Type of the Paper (Extended Abstract)

Flows on network with intermediate storage capability: evacuation planning perspective[†]

Phanindra Prasad Bhandari [1] and Shree Ram Khadka [2,*]

[1] Department of Science and Humanities, Khwopa Engineering College; phanindra.maths@gmail.com
[2] Central Department of Mathematics; shreeramkhadka@gmail.com
* Correspondence: shreeramkhadka@gmail.com
† Presented at the title, place, and date.

Abstract: Optimization models for evacuation with capability of holding evacuees at intermediate places are of particular interest when all the evacuees cannot be sent to the safe destination. We study the maximum flow evacuation planning problem that aims to lexicographically maximize the evacuees entering a set of capacitated terminals with respect to a given prioritization. We propose a polynomial time algorithm for the problem modeled on uniform path length (UPL) network. We also extend the solution idea to solve quickest flow evacuation planning problem that lexicographically minimizes the time required to fulfill the demand of evacuees at such terminals. Moreover, we consider an earliest arrival version of the problem with sufficient vertex capacities, and propose a polynomial time algorithm for uniform path length two terminal series parallel (UPL-TTSP) network.

Keywords: TTSP network; Uniform path length network; Lexicographically maximum flows; Evacuation planning problem

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Proceedings* **2021**, 68, x. https://doi.org/10.3390/xxxxx

Published: date

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Evacuation planning problems modeled with flow conservation at intermediate vertices allow evacuees to leave the source only if they reach the sink. However, in many evacuation scenarios, it would be crucial to send as many evacuees as possible to intermediate shelters, despite the safety, where evacuees can be provided medical aids or other necessary supports. The shelters might be prioritized, and constrained to some capacities which restrict the number of evacuees that can be held at. The prioritization depends upon evacuation scenario: facilities, distance from source, holding capacities, etc.

The dynamic version of maximum flow evacuation planning problem attempts to send a maximum number of evacuees from risk zone (source) to the safe destination (sink) within given time horizon [12]. Many dynamic network flow problems have been investigated in the context of evacuation planning problems since then, see [8, 9, 11, 15, 16, 21, 26, 27]. Problem closely related to a maximum dynamic flow problem is quickest flow problem that sends a given units of flow from the source to the sink in minimum possible time. For models and solutions, see [10, 17, 18, 22]. Problem that attempts to send a maximum number of evacuees from the source to the sink as earliest as possible within given time horizon is earliest arrival flow problem [1, 13, 17, 22, 28, 30].

Authors in [3-8, 21] studied the maximum flow evacuation planning problem modeled with relaxed flow conservation constraint that allows evacuees to be held at temporary shelters at intermediate vertices. Lexicographic maximum flow problem with multiple sources and multiple sinks of given priorities and sufficient sink capacities has been studied as an extension of maximum flow problem and showed that this problem can be solved in polynomial time in [23, 24, 25]. Authors in [17, 18] studied lexicographic maximum dynamic flows and developed a polynomial time algorithm based on temporally repeated flows. The problem that computes a feasible dynamic flow maximizing the amount of flow entering a set of terminals (sink and specified intermediate vertices) lexicographically with respect to a given prioritization and given vertex capacities has been considered in [8] (see also [7, 21]). The authors proposed a polynomial time algorithm for the static version of the problem and a pseudo-polynomial time algorithm for the dynamic case. They also showed that the dynamic version of the problem can be solved polynomially, if vertex capacities are sufficient. The earliest arrival flow problem in network with multiple sinks has been studied in [29] where all arc transit time are zero. For this setting, they have given a complete characterization of the class of networks that always allow for earliest arrival flows. An earliest arrival flow problem, maximizing the ratios of flow values to capacities on the sinks lexicographically instead of strictly obeying the capacity constraints on them, has been studied in [19]. A pseudo-polynomial and a polynomial time algorithms, for solving the problem with arbitrary and zero transit time for every arc, respectively, have also been proposed.

We consider lexicographically maximum dynamic flow (LexMDF) problem of [8] in Section 2 and propose an efficient solution procedure for UPL network in Section 3. We extend the solution to solve lexicographically earliest arrival flow problem with sufficient vertex capacities for UPL-TTSP network in Section 4. Moreover, we consider lexicographically quickest flow problem modeled on UPL network and solve it by using the solution procedure proposed for LexMDF problem in Section 5. Preliminary works on these problems are also appeared in [6]. Section 6 concludes the paper with future research direction.

2. Model discussion

We consider a directed graph $G = (V, A)$ without containing parallel arcs and loops to define evacuation planning problem. Here, V with $n := |V|$ and A with $m := |A|$ denote the vertex set and arc set, respectively, which are assumed to be finite. Vertices and arcs, in our case, represent the intersections of routes and the route segments joining these intersections, respectively. Two specified vertices s and d denote the source and the sink, respectively. We assume a terminal set $S \subset V$ with $S := \{v_1, \ldots, v_k\}$ prioritized from higher to lower priority, i.e., $v_1 \succcurlyeq \cdots \succcurlyeq v_k$, to be given, where $v_1 = d$. The arc capacity function $u: A \to N_0 := N \cup \{0\}$ bounds the number of flow units on each arc at each time step from above. Similarly, the vertex capacity function $k: S \to N_0$ bounds the total number of flow units, which may be held in each vertices $v \in S$. We set $k(d) = \infty$ and $k(v)$ to be finite for all $v \in S - \{d\}$. Further, the transit time function $\tau: A \to N$ specifies the time needed by a flow unit to traverse an arc. We assume a time horizon $T \in N$ to be given and treat time parameter in a discrete manner, i.e., $T := \{0, 1, \ldots, T\}$. With these setup for graph$G$, i.e., for evacuation network $N = (G, u, k, \tau, s, d, T)$, we give the network flow model for the evacuation planning problem in the following.

