Journal of Advanced College of Engineering and Management

A Brief Summary on Lexicographic Network Flow Problem

[1]Rajendra Paudyal, [2]Narayan Prasad Adhikari
[1,2]*Advanced College of Engineering and Management Tribhuvan University, Kathmandu, Nepal*
Email: rajendra.paudyal@acem.edu.np

DOI: 10.3126/jacem.v11i1.84550

Abstract

Since there have been more natural and man-made disasters in recent decades, evacuation preparation has become more important. A lexicographic network flow optimization model is one of the most effective approaches to simulating the evacuation scenario. The majority of the requirements for evacuation planning are covered by this model. Using models and solution strategies developed in this subject, this paper attempts to give an overview of lexicographic static and dynamic network flow problems linked to evacuation planning.

Keywords*—Evacuation, Lexicographic Network Flow Problem, Optimization*

1 Introduction

Network flow theory can be considered as one of the most thriving fields concerning optimization. It can be widely used in various applications in the fields of Science and Technology. In addition to this, it has clear applications regarding transportation and scheduling. The main aim of the Network Flow Problems (NFP) can be summarized as to transfer a commodity from one place to another through a network, using the available paths efficiently.

There are two important parts of a network and they are known as nodes and edges. A static network may not be capable enough to capture complex details of many real world scenarios, evacuation, for instance [1]. In dynamic networks,each edge is equipped with transit time in addition to capacity limit, where transit time can be defined as the time it takes for the flow to travel from one node to another [2]. Considering discrete time manner NFP, the edge capacity can be defined as the flow amount entering at each step of time. However, when continuous time manner NFP is taken into consideration, it makes quite sense to consider the transit times showing continuous behaviour, where edge capacity defines the entry flow rate the edge.In some real world problems, transit times are dependent on congestion levels.

Most studied network flow Problem (NFP) are maximum flow problems,these can be both static as well as dynamic in nature; quickest flow problems, earliest arrival flow problems, transshipment problems; and minimum cost flow problems adopting the various parameters based on the field of their applications. The extension of these problems with time parameters provides a crucial characteristics in real-world applications, likewise in evacuation scenarios, data transmission, telecommunication, scheduling, etc.

Observing the long history of network flow problems, it is seen that the common feature in network flow model is that the units of flow satisfy the flow conservation constraints for intermediate nodes. However, a few years back, Bhandari and Khadka in 2019, proposed ”A network flow model with non-conservation flow constraints at intermediate nodes”. Authors studied various network flow problems based on this model since then and justified the applicability of this flow model in evacuation planning problems. Taking Evacuation planning problems into consideration, specifically those modeled with the principle of conservation of flow at intermediate nodes, evacuees are allowed to exit the source in case they can successfully arrive at the sink. Generally, it is uncertain in advance to determine the number of people at source and the ability of the road and sink to accommodate the given number of people under disaster like floods, landslides, etc. or panic situations at given time frame. Therefore, evacuators should make a strong effort to relocate as many evacuees as possible to safer locations, where they can get healthcare support or other essential aids or establish urgency levels among evacuees for transportation towards the sink.The adjustment of flow conservation conditions change the available flow model making it suitable for representing the scenarios where the holding of evacuees at safer locations is practically feasible. Intermediate nodes and sink are considered as a terminal set which are given

Journal of Advanced College of Engineering and Management

priority based on assigned capacity in most cases. The ranking of priority is determined by the evacuation conditions: considering the factors like the amenities available at the sink, how far are the source and their storing capacities.

The network flow problems with such prioritization are known as lexicographic network flow problems (LNFP). Besides evacuation, network flow models and their solution mechanism, with intermediate holding capability of flow units based on priority criteria are of great interest in many real world application.

2 Literature Review

The maximum static flow problem aiming to send maximum flow from source to sink was first studied by Ford and Fulkerson in (1956) [1]. The classical Ford- Fulkerson algorithm and its variants–the [2]Edmonds-Karp Algorithm (1972) [3] and Dinic Algorithm (1970) [4] are commonly used to solve this problem by iteratively finding augmenting paths and updating the flow values. Efficient algorithm, known as Push-Relable Algorithm, is due to Goldberg and Tarjan (1988) [5].

