermediate storage is

$$\begin{aligned}
 (2.6) \quad & \max \quad \sum_{\theta=\tau_P}^T \sum_{P \in \mathcal{P}} \psi_P(\theta) + \sum_{\theta=\tau_{P_{[s \rightarrow e]}}}^T \sum_{e \in E_I} \psi_e(\theta) \\
 (2.7) \quad & \text{s.t.} \quad \sum_{P \in \mathcal{P}: e \in P} \psi_P(\theta) \leq u_e(\theta) \quad \forall e \in E, \quad \theta \in \mathcal{T} \\
 (2.8) \quad & 0 \leq \psi_e^{in}(\theta) - \psi_e^{out}(\theta) = \psi_e(\theta) \leq v_e \quad \forall e \in E_I, \quad \theta \in \mathcal{T} \\
 (2.9) \quad & \psi_P \geq 0 \quad \forall P \in \mathcal{P} \\
 (2.10) \quad & v_e \geq T \sum_{P \in \mathcal{P}: a \in P} u_a \quad \forall e \in E_I, \quad a <_P e,
 \end{aligned}$$

The objective function in Equation (2.6) is to maximize the total flow reaching at t and the excess flow stored at intermediate elements within time horizon T . Similarly, Equation (2.7) represents the capacity constraint of each element at $\theta \in \mathcal{T}$ and Equation (2.9) represents the non-negativity of the flow on each path. Non-conservation of the flow at each time step θ is presented by the left inequality of Equation 2.8 whose right inequality shows that the excess flow is bounded by the storage capacity of e at each time $\theta \in \mathcal{T}$. The lower bounds of storage capacity of intermediate elements is represented in Equation (2.10) which is essential for the existence of temporally repeated flow.

3. ABSTRACT TEMPORALLY REPEATED MDF WITH INTERMEDIATE STORAGE

For a dynamic network \mathcal{N} , let u and τ be non-negative capacity and transit time, respectively, that are assigned to transship the flow from one element to its adjacent element via some path P . As in Kappmeier [9], we define the time expanded ground set \mathcal{E}_T by creating $T + 1$ copies of elements for each time step $\theta \in \mathcal{T} = \{0, 1, \dots, T\}$ as

$$\mathcal{E}_T = \{e^\theta : e \in E, \theta \in \mathcal{T}\}.$$

Flow starting from the source element at time θ reaches to an intermediate element e along path P at $\theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a$. The set of temporal paths P^θ associated with path $P \in \mathcal{P}$ that start from source element at time step θ is

$$P_{[s \rightarrow e]}^\theta = \left\{ e^\beta \in \mathcal{E}_T : e \in E, \beta = \theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a \right\}.$$

For the convenient notation, we use P^θ instead of source-sink temporal paths $P_{[s \rightarrow t]}^\theta$. The order of elements in temporal path is same as in original path P . We represent the set of all paths reaching to the intermediate element e and the sink element t within time horizon T by $\mathcal{P}_{[s \rightarrow e], T}^\theta$ and \mathcal{P}_T^θ , respectively, and defined as

$$\mathcal{P}_{[s \rightarrow e], T}^\theta = \left\{ P_{[s \rightarrow e]}^\theta : P_{[s \rightarrow e]} \subset P \in \mathcal{P}, \theta \in \mathcal{T}, \theta + \sum_{a \in P_{[s \rightarrow e]}} \tau_a \leq T \right\}$$

and

$$\mathcal{P}_T^\theta = \left\{ P^\theta : P \in \mathcal{P}, \theta \in \mathcal{T}, \theta + \sum_{e \in P} \tau_e \leq T \right\},$$

Since the paths in $(\mathcal{E}_T, \mathcal{P}_T^\theta)$ may not satisfy the switching property, Kappmeier [9] presented time expanded network to obtain abstract maximum dynamic flow where paths P^δ with delay pattern δ are used to satisfy the switching property. Using this concept, we define the set of temporal paths with delay pattern δ arriving at the intermediate element e and destination sink t

within time T as

$$\mathcal{P}_{[s \rightarrow e], T}^\delta = \left\{ P_{[s \rightarrow e]}^\delta : P_{[s \rightarrow e]} \subset P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{a \in P_{[s \rightarrow e]}} (\tau_a + \delta_a) \leq T \right\}$$

and

$$\mathcal{P}_T^\delta = \left\{ P^\delta : P \in \mathcal{P}, \delta \in \{0, 1, \dots, T\}^P, \sum_{e \in P} (\tau_e + \delta_e) \leq T \right\},$$

respectively.