The non-negative flow variables $f(a, t)$, evacuees on the road segment at time t, defined by $f: A \times T \to N_0$ that specify the flow over time in the network N is the number of flow units entering arc a at time step $t \in T$. The number of flow units entering arc a at time step t are assumed to be bounded by the capacity of an arc, i.e., $f(a, t)$ satisfies the capacity constraints $0 \leq f(a, t) \leq u(a)$ for all $a \in A$ and for all $t \in T$. Moreover, $f(a, t)$ has to be equal to zero for all $t > T - \tau_a$and for all $a \in A$. The excess flow at vertex $v \in V$ at time $t \in T$ is defined as

$$0 \leq ex_f(v, t) := \sum_{a \in \delta^-(v)} \sum_{\xi=0}^{t-\tau_a} f(a, \xi) - \sum_{a \in \delta^+(v)} \sum_{\xi=0}^{t} f(a, \xi). \tag{1}$$

Further, we need to ensure that

$$ex_f(v, T) \leq k(v) \text{ for all } v \in S. \tag{2}$$

Consequently, the total flow of evacuees leaving the source s equals the total flow of the evacuees held at vertices $v \in S$ over the time horizon T, i.e.,

$$\sum_{a\in\delta^{+}(s)}\sum_{\xi=0}^{T} f(a,\xi) - \sum_{a\in\delta^{-}(s)}\sum_{\xi=0}^{T} f(a,\xi) = \sum_{v\in S} ex_f(v,T). \quad (3)$$

The objective function of the maximum flow evacuation planning problem asks to lexicographically maximize the vector $(ex_f(v_1,T),..,ex_f(v_1,T))^T$ such that $ex_f(v_i,T) \le ex_f(v_i,T)$ for $i = 1,\dots,k$. We call the flow problem on network N with this objective as lexicographic maximum dynamic flow (LexMDF) problem. Dynamic flow problem that aims to fulfill the objective of LexMDF problem at each time $t \in T$ is lexicographic earliest arrival flow (LexEAF) problem. For given prioritized set S of vertices with fixed demand at each $v \in S$, the dynamic flow problem that aims to lexicographically minimize the time required to fulfill these demands is lexicographic quickest flow (LexQF) problem (cf. Section 5).

3. Solution to LexMDF problem on UPL network

Here, the goal is to solve the LexMDF problem on N in polynomial time using temporally repeated flows (TRFs). For general network, flow computed by TRFs for some vertices $v_i \in S$ as the sink may exceed their capacities or may not induce optimal flows for these vertices due to non-uniqueness of path decomposition [8]. An ordinary TRF does not yield a maximum flow even for two terminal series parallel network [4]. These hurdles occur due to fixed vertex capacities at vertices. We fix this hurdle for the problem on a uniform path length (UPL) network N. A two terminal network N with source vertex s is a UPL network if, for any vertex $v \in N$, all possible directed $s - v$ path on N have equal distances. We consider the distance of the path with respect to its transit time. That is, a network N is a uniform path length network for which the sum of the transit times on arcs on any possible path from the source s to any vertex $v \in N$ are equal, see Figure 1.

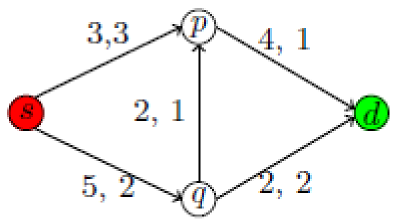

Figure 1. A uniform path length (UPL) network N with source vertex s.

The main idea of the solution procedure of the LexMDF problem on N is to find all possible $s - v_i$ paths at all possible time steps $t \in T$ with corresponding flow value and send as many units of flow as possible along paths as long as possible. Such paths can be found by decomposing the flow computed by solving Lexicographic Minimum Cost Circulation (LexMCC) problem on N, iteratively.

Any minimum cost circulation algorithm can be applied to solve LexMCC problem on N repeatedly for each $v_i \in S$ as a sink in given priority order on corresponding residual network of N with additional arc (v_i, s) with capacity equal to $k(v_i)$ and transit time $-(T+1)$. Also, the transit time $\tau(a)$ for all $a \in A$ is switched into the cost $c(a)$. This yields a set $\Gamma_{v_i}$of all $s - v_i$ paths, denoted as γ_{v_i}, that could be temporally repeated from time step zero for each $v_i \in S$. It is noteworthy to mention that path γ_{v_i} is a chain of vertices and arcs in the network N starting at the source s and terminating at vertex v_i. To each path γ_{v_i} we associate the following information: (a) $f(\gamma_{v_i})$-- the flow value that can be sent along γ_{v_i} at once, (b) $\tau(\gamma_{v_i})$-- the time required to travel γ_{v_i} by a flow unit, (c) $I_t(\gamma_{v_i})$-- the time step at which the flow along γ_{v_i}starts to get repeated and (d) $F_t(\gamma_{v_i})$--

the time step after which the flow along γ_{v_i} stops to get repeated. The procedure for solving LexMCC problem is termed as LexMCC Algorithm, hereafter.

Lemma 3.1 Given a UPL network N with prioritized set of vertices $S \subset V$. Then LexMCC problem can be solved in $O(n \times MCF(n,m))$ times on N where $MCF(n,m)$ is the time complexity for a single MCF problem.
Proof. Lemma follows directly from the fact that $|S| < |V| = n$. ■

