Journal of Mathematics and Statistics

Original Research Paper

Lexicographically Maximum Dynamic Flow with Vertex Capacities

[1]Phanindra Prasad Bhandari, [1]Shree Ram Khadka, [2]Stefan Ruzika and [2]Luca E. Schäfer

[1]Central Department of Mathematics, Tribhuvan University, Kirtipur, Kathmandu, Nepal
[2]Department of Mathematics, Technische Universitat Kaiserslautern, P.O. Box 3049, 67663 Kaiserslautern, Germany

Article history
Received: 11-06-2020
Revised: 29-06-2020
Accepted: 23-07-2020

Corresponding Author:
Shree Ram Khadka
Central Department of Mathematics, Tribhuvan University, Kirtipur, Kathmandu, Nepal
Email: shreeramkhadka@gmail.com

Abstract: We consider an evacuation planning problem in the sense of computing a feasible dynamic flow lexicographically maximizing the amount of flow entering a set of terminals with respect to a given prioritization and given vertex capacities. We propose a polynomial time algorithm for the static version of the problem and a pseudo-polynomial time algorithm for the dynamic case. We show that by neglecting the vertex capacities, the dynamic version can be solved in polynomial time by using temporally repeated flows.

Keywords: Evacuation Planning, Disaster Management, Lexicographically Maximum Flows, Dynamic Flows, Vertex Capacities

Introduction

Mathematical optimization provides important tools for modeling, preparing and managing evacuation tasks, (Borrmann *et al*., 2012; Göttlich *et al*., 2011; Hamacher *et al*., 2011). The maximum flow evacuation planning problem asks for a flow which sends a maximum number of evacuees from a disastrous zone (source) to a safe zone (sink) within a given time horizon. In practical applications there might be relatively safe places apart from the sink. Hence, sending as many of the remaining evacuees as possible to these prioritized spots is desired. However, these spots might be constrained to some given vertex capacities, which restrict the amount of flow that can enter these vertices within the given time horizon.

Since the introduction of the maximum flow problem by (Ford and Fulkerson, 1956), the problem and its applications have been extensively studied in the literature, (Borradaile *et al*., 2017; Cherkassky and Goldberg, 1997; Goldberg and Tarjan, 1988). The lexicographically maximum flow problem has been investigated as a variant of the classical maximum flow problem, (Megiddo, 1974; 1977; Minieka, 1973). The authors showed that this problem can be solved in polynomial time. Dynamic extensions of these problems often provide important features for modeling real-world applications, e.g., in evacuation scenarios. Many dynamic network flow problems have been investigated in the context of evacuation planning problems, (Dhamala, 2015; Hamacher and Tjandra, 2001; Khadka and Bhandari, 2017; Pyakurel *et al*., 2017; Rebennack *et al*., 2010). To solve the maximum dynamic flow problem, a pseudo-polynomial time algorithm based on the construction of a time-expanded graph and a polynomial time algorithm based on temporally repeated flows with transit times on the arcs treated as cost coefficients have been investigated by (Ford and Fulkerson, 1958; 1962). Hoppe and Tardos (1994; 2000) study lexicographically maximum dynamic flows. They developed a polynomial time algorithm based on temporally repeated flows that lexicographically maximizes the flow leaving the terminals of an ordered terminal set consisting of sources and sinks. This is equivalent to lexicographically minimizing the flow entering the sinks in the given order.

In this study, we consider a maximum flow evacuation planning problem with a prioritized terminal set with fixed vertex capacities for each of these vertices. We aim to lexicographically maximize the amount of flow entering the vertices in the terminal set within a given time horizon with respect to the prioritization and the holding capacities. This problem is motivated by the situation encountered in evacuation scenarios: As many evacuees as possible are to be sent to safety. However, if sending evacuees to safety is not possible within a given time horizon, it is desirable to send as many evacuees to shelters of limited capacity, which are located within the evacuation zone. In contrast to the above mentioned models, we assume that the terminals have a fixed vertex capacity delimiting the amount of flow that can enter a vertex within a given time horizon. We provide a

polynomial time algorithm for the static version of the problem and a pseudo-polynomial time algorithm for the dynamic case based on the construction of a time-expanded network. We show how to modify the lexicographically maximum dynamic flow algorithm (Hoppe and Tardos, 1994; 2000) to solve our problem in the case of neglecting the vertex capacities. The optimal routes identified by the procedure we propose in this study allows to send more evacuees out from risk zone to relatively safe places, besides maximum evacuees to the safe zone(sink), at least for some time during the period of response in emergency mitigation.