Lemma 3.1 ([9]). *If an abstract network $\mathcal{N} = (E, \mathcal{P})$ preserves the order on paths, then the network $\mathcal{N}_T = (\mathcal{E}_T, \mathcal{P}_T^\delta)$ with path system \mathcal{P}_T^δ is an abstract network.*

Now, we introduce the abstract maximum dynamic flow problem with intermediate storage as follows.

Problem 1. *For a given abstract dynamic network $\mathcal{N} = (E, \mathcal{P}, \tau, T)$, an abstract maximum dynamic flow problem with intermediate storage is to find the maximum flow leaving the source element that is to be sent to the sink via s - t paths $P \in \mathcal{P}$ by allowing the maximum storage of excess flow at intermediate elements e via paths $P_{[s \rightarrow e]}$ $\forall e \in E_I$ with storage capacity $v_e \geq T \sum_{P \in \mathcal{P}: a \in P} u_a \forall a <_P e$ within the given time horizon T .*

To solve the problem, we begin the procedure by fixing the priority of elements. As in Pyakurel and Dempe [19], first priority is given to the sink to transship as much flow as possible. The excess flow is to be stored at the intermediate elements with priority order as follows: For each $e \in E_I$ with storage capacity $v_e \geq T \sum_{P \in \mathcal{P}: a \in P} u_a \forall a <_P e$, calculate the shortest distance $d_{P_{[s \rightarrow e]}}$ from s by using algorithm of Dijkstra [2]. It is to be noted that the temporally repeated solution exists only if the elements have upper bound capacities. The minimum cost path is considered as the shortest path and the priority is given to the farthest element among the elements of shortest distance. For example, if $d_{P_{[s \rightarrow e_1]}} > d_{P_{[s \rightarrow e_2]}}$ for $e_1, e_2 \in E_I$, then e_1 is higher in priority than e_2 and is denoted by $e_1 \succ e_2$.

Let e_1, \dots, e_r be r intermediate elements with priority order $e_1 \succ e_2 \succ \dots \succ e_r$ then set of all prioritized element including sink is $D = \{t \succ e_1 \succ \dots \succ e_r\}$. For the notational convenient, if we write $t = e_0$ then set of prioritized elements becomes $D = \{e_0 \succ e_1 \succ \dots \succ e_r\}$. As a solution strategy, we introduce the temporally repeated solution with intermediate storage and present an algorithm to solve Problem 1 herein.

3.1. Temporally Repeated Flow with Intermediate Storage. Temporally repeated flow, introduced by Ford and Fulkerson, induces the maximum dynamic flow from the source to the sink by repeating the constant rate of flow on source-sink paths with fixed transit times. Due to the flow conservation constraint, every flow out from the source may not reach at the sink element. Thus, the settlement of excess flow at intermediate elements plays an important role in evacuation planning, [19]. To deal with this problem, we introduce the temporally repeated flow in abstract network that transship the flow to the sink and intermediate elements with priority order which maximizes the flow out from the source element.

As the storage capacity of each prioritized intermediate element is $v_e \geq T \sum_{P \in \mathcal{P}: a \in P} u_a \quad \forall e \in E_I$, $a <_P e$, the temporally repeated flow on the sink and intermediate elements can be obtained through the paths with waiting pattern δ as follows, [24]:

For $e_0 = t$ and $P \in \mathcal{P}_T^\delta$ with $\psi_P = \min\{u_a : a \in P\}$,

$$(3.1) \quad |\psi|_{e_0, T} = \sum_P (T - \tau_P + 1) \cdot \psi_P$$

For intermediate element $e_i \in E_I$ and $P_{[s \rightarrow e_i]} \in \mathcal{P}_{[s \rightarrow e_i], T}^\delta \subseteq \mathcal{P}_T^\delta$ with $e_i <_P e_j$,

$$(3.2) \quad |\psi|_{e_i, T} = \sum_{P_{[s \rightarrow e_i]}} \left[(T - \tau_{P_{[s \rightarrow e_j]}} + 1) \cdot \psi_{e_i} + (\tau_{P_{[s \rightarrow e_j]}} - \tau_{P_{[s \rightarrow e_i]}}) \cdot \psi_{P_{[s \rightarrow e_i]}} \right]$$

where, $|\psi|_{e_i, T}$ is the net flow at element e_i within time T , $\psi_{P_{[s \rightarrow e_i]}} = \min\{u_a : a \in P_{[s \rightarrow e_i]}\}$ and $\tau_{P_{[s \rightarrow e_i]}} = \sum_{a \in P_{[s \rightarrow e_i]}} (\tau_a + \delta_a)$.

Now we present an algorithm to solve Problem 1.

Algorithm 1: Abstract temporally repeated MDF algorithm with intermediate storage

Input : Given abstract dynamic network $\mathcal{N} = (E, \mathcal{P}, \tau, T)$.

Output: Abstract temporally repeated MDF with intermediate storage on \mathcal{N} .

- (1) Compute the shortest distance $d_{P_{[s \rightarrow e]}} \quad \forall e \in E$ by taking transit time as cost and using Dijkstra's algorithm.
 - (2) Fix the priority order as $t = e_0 \succ e_1 \succ \dots \succ e_r$ with first priority to the sink $t = e_0$ and priority for intermediate elements as $d_{P_{[s \rightarrow e_i]}} > d_{P_{[s \rightarrow e_{i+1}]}} \implies e_i \succ e_{i+1}$, for $i = 1, \dots, r - 1$.
 - (3) Calculate the temporally repeated flow for sink element by using Equation 3.1 and for the the intermediate elements by using Equation 3.2.
-

Theorem 3.2. *Abstract maximum dynamic flow with intermediate storage obtained by Algorithm 1 using temporally repeated flow is optimal for sink as well as each intermediate elements.*

Proof. First we prove the feasibility of Algorithm 1. Step 1 is to find the shortest distance of each element by using Dijkstra's algorithm and Step 2 is to set priority order of elements, so both steps are feasible. In Step 3, maximum flow at the sink element is obtained as in Ford and Fulkerson [5] but for the intermediate elements, flow is stored using paths $P_{[s \rightarrow e]}$, $e \in E_I$ within the time horizon T . So, the flow obtained at each element is feasible.

Next, the optimality of Algorithm 1 is assured by the optimality of Step 3. The optimal flow at sink element is obtained by [5]. Each flow out from the source element that can not reach the sink is stored at either of the intermediate element in priority order. Thus, Algorithm 1 provides the optimal abstract temporally repeated MDF with intermediate storage within the given time horizon T . \square

Theorem 3.3. *Algorithm 1 solves Problem 1 in polynomial time.*

Proof. Since the time complexity of Step 1 is $O(|E|^2)$ and by sorting algorithm, complexity of Step 2 is $O(|E| \log(|E|))$. Similarly, temporally repeated formula is applied for each element, so Step 3 can be computed within the time $O(|E|)$. Thus, Algorithm 1 solves an abstract MDF problem with intermediate storage in polynomial time. \square

4. ABSTRACT TEMPORALLY REPEATED MDCF WITH INTERMEDIATE STORAGE

In this subsection, we discuss the computation of an abstract maximum dynamic contraflow (MDCF) problem with intermediate storage and solve it by using temporally repeated solution. In two way network topology, contraflow means the reversal of the direction of movement capacity of element towards the destination. At the time of disasters, every individual aims to move away from the danger zones towards the safe zones so that paths towards the danger zone are almost empty. Thus, contraflow technique helps to increase the movement capacity of elements by utilizing the unused paths so that maximum amount of flow can be sent within the given time horizon.