Ford and Fulkerson in 1958 introduced maximum dynamic flow (MDF) along with introducing concepts like flow decomposition focusing on capacity constraints and further studied in 1962, and introudcued key algorithms and concept to analyze NFP, giving more emphasis on capacity limitations and their consequences. [2, 6]. Literature is flourished with the extension of MDF problem, in particular, quickest flow problems Burkard et al. (1993) [7], Fleischer and Skutella (2007) [8], etc., Ruzika et al. (2011) [9], Dhamala and Pyakurel (2013) [10], Khadka and Bhandari (2020)[11]. Dhamala in 2015 provides a thorough examination of models and algorithms for discrete evacuation planning network problems so as to improve strategies for evacuation in emergency situations. [12], Khadka and Bhandari (2017) [13], Pyakurel et al.(2017) [14] etc. Megiddo, (1977) [15] introduced an efficient algorithm for finding lexicographically optimal flows in NFP by emphasizing the importance of lexicogrpahic sequencing of multiple objectives optimization. The authors demonstrated that given problem is possible to be resolved even within polynomially bounded time. The paper in [16] introduces a particular framework for minimizing transportation time in networks by establishing optimal conditions.

Bhandari (2021) [17], in his PhD thesis, examined a flow model for network with capacitated nodes in a given priority order where flow conservation may not be proved right .They introduced a novel approach to optimize flow in networks by giving emphasis to lexicographic optimization in order to have clear understanding of contraflow problems in operations research. The research paper [18] provides a novel approach to optimize dynamic flow in networks while vertex capacities are also taken into consideration through a robust mathematical framework, an efficient algorithm and practical implications. The authors Bhandari and Khadka (2023) [19] introduced a particular framework to optimize evacuation routes by giving emphasis to prioritized destinations and uniform path lengths. Their work also include efficient algorithm development, simulations and practical applications, providing significant contribution in the field of evacuation planning and emergency management.However their work has certain limitations like highlighting the need of further research to address real world complex problems and inclusion of dynamic factors.

3 Formulation of Lexicographic Network flow Problem

The mathematical model can be summarized as follows based on(2020) [18]

Let $G = (V, A)$ represent a directed graph consisting of a finite set of vertices V and a finite set of arcs A. We denote the number of nodes and edges as $n|V|$ and $m|A|$, respectively. Let s and d represent the source and destination nodes, respectively. It is assumed that the graph G does not contain any parallel edges or mesh structures. Furthermore, no edge is allowed to enter the source node s or exit the destination node d.

For any node $v \in V$, we denote:

- $\delta^-(v)$: the set of incoming edges to node v,
- $\delta^+(v)$: the set of outgoing edges from node v.

Journal of Advanced College of Engineering and Management

A minimum and maximum flow capacity function are defined as:

$$l, u : A \to \mathbb{N}_0\mathbb{N} \cup \{0\}$$

which specifies the lower and upper bounds on the flow units allowed on each edge in each time step. Typically, we assume:

$$l(a) = 0 \quad \text{for all } a \in A.$$

In addition, a transit time function is defined as:

$$\tau : A \to \mathbb{N}_0$$

where $\tau(a)$ represents the time required for a unit of flow to traverse edge $a \in A$.

Let $S \subset V$ be a terminal set defined as:

$$S\{v_1, v_2, \ldots, v_k\}$$

with a strict priority ordering:

$$v_1 > v_2 > \cdots > v_k \quad \text{where } v_1 = d.$$

A node capacity function is defined as:

$$k : S \to \mathbb{N}_0$$

which limits the number of flow units that can be accommodated at node $v \in S$. It is assumed that:

$$k(d) = \infty \quad \text{and} \quad k(v) < \infty \quad \text{for all } v \in S \setminus \{d\}.$$

Let a time horizon $T \in \mathbb{N}$ be given, and assume discrete time steps denoted by:

$$\tau = \{0, 1, \ldots, T\}.$$

The overall model is referred to as a real-time network and denoted by:

$$N = (G, l, u, \tau, T, k).$$

If we aim to describe a static network (i.e., without transit time), then it is simplified as:

$$N = (G, l, u, k).$$

The flow over time in network N is modeled by the function:

$$f : A \times \tau \to \mathbb{N}_0,$$

where $f(a, t)$ represents the number of flow units entering edge $a \in A$ at time step $t \in \tau$.