3.1. Construction of extended set $\Gamma_{v_i}^E$

Consider a UPL network $N = (G, u, k, \tau, s, d, T)$. Any temporally repeated flow on N generated by Γ_{v_i}, the set of $s - v_i$ paths obtained by applying LexMCC algorithm, has limitation. The limitation is that there may exist $s - v_i$ path, say γ_{v_i} such that $\gamma_{v_i} \notin \Gamma_{v_i}$ for $v_i \in S$, on the residual network of N for an interval of time with transit time $\tau(\gamma_{v_i}) < T + 1 - I_t(\gamma_{v_i})$ along which some flow units could be sent at v_i. This situation occurs when any path $\gamma_{v_j} \in \bigcup_{j=1}^{i-1} \Gamma_{v_j}$ is free to carry flow units at v_i at time $I_t(\gamma_{v_i}) > 0$, due to time limit or capacity at vertex v_j for some $j < i$, before the time $T + 1 - \tau(\gamma_{v_i})$. In this situation, $I_t(\gamma_{v_i}) = F_t\left(\gamma_{v_j}\right) + 1 - N(\gamma_{v_j})$ where $N(\gamma_{v_j})$ is the actual number of times that the flow along γ_{v_j} is repeated. The number of actual repetitions $N(\gamma_{v_i})$ along any path γ_{v_i} depends upon vertex capacity $k(v_i)$ and is given by the Path Flows Repetition (PFR) technique (cf. Subsection 3.2). Thus, applying lexMCC Algorithm at time zero only may not be enough for the optimal solution at all possible vertices using the TRF approach. Thus, it is required to find an extended set $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ that contains all possible $s - v_i$ paths, say γ_{v_i}, which could be started to repeat at time $I_t(\gamma_{v_i}) \geq 0$.

An extended set of paths $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ is given by

$$\Gamma_{\mathrm{v_i}}^{\mathrm{E}} := \begin{cases} \Gamma_{v_i}\, for\ i = 1 \\ \Gamma_{v_i} \cup \Gamma'_{\mathrm{v_i}}\, for\ i > 1 \end{cases}$$

where $\Gamma'_{\mathrm{v_i}}$ is the set of all $s - v_i$ paths that are free to carry flow units at v_i at time intervals $I_1(\gamma_{v_{i-1}}) = [I_t(\gamma_{v_{i-1}}), F_t(\gamma_{v_{i-1}}) - N(\gamma_{v_{i-1}})]$ and $I_2(\gamma_{v_{i-1}}) = [F_t(\gamma_{v_{i-1}}) + 1, T]$ with respect to each path $\gamma_{v_{i-1}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$. Union of these two intervals are the complement of the interval of time period in which the path $\gamma_{v_{i-1}}$ is engaged in sending flow units at vertex v_{i-1}, given by $[F_t(\gamma_{v_{i-1}}) + 1 - N(\gamma_{v_{i-1}}), F_t(\gamma_{v_{i-1}})]$, on the time interval $[I_t(\gamma_{v_{i-1}}), T]$. First interval $I_1(\gamma_{v_{i-1}})$ is discarded if $F_t(\gamma_{v_{i-1}}) - N(\gamma_{v_{i-1}}) < I_t(\gamma_{v_{i-1}})$, and second interval $I_2(\gamma_{v_{i-1}})$ is discarded if its own immediate parent interval is $I_1(\gamma_{v_{i-2}})$. If no interval is discarded, they are merged in a single interval $[I_t(\gamma_{v_{i-1}}), T]$ if $N(\gamma_{v_{i-1}}) = 0$, and taken as two different intervals if $N(\gamma_{v_{i-1}}) > 0$. It is to be noted that $I_1(\gamma_{v_1}) = \emptyset$ for all $\gamma_{v_1} \in \Gamma_{v_1}$.

Residual network of N after solving LexMCC problem on it, say $N_{\Gamma_{v_k}}$, is renovated with respect to the path $\gamma_{v_{i-1}}$ for corresponding free time intervals I_1 and I_2, separately. Then LexMCC Algorithm is applied on it to find the set $\Gamma'_{\mathrm{v_i}}$. During renovation, the capacity of each arc $a \in N_{\Gamma_{v_k}}$ is increased by $f(\gamma_{v_{i-1}})$ if the arc $a = (v, w)$ also belongs to path $\gamma_{v_{i-1}}$; and the capacity of the arc $(w, v) \in N_{\Gamma_{v_k}}$ is decreased by the same value $f(\gamma_{v_{i-1}})$. That is,

$$u(a) := \begin{cases} u(a) + f\left(\gamma_{v_{i-1}}\right) for\ a = (v,w) \in N_{\Gamma_{v_k}} such\ that\ a \in \gamma_{v_{i-1}} \\ u(a) - f\left(\gamma_{v_{i-1}}\right) for\ a = (w,v) \in N_{\Gamma_{v_k}} such\ that\ a = (v,w) \in \gamma_{v_{i-1}} \end{cases}$$

The LexMCC Algorithm is applied only after renovation of residual network $N_{\Gamma_{v_k}}$ with respect to all paths $\gamma_{v_{i-1}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$ that are free at the same interval of time. This significantly reduces computational complexity (application of MCC Algorithm) of the entire algorithm. This is not possible if N is not a UPL network. While choosing the second, third and so on paths for renovation, the network which is renovated with respect to first, second and so on paths, respectively, is renovated. This process is repeated for all $\gamma_{v_{i-1}} \in \Gamma^{\mathrm{E}}_{\mathrm{v_{i-1}}}$.

3.2. Path flows repetition (PFR) technique

To compute a lexicographic maximum dynamic flow on N, it is required to repeat path flows in $\Gamma^{\mathrm{E}}_{\mathrm{v_i}}$ respecting capacities $k(\mathrm{v_i})$ for each $\mathrm{v_i} \in S$. For this we propose the following Path Flows Repetition (PFR) Technique.

Paths $\gamma_{v_i} \in \Gamma^{\mathrm{E}}_{\mathrm{v_i}}$ are indexed as $\gamma^p_{v_i}$; $p = 1,2,\dots,l$, in such a way that the path with highest final time, $F_t\,(\gamma^p_{v_i})$ among the paths $\gamma_{v_i} \in \Gamma^{\mathrm{E}}_{\mathrm{v_i}}$ with highest initial time, $I_t\,(\gamma^p_{v_i})$ gets the least vertex exponent p and so on. If two paths $\gamma_{v_i'}$ and $\gamma_{v_i''}$ have same final time, choice of path depends on the priority vertex the paths pass through. For example, if path $\gamma_{v_i'}$ passes through the vertex v_{i-1} and the path $\gamma_{v_i''}$ passes through the vertex v_{i-2} whiling reaching at vertex v_i, we choose the path $\gamma_{v_i''}$. Tie after this can be broken arbitrarily.