Consider an abstract dynamic contraflow network $\mathcal{N} = (E, \overleftrightarrow{\mathcal{P}}, \tau, T)$, where E represents the set of elements and $\overleftrightarrow{\mathcal{P}} = \overrightarrow{\mathcal{P}} \cup \overleftarrow{\mathcal{P}} \cup \overrightarrow{\mathcal{P}}_{[s \rightarrow e]} \cup \overleftarrow{\mathcal{P}}_{[e \rightarrow s]}$ represents the set of two-way paths. Here, $\overrightarrow{\mathcal{P}}$ and $\overleftarrow{\mathcal{P}}$ represent forward ($P_{[s \rightarrow t]}$) and backward ($P_{[t \rightarrow s]}$) source-sink paths, respectively. Similarly, we represent $\overrightarrow{\mathcal{P}}_{[s \rightarrow e]}$ and $\overleftarrow{\mathcal{P}}_{[e \rightarrow s]}$ for forward and backward intermediate paths, respectively. Our assumption is that the time to transship the flow from an element to its adjacent element in either direction is same. Let $\tau : E \rightarrow \mathbb{Z}^+$ be a symmetric transit time between pair of consecutive elements along a path. Then, $\tau_{\overrightarrow{\mathcal{P}}_{[e \rightarrow a]}} = \tau_{\overleftarrow{\mathcal{P}}_{[a \rightarrow e]}}$ with $e <_{\overrightarrow{\mathcal{P}}} a$ and $a <_{\overleftarrow{\mathcal{P}}} e$ so that $\tau_{\overrightarrow{\mathcal{P}}} = \tau_{\overleftarrow{\mathcal{P}}}$ and $\tau_{\overrightarrow{\mathcal{P}}_{[s \rightarrow e]}} = \tau_{\overleftarrow{\mathcal{P}}_{[e \rightarrow s]}}$. The temporal component T represents the time horizon. For static network, transit time τ is considered as cost c and time horizon T is absent. Contrary to general abstract network, incoming movement capacity to the source and outgoing movement capacity from the sink are nonzero for abstract contraflow network. Here, we introduce an abstract MDCF problem with intermediate storage and solve it by using temporally repeated flow.

Problem 2. For a given abstract dynamic network $\mathcal{N} = (E, \overleftrightarrow{\mathcal{P}}, \tau, T)$, abstract maximum dynamic contraflow problem with intermediate storage is to find the maximum flow leaving the source element that is to be sent to sink via s - t paths $\overrightarrow{\mathcal{P}} \cup \overleftarrow{\mathcal{P}}$ and allowing the storage of excess flow at intermediate elements e via paths $\overrightarrow{\mathcal{P}}_{[s \rightarrow e]} \cup \overleftarrow{\mathcal{P}}_{[e \rightarrow s]} \forall e \in E_I$ with storage capacity $v_e \geq T \sum_{P \in \mathcal{P}; a \in P'} u_a \forall a <_{P'} e$, $P' = \overrightarrow{\mathcal{P}} \cup \overleftarrow{\mathcal{P}}$ within given time horizon T by reverting the direction of paths $\overrightarrow{\mathcal{P}}$ and $\overleftarrow{\mathcal{P}}_{[e \rightarrow s]}$ at time zero.

To solve the problem, we construct an auxiliary network $\bar{\mathcal{N}} = (E, \bar{\mathcal{P}}, \bar{\tau}, T)$ by adding two-way movement capacities between two consecutive elements. Here, $\bar{\mathcal{P}}$ represent the set of paths in an auxiliary network obtained by reverting the direction of paths $\overleftarrow{\mathcal{P}}$ and $\overleftarrow{\mathcal{P}}_{[e \rightarrow s]}$ at time zero. The movement capacity \bar{u}_e and transit time $\bar{\tau}_e$ are defined as follows: For any two consecutive elements e and a with $e <_{\overrightarrow{\mathcal{P}}} a$ and $a <_{\overleftarrow{\mathcal{P}}} e$

$$\bar{u}_e = u_{e:e \in \overrightarrow{\mathcal{P}}} + u_{a:a \in \overleftarrow{\mathcal{P}}}$$

where $u_{a:a \in \overleftarrow{\mathcal{P}}} = 0$ if $a \notin \overleftarrow{\mathcal{P}}$ and

$$\bar{\tau}_e = \begin{cases} \tau_{e:e \in \vec{P}} & \text{if } e <_{\vec{P}} a \\ \tau_{a:a \in \overleftarrow{P}} & \text{otherwise.} \end{cases}$$