A flow unit entering edge a at time t arrives at the terminal node of edge a at time $t + \tau(a)$. The flow is subject to the edge capacity constraint:

$$0 \leq f(a, t) \leq u(a) \quad \text{for all } a \in A,\ t \in \tau.$$

To ensure no flow enters an edge after the time frame would exceed the horizon, we require:

$$f(a, t) = 0 \quad \text{for all } t > T - \tau(a),\ a \in A.$$

Journal of Advanced College of Engineering and Management

The surplus of a node $v \in V$ at time $t \in \tau$ is defined as:

$$\mathrm{ex}_f(v,t) \sum_{a \in \delta^-(v)} \sum_{p=0}^{t-\tau(a)} f(a,p) - \sum_{a \in \delta^+(v)} \sum_{p=0}^{t} f(a,p)$$

Consequently, it is of utmost requirement to make sure that $\mathrm{ex}_f(v,T) \leq k(v)$ for all $v \in \tau$.

The objective function of the maximum flow evacuation planning problem aims to maximize the vector in a lexicographic order.

$$\big(\mathrm{ex}_f(v_1,T),\ldots,\mathrm{ex}_f(v_k,T)\big)^T$$

such that $\mathrm{ex}_f(v_i,T) \leq k(v_i)$ for $i = 1,\ldots,k$. Note that $k(v_1) = k(d) = \infty$ and $v_i \in \tau$ for $i = 1,\ldots,k$.

4 Solution Procedure

We described the solution procedure developed in (2020) [18]

4.0.1 Solution for LexMSF Problem

Let $N = (G, l(a), u(a), k(v))$ be a static network without transit times.

- Let $S = \{v_1,\ldots,v_k\}$ be prioritized from higher to lower priority.
- Assume $l(a) = 0$ for all $a \in A$.
- Goal: Compute a lexicographic maximum flow in N that satisfies:
  - arc capacities for all arcs
  - vertex capacities for all $v \in S$

Max-flow computations on a transformed network:

- Introduce an artificial vertex v_i' for each $v_i \in S$.
- Let the artificial vertex $v_1' := d$ be the supersink.
- Connect v_i and v_i' by arc (v_i, v_i') with $u(v_i, v_i') = k(v_i)$.
- Add arc (v_i', d) with $u(v_i', d) = 0$.
- Add arc (d, d) with $u(d, d) = \infty$.
- Set $l(a) = 0$ for all arcs (original and artificial).

Thus, the original network $N = (G, l, u, k)$ is transformed into $N' = (G', l, u)$ with $G' = (V, A)$.

4.0.2 Solution for LexMDF Problem with Capacitated nodes

Let $N = (G, l(a), u(a), \tau(a), k(v), T)$ be a dynamic network, where $\tau(a)$ denotes the transit time function and T is the time horizon. Let $S = \{v_1,\ldots,v_k\}$ be a set of prioritized vertices from higher to lower priority, and assume that $l(a) = 0$ for all $a \in A$.

To solve the LexMDF problem, we first transform the dynamic network N into a time-expanded network $N^T = (V^T, A^T, l, u)$ defined as follows:

The set of time-expanded vertices is:

$$V^T := \{v^t \mid v \in V,\ t = 0,\ldots,T\}.$$

The set of time-expanded arcs is:

$$A^T := \{(v^t, w^{t+\tau(vw)}) \mid (v,w) \in A\}.$$

Journal of Advanced College of Engineering and Management

All lower capacities are zero, i.e.,

$$l(a) = 0 \quad \text{for all } a \in A^T.$$

The upper capacity function is preserved as:

$$u(v^t, w^{t+\tau(vw)}) = u(vw) \quad \text{for all } (v, w) \in A.$$

Additionally, storage arcs of the form (v^t, v^{t+1}) are added for all $v \in V$, with infinite capacity:

$$u(v^t, v^{t+1}) = \infty.$$

As in the static case, for each terminal $v_i \in S$, a corresponding vertex $v_i := v_i^T$ is introduced at the final time step. Each v_i^T is connected to v_i via an arc with upper capacity $k(v_i)$, representing the vertex capacity constraint. Since both vertices are on the same time level, the arc has zero transit time.

A static maximum s^0-d flow in the time-expanded network corresponds to a discrete dynamic s-d flow in N. After computing this maximum s^0-d flow, we fix the flow through the arc (d^T, d) by setting both its lower and upper bounds to the obtained flow value.