The computation of TRF $f(\gamma^p_{v_i}) := \sum_{q=1}^{p}(T + 1 - I_t\left(\gamma^q_{v_i}\right) - \tau(\gamma^q_{v_i}))f(\gamma^q_{v_i})$ for vertex v_i starts with $p = 1$. If $f(\gamma^p_{v_i}) = k(v_i)$, $f(\gamma^p_{v_i})$ is a maximum flow for v_i. If $f(\gamma^p_{v_i}) < k(v_i)$, $f(\gamma^{p+1}_{v_i})$ is computed if $p + 1 \leq l$, otherwise $f(\gamma^p_{v_i})$ is maximum. If $k(v_i) < f(\gamma^p_{v_i})$, $k(v_i)$ is maximum flow for $p = 1$; and $f(\gamma^{p-1}_{v_i}) + k_r(v_i)$ is maximum for $p > 1$ at the vertex v_i. The TRF is likely to get flow repeated more than once over the time horizon T. If flow repetition occurs more than once along the path $\gamma^p_{v_i}$ over T, the time interval $T' = \left[I_t\left(\gamma^p_{v_i}\right) + \tau\left(I_t\left(\gamma^p_{v_i}\right)\right), T\right]$ is halved and the TRF is computed in the second half. The computed flow is then added to $f(\gamma^{p-1}_{v_i})$. The total flow is compared to the vertex capacity. Flow in the first half is also computed and then added if the total flow is less than the vertex capacity. If the total flow exceeds the vertex capacity, the added flow is discarded. Then the second half is further halved and the procedure is repeated. Integral time units of time horizon T is preserved by rounding up or down to the nearest integer during halving the interval. The procedure is executed if the total flow equals the vertex capacity or if $l < p$.

A flow with value more than the residual vertex capacity $k_r(v_i)$ may occur along the path $\gamma^p_{v_i}$ while sending even at once at the vertex v_i. In this situation, the set $\Gamma^{\mathrm{E}}_{\mathrm{v_i}}$ is updated by splitting $\gamma^p_{v_i}$ into $\gamma^{p'}_{v_i}$ and $\gamma^{p''}_{v_i}$ with flow values $k_r(v_i)$and $f(\gamma^p_{v_i}) - k_r(v_i)$, respectively.

Algorithm 1 summarizes the procedure that yields the maximum flow on network N at each of the possible vertices in given priority order.

Algorithm 1: DT-LexMDF Algorithm for UPL Network

```
1. Given a dynamic UPL network $N = (G, u, k, \tau, s, d, T)$, $S := \{v_1, \dots, v_k\}$ with $d = v_1 \succcurlyeq \cdots \succcurlyeq v_k$.
2. Find $\Gamma_{v_i}$ for all $i = 1, 2, \dots, k$ by solving the LexMCC problem on $N$ with additional arcs $(v_i, s)$ with capacity $k(v_i)$ and transit times $-(T + 1)$.
3. For $i = 1$, set $\Gamma^{\mathrm{E}}_{\mathrm{v_i}} := \Gamma_{v_i}$ and apply PFR technique on $\Gamma^{\mathrm{E}}_{\mathrm{v_i}}$. For $i > 1$, go to step 4.
4. For each path $\gamma_{v_{i-1}} \in \Gamma^{\mathrm{E}}_{\mathrm{v_{i-1}}}$, find the interval $[F_t(\gamma_{v_{i-1}}) + 1 - N(\gamma_{v_{i-1}}), F_t(\gamma_{v_{i-1}})$ and intervals $I_1 = [I_t(\gamma_{v_{i-1}}), F_t(\gamma_{v_{i-1}}) - N(\gamma_{v_{i-1}})]$ and $I_2 = [F_t(\gamma_{v_{i-1}}) + 1, T]$.
5. Renovate the network $N_{\Gamma_{v_k}}$ with respect to path $\gamma_{v_{i-1}}$ for intervals $I_1$ and $I_2$.
6. Find $\Gamma'_{\mathrm{v_i}}$ for all $i = 2, \dots, k$ by solving the LexMCC problem on renovated $N_{\Gamma_{v_k}}$ as initial time $I_t(\gamma_{v_i})$, with additional arcs $(v_i, s)$ with capacity $k(v_i)$ and transit time $-(T + 1)$.
7. Set $\Gamma^{\mathrm{E}}_{\mathrm{v_i}} := \Gamma_{v_i}$ and update $\Gamma^{\mathrm{E}}_{\mathrm{v_i}} := \Gamma^{\mathrm{E}}_{\mathrm{v_i}} \cup \Gamma'_{\mathrm{v_i}}$ for all $i = 2, \dots, k$.
8. Apply PFR technique on $\Gamma^{\mathrm{E}}_{\mathrm{v_i}}$.
9. Obtain dynamic $s - v_i$ flow on $N$.
```

Lemma 3.2 In Algorithm 1, LexMCC problem is solved at most $2n$ times for each $v_i \in S$.

Proof. At first we prove that the application of PFR technique on $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$, for each $v_i \in S$, creates at most 2 new free time intervals for next prioritized vertex v_{i+1}. For, let TRF $f(\gamma_{v_i}^{p})$ be any optimal flow for v_i on N obtained by the application of PFR on $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$. Also, let $\gamma_{v_i}^{p}$ is the path that exists in the interval $[I_t(\gamma_{v_i}), F_t(\gamma_{v_i})]$. Here, if TRF $f(\gamma_{v_i}^{p})$ is obtained when all the paths that exist to carry flow at v_i are repeated temporally for all time steps in the interval, no new free time interval for v_{i+1} is formed. If all of such path are repeated for equal number of times less than the maximum possible time steps in the interval, only one new free time interval is formed. And, if any one of such paths needs to be split in to two paths and repeated one of them for some less or more number of times than other, one extra new free time interval is formed for next prioritized vertex v_{i+1}.