We now present an algorithm to solve abstract MDCF problems using temporally repeated flow. We first transform the given two-way network to an auxiliary network $\bar{\mathcal{N}} = (E, \bar{\mathcal{P}}, \bar{\tau}, T)$ and construct cycle free paths on $\bar{\mathcal{N}}$. On these cycle free paths, we solve abstract maximum dynamic flow problem as described in Section 3 by using Algorithm 1 to obtain optimal solution to the corresponding contraflow problem with intermediate storage.

Algorithm 2: Abstract temporally repeated MDCF algorithm with intermediate storage

Input : Given abstract two-way network $\mathcal{N} = (E, \overleftrightarrow{P}, \tau, T)$.

Output: Abstract temporally repeated MDCF with intermediate storage on \mathcal{N} .

- (1) Construct an auxiliary network $\bar{\mathcal{N}} = (E, \bar{\mathcal{P}}, \bar{\tau}, T)$.
 - (2) Construct cycle free paths on $\bar{\mathcal{N}}$ satisfying the switching property.
 - (3) Compute abstract maximum dynamic flow on $\bar{\mathcal{N}}$ with intermediate storage using Algorithm 1.
 - (4) A path $\overleftarrow{P}(\overleftarrow{P}_{[e \rightarrow s]})$ is reversed if and only if the flow along path $\vec{P}(\vec{P}_{[s \rightarrow e]})$ is greater than its capacity or if there is a non-negative flow along path $\vec{P}(\vec{P}_{[s \rightarrow e]}) \notin \overleftrightarrow{P}$.
-

Theorem 4.1. *An optimal solution to an abstract MDCF problem with intermediate storage can be obtained from Algorithm 2 by using temporally repeated flow.*

Proof. As each step of Algorithm 2 are feasible, solution obtained from it is feasible. The optimality of Algorithm 2 is dominated by the optimality of Step 3. We compute the abstract MDF in auxiliary network in polynomial time complexity by using Algorithm 1, which is optimal. So Algorithm 2 provides an optimal solution to an abstract MDCF problem with intermediate storage using temporally repeated flow. \square

Here, Step 1 of Algorithm 2 can be computed in $O(|E|)$ time and Step 2 can be obtained in polynomial time, [8]. Similarly, Step 3 can be obtained in polynomial time complexity by using Algorithm 1. So the polynomial time solvability of Algorithm 2 is at hand.

Theorem 4.2. *Algorithm 2 solves the abstract MDCF problems with symmetric transit times in polynomial time complexity.*

5. CONCLUSION

Abstract network permits the flows on paths rather than on arcs satisfying the switching property. In literature, the lexicographically maximum flow, maximum flow over time and earliest arrival flow problems have been solved efficiently in abstract networks. In this paper, we have presented the abstract flow models with intermediate storage in static and dynamic networks. We have solved the MDF problem with intermediate storage by using temporally repeated flow in polynomial time complexity. Similarly, for two way network topology, we have solved abstract

MDCF problem by using temporally repeated flow and present a polynomial time algorithm to solve it . To the best of our knowledge, temporally repeated flow with intermediate storage to solve MDF and MDCF problems in an abstract network topology is introduced for the first time.

Conflict of Interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Data Availability The authors have not used any additional data in this article.

Funding No funding from any source is available for this research.

Acknowledgements The first author (Durga Prasad Khanal) thanks to the German Academic Exchange Service - DAAD for Research Grants - Bi-nationally Supervised Doctoral Degrees/Cotutelle, 2021/22 and University Grants Commission Nepal for PhD Research Fellowship, 2020/21.