Next, an arc from the next highest-priority vertex v_i to d is introduced with infinite capacity, and a new static maximum s^0-d flow is computed in the modified time-expanded network. This process is repeated iteratively to compute the lexicographically maximum dynamic flow that respects both vertex capacities and priorities.

Given a dynamic network

$$N = (G, l(a), u(a), \tau(a), k(v), T, s, d),$$

where:

- s is the source,
- $S = \{v_1, \ldots, v_k\} \subseteq V$ is the prioritized terminal set,
- $d = v_1 \prec v_2 \prec \cdots \prec v_k$,
- $l(a) = 0$ for all $a \in A$,

the DT-LexMDF (Discrete-Time Lexicographic Maximum Dynamic Flow) problem in N can be solved in pseudo-polynomial time.

4.1 Solution for LexMDF Problem with Sufficient Capacitated nodes

1. $V \leftarrow V \cup \{v_i'\}, \quad A \leftarrow A \cup \{(v_i, v_i')\} \quad$ for all $v_i \in S$.
2. Set $u(v_i, v_i') \leftarrow \infty$ and $\tau(v_i, v_i') \leftarrow 0$.
3. Update terminal set: $S = \{v_1', v_2', \ldots, v_k', s\}$, with $S^+ \leftarrow S \setminus \{s\}$, and $S^- \leftarrow \{s\}$.
4. Construct the inverse network N^{inv}, where all arcs in N are reversed.
5. Apply the algorithm of Hoppe and Tardos (2000) on N^{inv}.

Complexity: The above procedure runs in polynomial time.

Solution for LexMDF Problem on UPL Network

Consider a uniform path length (UPL) network:

$$N = (G, u(a), k(v), \tau(a), s, d, T),$$

Journal of Advanced College of Engineering and Management

with terminal set $S \subseteq V$ given as $S := \{v_1, \ldots, v_k\}$ and priority order:

$$d = v_1 > v_2 ..... > v_k.$$

The goal is to solve the LexMDF problem in polynomial time using temporally repeated flows.

The key idea is to:

- Find all s-v_i paths at each time step $t \leq T$,
- Send flow along the longest available paths,
- Decompose flow using the **Lexicographic Minimum Cost Circulation (LexMCC)** problem.

To use LexMCC, perform the following steps:

- Use a minimum cost circulation algorithm iteratively.
- For each $v_i \in S$, add an arc (v_i, s) with capacity $k(v_i)$ and transit time $T + 1$.
- Treat each transit time $\tau(a)$ as the cost $c(a)$ on arc $a \in A$.

This generates a set of paths:

$$\mathcal{P}_{v_i} := \text{All temporally repeatable } s\text{-}v_i \text{ paths},$$

where each path $P \in \mathcal{P}_{v_i}$ has the following attributes:

- $f(P)$: flow value per repetition,
- $\tau(P)$: total traversal time of path P,
- $I_t(P)$: time step when repetition starts,
- $F_t(P)$: time step when repetition ends.

LexMCC Algorithm Notes:

- Runs in strongly polynomial time,
- The LexMCC problem is solved at most $2n$ times for each terminal node.

We explore a polynomial-time solution for the LexMDF problem in uniform path length networks using temporally repeated flows.

Process Outline:

1. **Use the LexMCC (Lexicographic Minimum Cost Circulation) framework.**
2. **Treat transit times $\tau(a)$ as costs $c(a)$.**
3. **For each terminal v_i, add arc (v_i, s) with capacity $k(v_i)$ and transit time $T + 1$, making v_i a sink.**
4. **Solve LexMCC to get temporally repeatable s-v_i paths.**
5. **Each path $P \in \mathcal{P}_{v_i}$ has:**
   - **$f(P)$: flow per repetition,**
   - **$\tau(P)$: traversal time,**
   - **$I_t(P), F_t(P)$: repetition interval.**

This compact representation avoids full time expansion, making LexMDF solvable efficiently in UPL networks.

Journal of Advanced College of Engineering and Management

4.2 Solution for Lexicographic Earliest Arrival Flow Problem on UPL-TTSP Network

Despite its significance in evacuation planning, an earliest arrival flow does not necessarily exist in a general network with multiple sinks, even with a single source. However, networks with multiple sinks and zero arc transit times have been studied, with polynomial-time algorithms proposed in [20] and [21].

In [21], the objective is to maximize the ratios of flow values to sink capacities lexicographically, rather than strictly enforcing capacity constraints. A pseudo-polynomial time algorithm also exists for arbitrary transit times.