For $i = 1$, the MCC Algorithm is applied only once and twice for $i = 2$; once with initial time as zero and next with initial time as $F_t(\gamma_{v_1})$, being sufficient capacity at $v_1 = d$. However, for $i > 2$, the number of times for the application of the algorithm is increased by at most 2 in each of at most n iterations, due to above argument and since it is applied only after the renovation of $N_{\Gamma_{v_k}}$ with respect to all paths $\gamma_{v_{i-1}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$ that are free at the same interval of time. Therefore, to compute the extended set $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$, LexMCC problem is solved at most $2n$ times for each $v_i \in S$. ■

Lemma 3.3. Renovation of the residual network $N_{\Gamma_{v_k}}$ is well defined for each iteration.

Proof. The residual network $N_{\Gamma_{v_k}}$ is well defined by its definition. It is renovated with respect to each path $\gamma_{v_{i-1}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$ that exist in the same interval of time by taking any one of such paths at the first. While choosing the second, third and so on paths, the network which is renovated with respect to first, second and so on paths, respectively, is considered for renovation. During renovation with respect to path $\gamma_{v_{i-1}}$, the capacity of each arc $a = (v, w) \in N_{\Gamma_{v_k}}$ is increased by $f(\gamma_{v_{i-1}})$ if the arc a also belongs to $\gamma_{v_{i-1}}$, and the capacity of the arc $(w, v) \in N_{\Gamma_{v_k}}$ is decreased by the same value $f(\gamma_{v_{i-1}})$. The renovation of the network is done only for those interval of time which was never been used by the path $\gamma_{v_{i-1}}$ during temporal repetition. Thus, the renovation of the residual network $N_{\Gamma_{v_k}}$ is well defined for each iteration. ■

Lemma 3.4. For any vertex $v_i \in S$, the number of paths in extended set $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ is bounded above by $2nm$.

Proof. By Lemma 3.2, LexMCC problem is solved at most $2n$ times for each $v_i \in S$. And, at most m minimum cost flow paths from the source s to the vertex v_i do exist in each iteration. Therefore, the number of paths in $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$, for $v_i \in S$, does not exceed $2nm$. ■

Lemma 3.5. The residual network $N_{\Gamma_{v_k}}$ is renovated in time $O(nm)$ for each $v_i \in S$.

Proof. For each vertex $v_i \in S$, the residual network $N_{\Gamma_{v_k}}$ is renovated with respect to each path $\gamma_{\mathrm{v_{i-1}}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$, separately. By Lemma 3.4, there are at most $2nm$ paths in $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$. Therefore, the number of iterations for renovation of network $N_{\Gamma_{v_k}}$ for a vertex $v_i \in S$ is $2nm$. This concludes that the residual network $N_{\Gamma_{v_k}}$ can be renovated in time $O(nm)$ for each $v_i \in S$. ■

Lemma 3.6. The PFR technique executes in time $O(nm + logT)$.

Proof. The extended set $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ has at most $2nm$ paths, by Lemma 3.4. Therefore, TRF $f(\gamma_{v_i}^{p})$ is computed on $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ and compared to the vertex capacity $k(v_i)$ at most $2nm$ times. Additionally, while the computed TRF $f(\gamma_{v_i}^{p})$ exceeds the vertex capacity at v_i, the interval $T' = [I_t(\gamma_{v_i}^{p}), F_t(\gamma_{v_i}^{p})]$ is halved and the TRF is computed in one of the half intervals. This process needs repetition until the length of halved interval is unity in worst case. Therefore, this process takes $O(logT)$ time to execute. This concludes that the PFR technique executes in time $O(nm + logT)$. ■

Theorem 3.7. Given a UPL network $N = (G, u, k, \tau, s, d, T)$ and terminal set $S := \{v_1, \dots, v_k\} \subset V$ with $d = v_1 \succcurlyeq \cdots \succcurlyeq v_k$. Then, Algorithm 1 yields an optimal solution to the LexMDF problem on N.
Proof. As the vertex $v_1 (= d)$ has sufficient storage capacity, applying Path Flows Repetition technique for this vertex as the sink is equivalent to pushing the flow units with value $f(\gamma_{v_1})$ along each path $\gamma_{v_1} \in \Gamma_{v_1}$for each time step $t \in \{0, 1, \dots, T - \tau_{\gamma_{v_1}}\}$ from the source s to the sink d. This temporally repeated flow for sink d induces a dynamic $s - d$ flow which is feasible and optimal [12].

For each > 1, the extended set $\Gamma_{v_i}^E$ contains all minimum cost $s - v_i$ paths that exist at any time step $t \in \{0, 1, \dots, T\}$ on residual network of N with respect to the optimal flow $f(\gamma_{v_{i-1}}^p)$ at previous immediate prioritized vertex v_{i-1}. Thus, the TRF $f(\gamma_{v_i}^p)$ obtained by applying PFR technique on $\Gamma_{v_i}^E$ is feasible. The technique pushes flows of corresponding values along each path as long as possible unless $k(v_i)$ is satisfied. Moreover, the flow is pushed along the paths in $\Gamma_{v_i}^E$ with the strategy of saving unused paths in $\Gamma_{v_i}^E$ for the use of next less prioritized vertex v_{i+1} without violating the optimality at v_i. This is assured by selecting the path with highest final time, $F_t(\gamma_{v_i}^p)$ among the paths $\gamma_{v_i} \in \Gamma_{v_i}^E$ with highest initial time, $I_t(\gamma_{v_i}^p)$ at the first and so on. Thus, TRF $f(\gamma_{v_i}^p)$ is optimal on N for each $v_i \in S$. ■