The remainder of this paper is structured as follows. At first, we formally introduce the maximum flow evacuation planning problem. Then we consider the problem in static and dynamic versions one after another, respectively. Finally, we conclude the paper with some further research objectives.

Problem Formulation

Let $G = (V, A)$ denote a directed graph with vertex set V and arc set A. Both, the set of vertices and the set of arcs are assumed to be finite and we set $n := |V|$ and $m := |A|$. We denote the source and the sink vertex by s and d, respectively. We assume that the graph G does not contain parallel arcs nor loops. Further, we assume that there are no arcs entering s and leaving t, respectively. By $\delta^-(v)$ and $\delta^+(v)$ we denote the set of arcs entering and leaving vertex $v \in V$, respectively. We assume a lower and upper arc capacity function l, u: $A \to \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ to be given, which bounds the number of flow units on each arc at each time step from below and from above. Most of the time, we set $l(a) = 0$ for all $a \in A$. Further, a transit time function t: $A \to \mathbb{N}_0$ specifies the time needed by a flow unit to traverse an arc. We assume a terminal set $S \subset V$ with $S := \{v_1,\ldots,v_k\}$ prioritized from higher to lower priority, i.e., $v_1 \succ v_2 \succ \cdots \succ v_k$, to be given, where $v_1 = d$. Further, we define a vertex capacity function k: $S \to \mathbb{N}_0$ delimiting the total number of flow units, which may be held in each of the vertices $v \in S$. We set $k(d) = \infty$ and $k(v)$ to be finite for all $v \in \mathscr{S} \setminus \{d\}$. In the following, we assume a time horizon $T \in \mathbb{N}$ to be given and treat time in a discrete manner, i.e., $\mathscr{T} := \{0, 1,\ldots,T\}$. Summing it up, we denote by $\mathscr{N} = (G, l, u, \tau, T, k)$ a dynamic network. If we aim to refer to a static network without transit times, we just write $\mathscr{N} = (G, l, u, k)$.

To this end, nonnegative flow variables f: $A \times \mathscr{T} \to \mathbb{N}_0$ specify the flow over time in the network $\mathscr{N}$. More precisely, $f(a, t)$ equals the number of flow units entering arc a at time step t. Further, flow that enters arc a at time t, reaches the end of arc a at time $t + \tau(a)$. 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., $0 \le f(a, t) \le u(a)$ for all $a \in A$ and for all $t \in \mathscr{T}$. Moreover, $f(a, t)$ has to be equal to zero for all $t > T\text{-}\tau(a)$ and for all $a \in A$. The excess of a vertex $v \in V$ at time $t \in \mathscr{T}$ is defined as:

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

Consequently, we need to ensure that $ex_f(v, T) \le k(v)$ for all $v \in \mathscr{S}$.

The objective function of the maximum flow evacuation planning problem asks to lexicographically maximize the vector $(ex_f(v_1, T),\ldots, ex_f(v_k, T))^\top$ such that $ex_f(v_i) \le k(v_i)$ for $i = 1,\ldots,k$. Note that $k(v_1) = k(d) = \infty$ and $v_i \in \mathscr{S}$ for $i = 1,\ldots,k$.

Static Version

Let $\mathscr{N} = (G, l, u, k)$ be a static network without transit times. Further, let $\mathscr{S} = \{v_1,\ldots,v_k\}$ with $d = v_1 \succ \ldots \succ v_k$ prioritized from higher to lower priority. Moreover, we assume that $l(a) = 0$ for all $a \in A$. The goal is to compute a lexicographically maximum flow in $\mathscr{N}$ satisfying the arc capacities for all arcs and the vertex capacities for all $v \in \mathscr{S}$. This is achieved by iterative maximum flow computations in a transformed network as described in the following. First, the network $\mathscr{N}$ with vertex capacities is transformed into a network $\mathscr{N}'$ without vertex capacities. We introduce an artificial vertex v_i' for each vertex $v_i \in \mathscr{S}$. We call the artificial vertex $v_1' := d'$ the supersink. Then, the vertices v_i and v_i' are connected by an artificial arc (v_i, v_i') with $u(v_i, v_i') = k(v_i)$. Moreover, each vertex v_i' is linked to the supersink d' by introducing an artificial arc (v_i', d') having zero arc capacity. Only the artificial arc (d, d') gets infinite arc capacity. Further, every arc (original and artificial) gets a lower arc capacity of zero, i.e., $l(a) = 0$ for all a. Doing this, the network $\mathscr{N} = (G, l, u, k)$ is transformed into the network $\mathscr{N}' = (G', l, u)$ with $G' = (V', A')$, Fig. 1.