Here, we consider a different setting with a prioritized terminal set S, The Discrete-Time Lexicographic Earliest Arrival Flow (DT-LexEAF) problem seeks to fulfill its objective at every time step $t \leq T$: to send as many evacuees as early as possible from the risk zone to the prioritized safe zones.

Although every earliest arrival flow is a maximum dynamic flow for the time horizon, the converse does not hold for general networks.

We study the DT-LexEAF problem on a Uniform Path Length Two-Terminal Series-Parallel (UPL-TTSP) network:

$$N = (G, u(a), k(a), \tau(a), T),$$

where:

- **Vertex v_1 always has sufficient storage capacity.**
- **Other vertices v_i may have zero or sufficient capacities.**

The objective is to maximize the number of flow units sent to the prioritized terminals S at each time step.

The solution strategy is based on the MCC Algorithm , which solves the LexMCC problem iteratively. The steps are as follows:

- **For each $v_i \in S$, add an arc (v_i, s) with capacity $k(v_i)$ and transit time $T + 1$.**
- **Solve LexMCC to find the set $\mathcal{P}_{v_i}$ of all s-v_i paths for temporally repeated flows.**

To ensure optimality at all relevant vertices, we extend $\mathcal{P}_{v_i}$ to an enhanced path set E_{v_i}, as done in the DT-LexMDF problem. Since $k(v_i)$ is assumed to be sufficient, the free time interval I_1 does not exist, which simplifies the LexMCC computation.

This extended path set E_{v_i} induces an optimal dynamic flow for each terminal $v_i \in S$. The complete solution procedure is formally described in Algorithm

Note

This section introduces the DT-LexEAF problem — Discrete-Time Lexicographic Earliest Arrival Flow — which plays a key role in evacuation planning.

In general, earliest arrival flows may not exist in multi-sink networks. However, for UPL-TTSP networks with prioritized terminals, a structured and efficient solution is achievable.

The proposed approach builds upon the LexMDF method but emphasizes achieving the earliest possible arrival at each time step, respecting terminal priorities.

Key distinctions in this approach include:

- **Use of the MCC Algorithm to solve LexMCC with added reverse arcs.**
- **Extraction of temporally repeated s-v_i paths for each terminal.**
- **Construction of extended path sets E_{v_i} to ensure dynamic optimality.**
- **Simplified analysis when $k(v_i)$ is sufficient due to the absence of the free interval I_1.**

Journal of Advanced College of Engineering and Management

This strategy guarantees that the flow plan is not only maximum over time but also lexicographically optimal with respect to the earliest arrival at each prioritized terminal.

Result: The DT-LexEAF problem on the UPL-TTSP network N can be solved in polynomial time.

4.3 Lexicographic Quickest Flow Problem on UPL Network

Consider a network $N = (V, E)$ with fixed vertex holding capacities. Each vertex $v \in V$ is associated with a capacity $\phi(v)$, which serves both as an upper and lower bound on the amount of flow that must be held at that vertex. These capacities can be interpreted as demands that must be satisfied at each vertex $v \in V \setminus \{s\}$, where s is the designated source node.

This formulation leads to a dynamic flow problem on the network N, where the objective is to determine the minimum time required to satisfy all vertex demands while respecting a given priority order among the vertices. The flow must adhere to edge capacities and transit times, and the demands at the vertices must be fulfilled in the specified order of priority.

A natural application of this problem arises in evacuation planning. Suppose the number of evacuees at the source node is known in advance. The goal is to route these evacuees to various safety locations, each with a fixed holding capacity, in such a way that the total evacuation is completed in the shortest possible time. Moreover, the safety locations are prioritized, meaning that higher-priority locations must be filled before lower-priority ones are considered. This lexicographic objective ensures that the most critical demands are met as early as possible, aligning with real-world emergency response strategies.

Let $N = (G, u(a), \phi(v))$ be a network, where:

- **$G = (V, A)$ is a directed graph,**
- **$u(a) > 0$ is the capacity of arc $a \in A$,**
- **$\phi(v) \in \mathbb{N}_0$ denotes the demand at vertex $v \in V$.**

The terminal set is defined as:

$$S := \{v_1, v_2, \ldots, v_k\},$$

where the vertices are prioritized from highest to lowest:

$$v_1 \succ v_2 \succ \cdots \succ v_k.$$

The source vertex $s \in V$ has a negative demand:

$$\phi(s) = -\sum_{i=1}^{k} \phi(v_i),$$

representing the total supply.