Theorem 3.8. Algorithm1 runs in strongly polynomial time.
Proof. Due to Lemmas 3.1 and 3.2, the LexMCC problem can be solved in time $O(n \times MCF(n, m))$ for at most $2n$ times for each of at most n vertices $v_i \in S$. Due to Lemma 3.5, the residual network $N_{\Gamma_{v_k}}$ is renovated in time $O(nm)$ for each $v_i \in S$. The PFR technique can be performed in $O(nm + logT)$ time for each vertex $v_i \in S$, by Lemma 3.6. If one wishes to apply the MCF algorithm of [14], LexMCC Algorithm has complexity of order $(n^2 m^3 \, logn)$. Thus, Algorithm 1 runs in $O(n^3 \, (n^2 \, m^3 \, logn) + n(n\, m) + n(nm + logT))$. Equivalently, the algorithm has time complexity of order $O(n^5 m^3 \, logn + n^2 m + n \, logT)$ which is strongly polynomial. ■

4. Lexicographic earliest arrival flow problem on UPL-TTSP network

A DT-LexMDF problem that fulfills its objective at each time step $t \in T$ is a discrete time lexicographic earliest arrival flow (DT-LexEAF) problem. That is, the objective of a DT-LexEAF evacuation planning problem is to send a maximum number of evacuees at the possible earliest time from risk zone to the safety together with relatively safe prioritized intermediate spots within given time horizon T in given priority order.

It is clear that every earliest arrival flow is a maximum dynamic flow for given time horizon. However, the converse is not always true for general network. In this section, we purpose a solution procedure that computes a lexicographic maximum dynamic flow on a typical network and claim that this flow schedule has an earliest arrival property.

Consider a UPL-TTSP network $N = (G, u, k, \tau, T)$ with terminal set $S \subset V$ as in the case of LexMDF problem in Section 3. Here, the vertex v_1 always gets sufficient holding capacity whereas vertices v_i for $i > 1$, get either zero or sufficient capacities. That is, not all vertices in V have holding capacities on them. With these settings the LexEAF problem on N aims to maximize the flow units sent to the terminals in S in given priority order at each time step $t \in T$.

The solution strategy to DT-LexEAF problem on UPL-TTSP network N is similar to that of EAF problem on TTSP network given in [28]. The strategy is applied to solve the LexMCC problem on N repeatedly for each $v_i \in S$ with additional arc (v_i, s) with capacity equal to $k(v_i)$ and transit time $-(T + 1)$. This yields a set Γ_{v_i} of all $s - v_i$ paths that could be temporally repeated for each $v_i \in S$. However, dynamic flow generated by temporally repeated flow along the paths obtained by solving this problem may not be optimal on N at all possible vertices. This hurdle can be overcome by the construction of extended set of paths $\Gamma_{v_i}^{E}$ as in the case of DT-LexMDF problem in Section 3 for UPL network. Here, $k(v_i)$ being sufficient, there does not exist the free time interval I_1 which significantly reduces computational complexity of the LexMCC problem. The set $\Gamma_{v_i}^{E}$ induces an optimal dynamic flow for each v_i on N.

The exact solution procedure that yields the discrete time maximum dynamic flow at each vertices $v_i \in S$ is given in Algorithm 2.

Algorithm 2: DT-LexEAF Algorithm for UPL-TTSP Network

1. Given a UPL-TTSP network $N = (G, u, k, \tau, s, d, T)$, $S := \{v_1, \dots, v_k\}$ with $d = v_1 \succcurlyeq \cdots \succcurlyeq v_k$.
2. Solve LexMCC problem on N with additional arcs (v_i, s) with capacity $k(v_i)$ and transit times $-(T+1)$ using algorithm in [2].
3. Construct extended set $\Gamma_{v_i}^{E}$ as in Algorithm 1.
4. Push as much flow as possible along each path in $\Gamma_{v_i}^{E}$as long as possible within T.
5. Obtain dynamic $s - v_i$ flow on N.

Theorem 4.1. Algorithm 2 yields an optimal solution to DT-LexEAF problem on UPL-TTSP network $N = (G, u, k, \tau, s, d, T)$ in strongly polynomial time.
Proof. The algorithm pushes flow of value $f(\gamma_{v_i})$ along each path on $\Gamma_{v_i}^{E}$ for each possible time step $t \in \{0, 1, 2, \dots, T - \tau_{\gamma_{v_i}}\}$ from the source vertex s to each of the destination vertex $v_i \in S$ in given priority order. Therefore, a maximum flow at each $v_i \in S$ is obtained at the termination of algorithm, [12]. Moreover, the network N being a two terminal series parallel in structure, this flow has an earliest arrival property [28].

The extended set of paths $\Gamma_{v_i}^{E}$ in step 3 of algorithm is constructed by applying the MCC Algorithm in [2] with time complexity of order $O(mn + m\,logm)$ at most nm times for each vertex $v_i \in S$. Step 4 is executed in constant time for each of at most n vertices $v_i \in V$. Therefore, Algorithm 2 yields a lexicographic earliest arrival flow on N in strongly polynomial time. ■

5 Lexicographic quickest flow problem on UPL network

Consider the network N with fixed vertex holding capacities. Let us impose the condition for these capacities to be fulfilled as an upper bound as well as a lower bound by the total flow value that is supposed to be held at that vertex. Then vertex capacities can be taken as demands, say, $\mu(v)$ at v for $v \in V - \{s\}$. This consideration allows to see a dynamic flow problem on N with demands at vertices and asking for a minimum time to satisfy these demands in given priority order. In the following, we define this problem formally. The application of the problem on evacuation planning is obvious when evacuees at the source are known in advance and one wishes to send them to different prioritized safety places of fixed holding capacities.

Given a network $N = (G, u, \tau, \mu)$ with terminal set $S \subset V$ with $S := \{v_1, \dots, v_k\}$ prioritized from higher to lower priority, i.e., $d = v_1 \succcurlyeq \cdots \succcurlyeq v_k$; such that $\sum_i \mu(v_i) = -\mu(s)$ where $\mu : V \to N_0$ is the demand at the vertex $v \in V$. The negative demand at the source s is termed as supply. Moreover, we restrict the arc capacity $u(a)$ for each arc $a \in A$ to be strictly positive and consider the network N in such a way that the source vertex s is the *mother vertex* for all the vertices $v_i \in S$. Then the lexicographic quickest flow (LexQF) problem finds a feasible dynamic flow $f(v_i)$of given value $\mu(v_i)$ on network N from the source s to the vertex v_i, in given priority order, which sends the given $\mu(v_i)$units of flow from s to v_i in minimum number $\mu(v_i)$ of time units. Moreover, the excess $ex_f(v_i, T(\mu(v_i))$at each $v_i \in S$ given by equation 2.1 should be equal to the demand at $T(\mu(v_i))$. That is,

$$ex_f(v_i, T(\mu(v_i))) = \mu(v_i) \text{ for all } v_i \in S. \quad (4)$$

Thus, the objective function of the lexicographic quickest flow evacuation planning problem asks to lexicographically minimize the vector $(T(\mu(v_1)), T(\mu(v_2)), \ldots, T(\mu(v_k)))^T$.