REFERENCES

- [1] T.N. Dhamala, S.P. Gupta, D.P. Khanal, U. Pyakurel, Quickest multi-commodity flow over time with partial lane reversals, *Journal of Mathematics and Statistics*, Vol. 16, pp. 198-211, 2020, DOI: <https://doi.org/10.3844/jmssp.2020.198.211>
- [2] E.W. Dijkstra, A note on two problems in connection with graph, *Numer. Math.*, Vol. 1(1), pp. 269-271, 1959.
- [3] L. Fleischer, E. Tardos, Efficient continuous-time dynamic network flow algorithms, *Operations Research Letters*, Vol. 23, pp. 71-80, 1998.
- [4] L.R. Ford, D.R. Fulkerson, Maximal flow through a network, *Canadian Journal of Mathematics*, Vol. 8, pp. 399-404, 1956.
- [5] L.R. Ford, D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New jersey, 1962.
- [6] S.P. Gupta, D.P. Khanal, U. Pyakurel, T.N. Dhamala, Approximate algorithms for continuous-time quickest multi-commodity contraflow problem, *The Nepali Mathematical Sciences Report*, Vol. 37, pp. 30-46, 2020, DOI: <https://doi.org/10.3126/nmsr.v37i1-2.34068>.
- [7] A.J. Hoffman, A generalization of max flow - min cut, *Math. Prog.*, Vol. 6, pp. 352-359, 1974.
- [8] J.-P.W. Kappmeier, J. Matuschke, B. Peis, Abstract flow over time: A first step towards solving dynamic packing problems, *Theoretical Computer Science, Algorithm and Computation*, Vol. 544, pp. 74-83, 2014.
- [9] P.W. Kappmeier, *Generalizations of flows over time with application in evacuation optimization*, PhD Thesis, Technical University, Berlin, Germany, 2015.
- [10] D.P. Khanal, U. Pyakurel, T.N. Dhamala, Prioritized multi-commodity flow model and algorithm, *International Symposium on Analytic Hierarchy Process (ISAHP2020)*, 2020, Paper DOI: <https://doi.org/10.13033/isahp.y2020.049>.
- [11] D.P. Khanal, U. Pyakurel, T.N. Dhamala, Maximum multicommodity flow with intermediate storage, *Mathematical Problems in Engineering, Hindawi*, 2021, <https://doi.org/10.1155/2021/5063207>.
- [12] D.P. Khanal, U. Pyakurel, S. Dempe, Dynamic contraflow with orientation dependent transit times allowing intermediate storage, *The Nepali Mathematical Sciences Report*, Vol. 38(2), pp. 1-12, 2021, DOI: <http://doi.org/10.3126/nmsr.v38i2.42700>.
- [13] D.P. Khanal, U. Pyakurel, T.N. Dhamala, S. Dempe, Efficient algorithms for abstract flow with partial switching, *SN Operations Research Forum*, Vol. 3(55), 2022, <https://doi.org/10.1007/s43069-022-00168-2>.
- [14] D.P. Khanal, U. Pyakurel, T.N. Dhamala, S. Dempe, Maximum multi-commodity flow with proportional and flow-dependent capacity sharing, *Comput. Sci. Math. Forum*, Vol. 2(5), 2022, DOI: <https://doi.org/10.3390/IOCA2021-10904>.
- [15] S. Kim, S. Shekhar, M. Min, Contraflow transportation network reconfiguration for evacuation route planning, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 20(8), pp. 1115-1129, 2008.
- [16] M. Martens, *Path-Constrained Network Flows*, PhD Thesis, Technical University, Berlin, Germany, 2007.