Algorithm 1 works as follows. Since $v_1 = d$ is the vertex with the highest priority, a maximum flow from source s to sink d is computed by applying a maximum flow algorithm in the transformed network $\mathscr{N}'$. Next, lower and upper capacities are updated in the following manner: The lower capacities are set to zero, whereas the upper capacities remain the same; only the arc (d, d') gets a lower and upper capacity equal to the value of the previously computed maximum flow from s to d'. Next, we aim to maximize the flow from s to the vertex v_i' with the next highest priority.

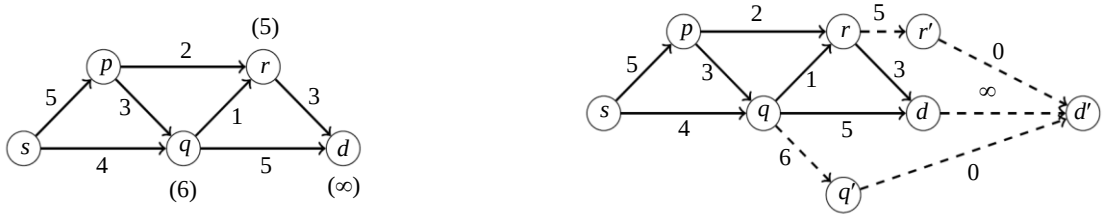

Fig. 1: Network $\mathcal{N}$ with $\mathcal{S} = \{d, r, q\}$, where the numbers on the arcs refer to the arc capacities and the numbers in parenthesis above and below the vertices refer to the vertex capacities (left). Transformed network $\mathcal{N}'$ (right).

Algorithm 1 A lexicographic maximum static flow algorithm

Input: A static network $\mathcal{N} = (G, l, u, k)$, $\mathcal{S} = \{v_1,\ldots,v_k\} \subset V$ with $d = v_1 \succ \cdots \succ v_k$, $l(a) = 0$ for all $a \in A$

Output: A lexicographically maximum flow satisfying the vertex capacities for all vertices in $\mathcal{S}$

1: $d' \leftarrow$ super sink, $V \leftarrow V \cup \{d'\}$, $A \leftarrow A \cup \{(v_1, d')\}$ with $u(v_1, d') = \infty$
2: Compute maximum s-d'-flow in G and let f^* be the optimal value
3: $l(v_1, d') \leftarrow f^*$ and $u(v_1, d') \leftarrow f^*$
4: *for* $i = 2,\ldots,k$ *do*
5: $V \leftarrow V \cup \{v_i'\}, A \leftarrow A \cup \{(v_i, v_i'), (v_i', d')\}$
6: $u(v_i, v_i') \leftarrow k(v_i), l(v_i, v_i') \leftarrow 0$
7: $u(v_i', d') \leftarrow \infty$ and $l(v_i', d') \leftarrow 0$
8: Compute maximum s-d'-flow in $\mathcal{N}$ and let f^* be the optimal value
9: Let $f^*(v_i', d')$ be the flow value on arc (v_i', d') w.r.t. f^*
10: $u(v_i', d') \leftarrow f^*(v_i', d')$ and $l(v_i', d') \leftarrow f^*(v_i', d')$

This is achieved by setting $u(v_i', d')$ to infinity and again computing a maximum flow from s to d' with lower and upper arc capacities. Note that we do not have to find a feasible flow in the transformed network, since the maximum flow computed in the previous iteration is already a feasible flow in the modified network. Further, note that due to the lower and upper capacities on (d, d'), it is ensured that the previously computed maximum flow value from s to d' remains the same. This procedure of computing maximum s-v_i-flows based on previously computed flows is iteratively repeated for all vertices $v_i \in \mathcal{S}$.