5.1 Existence of lexicographic quickest flow

The existence of lexicographic quickest flow on a uniform path length network N is obvious, if $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ (cf. Section 3) is not empty for all $v_i \in S$. There always exists at least one path from the source s to the vertex v_i in the extended set of paths $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ with corresponding positive flow value since every vertex $v_i \in S$ being reachable from s and $u(a)$ being positive for each $a \in A$. This is ensured from the fact that during the construction of extended set $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$ the renovation of the network $N_{\Gamma_{v_k}}$ with respect to at least one path $\gamma_{v_{i-1}} \in \Gamma_{\mathrm{v_{i-1}}}^{\mathrm{E}}$ makes the renovated network free to exist at least one path from s to v_i. Thus, $\Gamma_{\mathrm{v_i}}^{\mathrm{E}} \neq \emptyset$ for each $v_i \in S$.

5.2 Solution for lexicographic quickest flow problem

Here we discuss the solution procedure for the lexicographic quickest flow problem for a UPL network N. The procedure to the problem is similar to the binary search method for solving a quickest flow problem in [10]. In this method, for an strictly increasing sequence of integral time points $\{T_n\}$, an initial interval $I_0 = [T_l, T_u]$ such that $f(T_l) < \mu(v_i) < f(T_u)$, is taken. Here, $f(T_l)$ and $f(T_u)$ denote the maximum dynamic flow value for time horizon T_l and T_u, respectively. Clearly, $T_l \leq T(\mu(v_i)) \leq T_u$ where $T(\mu(v_i))$ is the minimum time that requires for flow units of value $\mu(v_i)$ to send from the source to the vertex v_i. Then the mid-point, say, T_m, of the interval I_0 is computed and $f(T_m)$is checked for whether it is equal to, less than or greater than $\mu(v_i)$. Depending upon this value, it is decided whether the procedure ends, or should work on the next interval on the left or right of the mid-point T_m.

Since we are interested in finding such minimum time T_m for each vertices $v_i \in S$ in a priority order, the maximum flow computation technique developed in Section 3 is adopted as a subroutine of the procedure with necessary modification. During the procedure, the major step is to construct extended set of paths $\Gamma_{\mathrm{v_i}}^{\mathrm{E}}$. Here, the free time intervals I_1 and I_2 , if exist, with respect to each path $\gamma_{\mathrm{v_{i-1}}} \in \Gamma_{\mathrm{v_{i-1}}}$ are to be calculated in each of new selections of mid-point time T_m before renovation of the network $N_{\Gamma_{v_k}}$. Now, the following cases arises: If $F_t(\gamma_{\mathrm{v_{i-1}}}) < T_m$, replace T by $T_m$in I_2. If $F_t(\gamma_{\mathrm{v_{i-1}}}) - N < T_m < F_t(\gamma_{\mathrm{v_{i-1}}}) + 1$, discard I_2. If $I_t(\gamma_{\mathrm{v_{i-1}}}) < T_m < F_t(\gamma_{\mathrm{v_{i-1}}}) + 1 - N$ discard I_2 and replace $F_t(\gamma_{\mathrm{v_{i-1}}}) - N$ by T_m in I_1. And, if $I_t(\gamma_{\mathrm{v_{i-1}}}) > T_m$, discard both I_1 and I_2 . It is to be noted that Γ_{v_j} for all $j > i$ are discarded until we found Γ_{v_i} that sends all flow $\mu(v_i)$ in time horizon T_m such that it is minimum. We proceed with this procedure for each vertices $v_i \in S$ in given priority order.

Due to the nature of construction of a maximum flow (cf. Algorithm 1), maximum flow of value $\mu(v_i)$obtained for time horizon T_m, could also be possible to find in lesser time horizon T_m' for some vertices v_i. That is, it cannot be guaranteed that the time T_m at which dynamic flow of value $\mu(v_i)$ can be sent to v_i is the minimum time to attain this flow value. One should check whether the same flow value is attained for some lesser time. Thus, for $(T_m) = \mu(v_i)$, T_m is optimal if and only if $f(T_m - 1) < \mu(v_i)$ for all $v_i \in S$ as suggested in [10] for $s - d$ quickest flow problem.

During the procedure, flow computed by the application of LexMDF Algorithm as a subroutine is optimal due to Theorem 3.7. Also, this algorithm runs in strongly polynomial time due to Theorem 3.8. The next major step in the procedure is to perform a binary search over time horizon repeatedly. This can be done in strongly polynomial time. Thus, from above algorithmic discussion we can assert that LexQF problem on UPL network N can be solved optimally in strongly polynomial time.

Concluding remark

Evacuation problems that aim to keep maximum evacuees on the intermediate places besides a maximum evacuees into the specified safe destination are of particular interest to the evacuators. Intermediate places could be of limited capacities and prioritized with respect to facilities at shelter, distance from source, holding capacities, etc. In this paper we studied the maximum version of problem that aims to lexicographically maximize the evacuees entering a set of capacitated terminals with respect to a given prioritization. We proposed an efficient algorithm, based on temporally repeated flows, for the problem modeled on UPL network. We also applied this solution as a subroutine to solve LexQF problem. Moreover, we studied LexEAF problem with sufficient vertex capacities, and propose an efficient algorithm for UPL-TTSP network. Searching of efficient solutions for the problems with more general network settings would be future research work. Multi-commodities flow problems with vertex capacities would also be interesting for researchers.