- [17] M. Martens, S.T. McCormick, A polynomial algorithm for weighted abstract flow, In *Integer Programming and Combinatorial Optimization* in Lecture Notes in Computer Sciences, Vol. 5035, pp. 97-111, 2008.
- [18] S.T. McCormick, A polynomial algorithm for abstract maximum flow, In *Proceeding of the 7th annual ACM-SIAM symposium on discrete algorithms*, pp. 490-497, 1996.
- [19] U. Pyakurel, S. Dempe, Network flow with intermediate storage: models and algorithms, *SN Operations Research Forum*, 2020, DOI:10.1007/s43069-020-00033-0.
- [20] U. Pyakurel, T.N. Dhamala, Continuous time dynamic contraflow models and algorithms, *Advances in Operations Research - Hindawi*; Article ID 368587, pp. 1-7, 2016.
- [21] U. Pyakurel, S. Dempe, Universal maximum flow with intermediate storage for evacuation planning, In: *Kotsireas I.S., Nagurney A., Pardalos P.M., Tsokas A. (eds) Dynamics of Disasters. Springer Optimization and Its Applications*, 169, Springer, Cham. 2021, https://doi.org/10.1007/978-3-030-64973-9_14.
- [22] U. Pyakurel, T.N. Dhamala, S. Dempe, Efficient continuous contraflow algorithms for evacuation planning problems, *Annals of Operations Research (ANOR)*, Vol. 254, pp. 335-364, 2017.
- [23] U. Pyakurel, S.P. Gupta, D.P. Khanal, T.N. Dhamala, Efficient algorithms on multicommodity flow over time problems with partial lane reversals, *International Journal of Mathematics and Mathematical Sciences, Hindawi*, 2020, DOI: <https://doi.org/10.1155/2020/2676378>.
- [24] U. Pyakurel, D.P. Khanal, T.N. Dhamala, Abstract network flow with intermediate storage for evacuation planning, *European Journal of Operational Research*, 2022, <https://doi.org/10.1016/j.ejor.2022.06.054>.
- [25] U. Pyakurel, H.N. Nath, S. Dempe, T.N. Dhamala, Efficient dynamic flow algorithms for evacuation planning problems with partial lane reversal, *Mathematics*, Vol. 7, pp. 1-29, 2019.
- [26] U. Pyakurel, H.N. Nath, T.N. Dhamala, Partial contraflow with path reversals for evacuation planning, *Annals of Operations Research*, 2019, <http://doi.org/10.1007/s10479-018-3031-8>.
- [27] S. Rebenack, A. Arulselvan, L. Elefteriadou, P.M. Pardalos, Complexity analysis for maximum flow problems with arc reversals, *Journal of Combinatorial Optimization*, Vol. 19, pp. 200-216, 2010.

Appendix D

List of Presentations

- “International Webinar on Recent Advances in Pure and Applied Mathematics 2020 (RAPAM 2020)” August 24-25, 2020 at Kurseong College, Darjeeling, India, entitled “Approximation to Quickest Multi-Commodity Contraflow Over Time with Length Bound”.
- “International Conference of Computational Sciences – Modeling, Computing and Soft Computing (CSMCS 2020)” September 10-12, 2020 at NIT, Kerala, India, entitled “Length Bound Approximation to Quickest Multi-Commodity Contraflow Problem”.
- “International Symposium on Analytic Hierarchy Process 2020 (ISAHP2020)” December 3-6, entitled “Prioritized Multi-Commodity Flow Model and Algorithm”. Paper DOI- <https://doi.org/10.13033/isahp.y2020.049>
- “5th International Conference on Dynamics of Disaster, DOD 2021" July 16-18, 2021 Athens Greece, (Virtually) entitled "Efficient Algorithms for Abstract Flow with Partial Switching".
- “1st Online Conference on Algorithms (IOCA2021)", Sep 27- Oct 10, 2021, Germany, entitled “Maximum Multi-Commodity Flow with Proportional and Flow-Dependent Capacity Sharing”. Paper DOI- <https://doi.org/10.3390/IOCA 2021-10904>
- Presented poster on “Universe Winter School on Optimization, Games and Markets” organized at Chemnitz University of Technology, Germany, November 14-17, 2021 entitled “Multi-commodity Evacuation Planning with Intermediate Storage: A Max-Flow Problem”.
- Presented poster on “Managing Disaster Risk: A Way to Sustainability” organized by Nepal German Academic Association (NEGAAS) - Programme Migration & Diaspora (PDM) Activities, November 21-22, 2021 entitled “Prioritized Evacuation Planning with Multi-commodity Flow Model: A Quickest Flow Problem”.
- “International Online Conference on Optimization ICOP22" Fez, Morocco, January 19-

21, 2022, entitled "Multi-Commodity Flow Transmission with Intermediate Storage by Flow-Dependent Capacity Sharing on Arcs".