We assume that the source s is a mother vertex, i.e., there exists a path from s to every terminal $v_i \in S$.

The Lexicographic Quickest Flow (LexQF) problem seeks a feasible dynamic flow f_{v_i} of value $\phi(v_i)$ from s to each terminal $v_i \in S$, such that the flow to each terminal is completed in the minimum number of time units $T(\phi(v_i))$, while respecting the given priority order.

At the completion time $T(\phi(v_i))$, the flow arriving at each terminal must satisfy:

$$\mathbf{ex}_f(v_i, T(\phi(v_i))) = \phi(v_i), \quad \forall v_i \in S.$$

The objective of the LexQF problem is to lexicographically minimize the vector of completion times:

$$(T(\phi(v_1)), T(\phi(v_2)), \ldots, T(\phi(v_k))).$$

Journal of Advanced College of Engineering and Management

This objective ensures that higher-priority demands are fulfilled as early as possible, aligning well with applications such as prioritized evacuation, logistics, and critical resource allocation.

4.3.1 Solution Procedure for the Lexicographic Quickest Flow Problem on UPL Network

We discuss the solution procedure for the Lexicographic Quickest Flow (LexQF) problem on a UPL network N. The approach is inspired by the binary search method used for solving the classical quickest flow problem.

Let T_n be a strictly increasing sequence of integral time points. We begin with an initial interval $I_0 = [T_l, T_u]$ such that:

$$f(T_l) < \phi(v_i) < f(T_u),$$

where $f(T)$ denotes the maximum dynamic flow value achievable within time horizon T. Clearly, the true completion time lies in this range:

$$T_l \leq T(\phi(v_i)) \leq T_u,$$

where $T(\phi(v_i))$ is the minimum time required to send $\phi(v_i)$ units of flow from the source s to vertex v_i.

At each iteration, the midpoint T_m of the current interval is computed, and the value $f(T_m)$ is evaluated. The logic is as follows:

- **If $f(T_m) = \phi(v_i)$, then T_m is a candidate solution.**
- **If $f(T_m) < \phi(v_i)$, the search continues in the right half of the interval.**
- **If $f(T_m) > \phi(v_i)$, the search proceeds in the left half.**

This iterative process continues until the minimum feasible time is identified.

Since we aim to find the minimum T_m for each vertex $v_i \in S$ in priority order, the maximum flow computation technique is used as a subroutine with necessary modifications.

A critical step involves constructing the extended set of paths E_{v_i}. For each path $v_{i-1} \rightarrow v_i$, we must compute the free time intervals I_1 and I_2, if they exist, for each newly selected midpoint time T_m before updating the network N_{v_k}.

The following cases arise based on T_m:

- **If $F_t(v_{i-1}) < T_m$, update $T \leftarrow T_m$ in I_2.**
- **If $F_t(v_{i-1}) \leq T_m < F_t(v_{i-1}) + 1$, discard I_2.**
- **If $I_t(v_{i-1}) < T_m < F_t(v_{i-1}) + 1$, discard I_2 and update $F_t(v_{i-1}) \leftarrow T_m$ in I_1.**
- **If $I_t(v_{i-1}) > T_m$, discard both I_1 and I_2.**

Vertices v_j for all $j > i$ are excluded until a feasible flow of value $\phi(v_i)$ is found for v_i within time horizon T_m, and T_m is proven to be minimal. This procedure is repeated for each vertex $v_i \in S$ according to the given priority.

Due to the nature of the maximum flow construction it is possible that a flow of value $\phi(v_i)$ obtained for time horizon T_m could also be achieved in a shorter time. Therefore, it cannot be guaranteed that T_m is minimal. One must verify whether the same flow value can be achieved for a smaller time horizon. This aligns with a lemma proposed in [7] for the s-d quickest flow problem, which holds true in our case as well.

4.3.2 Contraflow in Network Flow

Definition: Contraflow in network flow refers to the concept where the direction of flow on certain arcs within a network can be reversed to improve overall system performance. Unlike traditional flow models that operate under fixed arc directions, contraflow models offer flexibility by allowing selected arcs to carry flow in the opposite direction when needed.