Supplementary Materials: Not applicable.

Author Contributions: Phanindra Prasad Bhandari solved the problems and prepared the initial manuscript. Shree Ram Khadka improved the solution procedures and polished the manuscript.

Funding: Please add: "This research received no external funding" or "This research was funded by NAME OF FUNDER, grant number XXX" and "The APC was funded by XXX".

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Please refer.

Acknowledgments: In….

Conflicts of Interest: Declare conflicts of interest

References

1. Baumann, N.; Skutella, M. Earliest arrival flows with multiple sources. *Math. Oper. Res.* **2009**, *34*, 499--512.
2. Bein, W.W.; Brucker, P.; Tamir, A. Minimum cost flow algorithms for series-parallel networks}, *Discrete Appl. Math.* **1985**, *10*, 117--124.
3. Bhandari, P.P. Dynamic Network Contraflow Evacuation Planning Problem, doctoral dissertation, Central Department of Mathematics, Tribhuvan University, Nepal, 2021.
4. Bhandari, P.P.; Khadka, S.R. Maximum flow evacuation planning problem with non-conservation flow constraints at the intermediate nodes. In: International Conference on Mathematical Optimization (8-13 April 2019, Beijing). Retrieved on 12 April 2020 from: https://www.researchgate.net/publication/332344781
5. Bhandari, P.P.; Khadka, S.R. Evacuation planning problems with intermediate storage. In: AIJR Proceedings of International Conference on Applied Mathematics \& Computational Sciences (ICAMCS-2019), Dehradun (2020) 90--95.
6. Bhandari, P.P.; Khadka, S.R. Maximum flow evacuation planning problem with non-conservation flow constraint. *Int. Ann. Sci.* **2020**, 10, 25--32.
7. Bhandari, P.P.; Khadka, S.R. Lexicographically Maximum Contraflow Problem with Vertex Capacities. *International Journal of* Mathematics *and Mathematical Sciences.* **2021**, *2021, Article ID 6651135, 7 pages. https://doi.org/10.1155/2021/6651135*.
8. Bhandari, P.P.; Khadka, S.R., Ruzika, S.; L.E. Schafer, L.E. Lexicographically maximum dynamic flow with vertex capacities. *J. Math. Stat.* **2020**, 16(1), 142--147.
9. Borradaile, G.; Klein, P.N.; Mozes, S.; Nussbaum, Y.; Wulff-Nilsen, C. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.* **2017**, 46(4), 1280--1303.
10. Burkard, R.E.; Dlaska, K.; Klinz, B. The quickest flow problem. *ZOR-Methods and Models of Operations Research* **1993,** 37(1) 31--58.
11. Cherkassky, B. V.; Goldberg, A.V. On implementing the push-relabel method for the maximum flow problem. *Algorithmica* **1997**, 19(4), 390--410.
12. Ford, L.R.; Fulkerson, D.R. Constructing maximal dynamic flows from static flows. *Oper. Res.* **1958**, 6(3), 419--33.
13. Gale, D. Transient flows in networks. *Michigan Math. J.* **1959**, 6, 59--63.

14. Goldberg, A.V.; Tarjan, R.E. Finding minimum-cost circulations by canceling negative cycles. *J. ACM* **1989**, 36(4), 873--886.
15. H. Hamacher, H.; Heller, S.; Klein, W.; Koster, G.; Ruzika, S. A sandwich approach for evacuation time bounds. *Pedestrian and Evacuation Dynamics* **2011**, 503--513. Springer.
16. Hamacher, H.; Tjandra, S. Mathematical modelling of evacuation problems: A state of art. Technical Report 24, Fraunhofer (ITWM) 2001.
17. Hoppe, B.; Tardos, E. Polynomial time algorithms for some evacuation problems. In: SODA 1994, 94, 433--441.
18. Hoppe, B.; Tardos, E. The quickest transshipment problem. Math. Oper. Res. 2000, 25, 36--62.
19. Kamiyama, N. Lexicographically optimal earliest arrival flows. Networks 2020, 75(1), 18--33.
20. Khadka, S.R.; Bhandari, P.P. Dynamic network contraflow evacuation planning problem with continuous time approach. Int. J. Oper. Res. (Taichung) 2017 14(1), 27--34.
21. Khadka, S.R.; Bhandari, P.P. Model and solution for non-conservation flow evacuation planning problem. Nepali Math. Sci. Rep. 2019, 36, 11--16.
22. M. Lin, M.; Jaillet, P. On the quickest flow problem in dynamic networks: a parametric min-cost flow approach. In: Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms 2015, 1343--1356.
23. Megiddo, N. Optimal flows in networks with multiple sources and sinks. Math. Program. 1974 7(1) 97--107.
24. Megiddo, N. A good algorithm for lexicographically optimal flows in multi-terminal networks. Bull. Amer. Math. Soc. 1977, 83(3) 407--409.
25. Minieka, E. Maximal, lexicographic, and dynamic network flows. Oper. Res. 1973, 21(2), 517--527.
26. Pyakurel, U.; Dhamala, T.N.; Dempe, S. Efficient continuous contraflow algorithms for evacuation planning problems. Ann. Oper. Res. 2017, 254, 335--364.
27. Rebennack, S.; Arulselvan, A.; Elefteriadou, L.; Pardalos, P.M. Complexity analysis for maximum flow problems with arc reversals. J. Comb. Optim. 2010, 19(2), 200--216.
28. Ruzika, S.; H. Sperber, H.; Steiner, M. Earliest arrival flows on series-parallel graphs. Networks 2011, 57(2), 169--173.
29. Schmidt, M.; Skutella, M. Earliest arrival flows in networks with multiple sinks. Discrete Appl. Math., 2014, 164, 320--327.
30. Wilkinson, W.L. An algorithm for universal maximal dynamic flows in a network. Oper. Res. 1971, 19(7), 1602--1612.