- "International Online Conference on Applied Mathematics IOCAM22" Fez, Morocco, June 1-3, 2022, entitled "Route Based Evacuation in Asymmetric Contraflow Network with Flow Circulation at Destination".
- Presented poster on "KATHMANDU HUMBOLDTKOLLEG 2022 (KHK-2022) - International Conference in Interdisciplinary Collaboration for Strengthening Science and Culture" at Kathmandu, Nepal supported by Alexander von Humboldt Foundation, Germany, October 16-19, 2022 entitled "Abstract Flow with Holding Capacity for Congestion Minimization".
- "6th International Conference on "Dynamics of Disasters, DOD 2023" July 3-6, Piraeus Athens, Greece, entitled "Prioritized Maximum Multi-Commodity Flow in Evacuation Planning".
- Presented poster and paper on "PhD Festival 2023, University Campus, Tribhuvan University Kathmandu, Nepal" 9-10 October 2023 entitled "Commodity Prioritized Multi-commodity Flow with Excess Storage".





Sixth
International
conference
on
Dynamics of Disasters

DOD 2023

<http://www.caopt.com/DOD2023/>

DOD 2023 co-General Chair:

Dr. Ilias S. Kotsireas, Professor
Dept. of Physics & Computing
Wilfrid Laurier University
75 University Avenue West
Waterloo Ontario N2L 3C5 Canada
Tel: (+1) 519 884-0710 ext. 2218
Fax: (+1) 519 746-0677
ikotsire@wlu.ca

CERTIFICATE

On behalf of the DOD 2023 Organizing
Committee,

I am pleased to certify that

Durga Prasad Khanal

Participated at the DOD 2023 conference
held in Athens, Greece and delivered a
talk.

Sincerely Yours

Ilias S. Kotsireas,

DOD 2023 co-General Chair



UNIVERS European Cross-Border University

Certificate

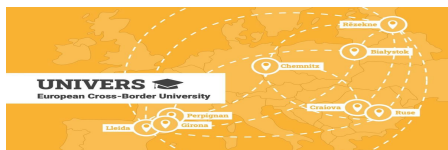
Durga Prasad Khanal
at Technische Universität Bergakademie Freiberg, Germany
successfully participated in UNIVERS WINTER SCHOOL on OPTIMIZATION, GAMES and MARKETS held at the Chemnitz University of Technology, 14.11.-17.11.2021, and contributed to a poster session presenting own research results.

Wladimir Shikhman

Prof. Dr. Vladimir Shikhman
 Chemnitz University of Technology
 Germany

Didier Aysel

Prof. Dr. Didier Aysel
 University of Perpignan
 France



DAAD Deutscher Akademischer Austauschdienst
 German Academic Exchange Service





Kurseong College
Dowhill Road, Kurseong,
Darjeeling, West Bengal, India

*A Two-days International Webinar on "Recent Advances in Pure and Applied Mathematics 2020
 (RAPAM 2020)
 24th - 25th August, 2020*



Certificate of Paper Presentation

This is to certify that Mr. Durga Prasad Khanal, Assistant Professor of Saraswati Multiple Campus has actively participated and presented paper entitled "Approximation to quickest multi-commodity contraflow over time with length bound" in two days International Webinar on "Recent Advances in pure and applied mathematics 2020". Organised by Kurseong College

Kanak Kanti Baishya

Dr. Kanak Kanti Baishya
 Convener

Samir Bal

Dr. Samir Bal
 Principal, Kurseong College



**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. Durga Prasad Khanal, Central Department of Mathematics, Tribhuvan University, 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 **Length Bound Approximation to Quickest Multi-commodity Contraflow Problem**.

Ashish Awasthi
Dr. Ashish Awasthi
 Convener

Sunil Jacob John
Dr. Sunil Jacob John
 Convener

Satyananda Panda
Dr. Satyananda Panda
 Chairperson

TEQIP - III



Wolfram
 Mathematics

AIP