Journal of Advanced College of Engineering and Management

This technique is particularly applicable in emergency scenarios, such as large-scale evacuations, where traffic lanes are reversed to maximize outbound capacity from a danger zone. Similarly, it finds use in traffic management and disaster logistics, enabling faster movement of people or resources.

The key idea behind contraflow is to determine, often through optimization, which arcs should be reversed in order to maximize flow or minimize travel time. The decision to reverse an arc typically involves binary decision variables that ensure only one direction is active at a time.

For example, for an arc (i, j) with capacity c_{ij}, we can introduce a binary variable r_{ij} such that:

$$x_{ij} \leq c_{ij}(1 - r_{ij}), \quad x_{ji} \leq c_{ij} r_{ij},$$

ensuring that flow is permitted in only one direction based on whether $r_{ij} = 0$ or $r_{ij} = 1$.

Contraflow models are inherently more complex than standard flow models due to the inclusion of arc reversal decisions, but they are powerful tools in optimizing dynamic and real-world systems. Applications include:

- **Evacuation planning,**
- **Urban traffic control,**
- **Emergency supply chain management.**

4.3.3 Lexicographic Maximum Static Contraflow Problem (LexMSCF)

Consider a static network $N = (G, l(a), u(a), k(v), s, d)$ where the terminal set $S \subseteq V$ is given by

$$S := \{v_1,\ v_2, \ldots, v_k\}, \quad \textbf{with priority} \quad d = v_1 > .... > v_k.$$

The function $k(v)$ denotes the vertex capacity at each terminal $v \in S$. In this setup, we aim to solve the lexicographic maximum static flow (LexMSF) problem with capacitated vertices, allowing the direction of arcs in the network N to be reversed. This variant is known as the Lexicographic Maximum Static Contraflow (LexMSCF) problem.

To solve the LexMSCF problem, we modify the existing solution approach used for solving the Maximum Static Contraflow (MSCF) problem as outlined in above. The key modification lies of that algorithm, where the standard maximum flow computation is replaced by the LexMSF computation method defined in Theorem 4.3.3.1. This replacement is necessary to ensure that intermediate holding capacities are respected and that vertex capacities are not violated.

The final solution procedure for the LexMSCF problem is formally described in Theorem 4.3.3.1.

Theorem 4.3.3.1. Given a static network $N = (V, A, l(a), u(a), k(v), s, d)$, a source node s, and a terminal set

$$S = \{v_1, v_2, \ldots, v_k\} \subseteq V, \quad \textbf{with} \quad d = v_1 > v_2 \cdots > v_k,$$

and assuming $l(a) = 0$ for all $a \in A$, computes a lexicographic maximum static contraflow on N optimally in strongly polynomial time.

Proof (Sketch). The LexMSF algorithm computes an optimal static flow for each terminal $v_i \in S$, treating each as a sink in the reduced network N, as guaranteed by Theorem 4.3.3.1. According to above theorythe resulting flows are equivalent to the maximum static contraflows on the original network N. Hence, the modified algorithm remains correct and optimal. The time complexity of Algorithm 4.3.3.1 is primarily determined by the LexMSF problem solved on the reduced network and remains strongly polynomial.

Journal of Advanced College of Engineering and Management

4.3.4 DT-LexMDCF Algorithm for Multi-network

We consider a dynamic multi-network

$$N = (V, A, l(a), u(a), \theta(a), k(v), s, d, T),$$

with terminal set $S = \{v_1, \ldots, v_k\}$ such that $d = v_1$. Each arc $a \in A$ has zero transit time, i.e., $l(a) = 0$, and all parameters are integer-valued.

To begin, the network N is transformed into an undirected multi-network as per Algorithm ensuring $l(a) = 0$ for all arcs. Each set of parallel arcs (v, w) is labeled distinctly as $(v, w)_i$, with increasing order:

$$(v, w)_i < (v, w)_{i+1} \quad \textbf{for all } i < m,$$

where $q < m$.

With the transformed network structure, we compute the LexMDF solution using Algorithm to obtain a lexicographically optimal dynamic flow over time horizon T. From this solution, a flow decomposition is carried out into path and cycle flows. All cycle flows are removed to isolate valid path-based flow assignments.

Reversibility of arcs is determined next: for an arc $(v, w) \in A$, the reverse arc (w, v) is introduced only if the flow along (v, w) exceeds its capacity $u(v, w)$, or if there exists non-negative flow along arc $a \in A$.

Finally, the discrete time lexicographic maximum dynamic contraflow (DT-LexMDCF) solution for the multi-network N is realized, satisfying the required terminal priorities, capacity constraints, and temporal consistency across the network.

5 Conclusion

After careful consideration, the solution process created for discrete time setting problems can also be used to continuous time setting problems. Therefore, using time-expanded networks for general networks with integer inputs and temporally-repeated networks for UPL networks can solve the LexMDF problem with continuous timesetting.

flows. While the latter problem can be resolved in strongly polynomial time, the former can only be resolved in pseudopolynomial time. For the UPL-TTSP network, the LexEAF problem with continuous timesetting can be similarly formulated and solved in strongly polynomial time. The DT-LexMDCF problem and DT-LexEACF problem can be extended into one with continuous timesetting because of the same rationale.

References

[1] L. R. Ford Jr and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[2] L. R. Ford Jr and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations research*, vol. 6, no. 3, pp. 419–433, 1958.

[3] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

[4] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," in *Soviet Math. Doklady*, vol. 11, pp. 1277–1280, 1970.

[5] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM (JACM)*, vol. 35, no. 4, pp. 921–940, 1988.

[6] L. Ford Jr and D. Fulkerson, "Flows in networks, princeton universitypress, princeton, nj, 1962," *Ford-Flows in Networks1962*.

[7] R. E. Burkard, K. Dlaska, and B. Klinz, "The quickest flow problem," *Zeitschrift für Operations Research*, vol. 37, no. 1, pp. 31–58, 1993.

[8] L. Fleischer and M. Skutella, "Quickest flows over time," *SIAM journal on computing*, vol. 36, no. 6, pp. 1600–1630, 2007.

[9] S. Ruzika, H. Sperber, and M. Steiner, "Earliest arrival flows on series-parallel graphs," *Networks*, vol. 57, no. 2, pp. 169–173, 2011.

[10] T. N. Dhamala and U. Pyakurel, "Earliest arrival contraflow problem on series-parallel graphs," *International Journal of Operations Research*, vol. 10, no. 1, pp. 1–13, 2013.

[11] S. R. Khadka and P. P. Bhandari, "Lexicographic earliest arrival contraflow evacuation problem on upl-ttsp network," *International Journal of Operational Research/Nepal*, vol. 9, no. 1, pp. 1–7, 2020.

[12] T. N. Dhamala, "A survey on models and algorithms for discrete evacuation planning network problems.," *Journal of Industrial & Management Optimization*, vol. 11, no. 1, 2015.

[13] S. R. Khadka and P. P. Bhandari, "Dynamic network contraflow evacuation planning problem with continuous time approach," *International Journal of Operations Research*, vol. 14, no. 1, pp. 27–34, 2017.

[14] U. Pyakurel, T. N. Dhamala, and S. Dempe, "Efficient continuous contraflow algorithms for evacuation planning problems," *Annals of Operations Research*, vol. 254, pp. 335–364, 2017.

[15] N. Megiddo, "A good algorithm for lexicographically optimal flows in multi-terminal networks," 1977.

[16] B. Hoppe and É. Tardos, "The quickest transshipment problem," *Mathematics of Operations Research*, vol. 25, no. 1, pp. 36–62, 2000.

[17] P. P. Bhandari and S. R. Khadka, "Lexicographically maximum contraflow problem with vertex capacities," *International Journal of Mathematics and Mathematical Sciences*, vol. 2021, no. 1, p. 6651135, 2021.

[18] P. P. Bhandari, S. R. Khadka, S. Ruzika, and L. E. Schäfer, "Lexicographically maximum dynamic flow with vertex capacities," *Journal of Mathematics and Statistics*, vol. 16, no. 1, pp. 142–147, 2020.

[19] P. P. Bhandari and S. R. Khadka, "Evacuation planning problems with intermediate storage," in *AIJR Proceedings of International Conference on Applied Mathematics & Computational Sciences (ICAMCS-2019), Dehradun*, pp. 90–95, 2020.

[20] M. Schmidt and M. Skutella, "Earliest arrival flows in networks with multiple sinks," *Discrete Applied Mathematics*, vol. 164, pp. 320–327, 2014.

[21] N. Kamiyama, "Lexicographically optimal earliest arrival flows," *Networks*, vol. 75, no. 1, pp. 18–33, 2020.