Theorem 2.1

Given a network $\mathcal{N} = (G, l, u, k)$, source s and terminal set $\mathcal{S} = \{v_1,\ldots,v_k\} \subset V$ with $d = v_1 \succ \cdots \succ v_k$ and $l(a) = 0$ for all $a \in A$. Then, Algorithm 1 computes a lexicographical maximum flow in $\mathcal{N}$ in polynomial time.

Proof

Obviously, the maximum flow value f^* from s to d' (see line 2) is equal to the maximum flow value in the original network $\mathcal{N}$ from s to d. Let v_j be the vertex with the next highest priority. One can see that the maximum s-d'-flow remains feasible when setting $u(v_1, d')$ and $l(v_1, d')$ to be equal to f^* and introducing arc (v_j', d') with $u(v_j', d') = \infty$. If we compute again a maximum s-d'-flow, then the flow value on arc (v_1, d') is equal to the maximum s-d'-flow computed in the previous iteration, whereas the flow value on arc (v_j', d') is equal to the maximal possible flow that can be sent to v_j' among all maximal s-d'-flows. Repeating this argument, we get a lexicographical maximum s-v_i-flow for all $v_i \in \mathcal{S}$. The polynomial running time follows from the fact that at most $k \leq n$ maximum flow problems are to be solved, which can be done in polynomial time.

Dynamic Version

Time-Expanded Network

Let $\mathcal{N} = (G, l, u, \tau, T, k)$ be a dynamic network. Again, let $\mathcal{S} = \{v_1,\ldots,v_k\}$ with $v_1 \succ \ldots \succ v_k$ be prioritized from higher to lower priority and $l(a) = 0$ for all $a \in A$. Further, without loss of generality we assume that vertices in $\mathcal{S}$ have no outgoing arcs. We transform the dynamic network $\mathcal{N}$ into a time-expanded network $\mathcal{N}^T = (V^T, A^T, l', u')$ in the following way:

- $V^T := \{v_t \mid t = 0,1,\ldots,T\}$,
- $A^T := \{(v_t, w_{t'}) \mid (v = w \wedge t' = t + 1) \vee ((v, w) \in A \wedge \tau(v, w) = t'\text{-}t)\}$,
- $l'(a) = 0$ for all $a \in A^T$,
- $u'(v_t, w_{t'}) = u(v, w)$ for all $(v, w) \in A$ and $u'(v_t, v_{t+1}) = \infty$ for all $v \in V$.

Algorithm 2 A discrete dynamic lexicographic maximum flow algorithm

Input: A dynamic network $\mathcal{N} = (G, l, u, \tau, T, k)$, $\mathcal{S} = \{v_1,\ldots,v_k\}$ with $d = v_1 \succ \cdots \succ v_k$, $l(a) = 0$ for all $a \in A$

Output: A discrete dynamic lexicographically maximum flow satisfying the vertex capacities for all vertices in $\mathcal{S}$

1: Create the time-expanded network as described above
2: **for all** $v \in \mathcal{S}$ **do**
3: $V^T \leftarrow V^T \cup \{v'\}$, $A^T \leftarrow A^T \cup \{(v_T, v')\}$ $\triangleright v' := v'_T$
4: $u'(v_T, v') \leftarrow k(v_i)$, $l'(v_T, v') \leftarrow 0$
5: Compute a static maximum s_0-d'-flow and let f^* be the optimal value
6: $l'(d_T, d') \leftarrow f^*$ and $u'(d_T, d') \leftarrow f^*$
7: **for** $i = 2,\ldots,k$ **do**
8: $A^T \leftarrow A^T \cup \{(v'_i, d')\}$
9: $u'(v'_i, d') \leftarrow \infty$, $l'(v'_i, d') \leftarrow 0$
10: Compute a maximum s_0-d'-flow and let f^* be the optimal value
11: Let $f^*(v'_i, d')$ be the flow value on arc (v'_i, d') w.r.t. f^*
12: $u'(v'_i, d') \leftarrow f^*(v'_i, d')$ and $l'(v'_i, d') \leftarrow f^*(v'_i, d')$

Next, as described in the static version, we introduce a vertex $v'_i := v'_{i_T}$ for all v_{i_T} with $v_i \in \mathcal{S}$. We connect vertices v_{i_T} and v'_i by arcs with upper capacity $k(v_i)$ for all $v_i \in \mathcal{S}$ (and zero transit time, since v_{i_T} and v'_i are on the same time level). Obviously, a static maximum s_0-d'-flow, i.e., a s_0-v'_1 -flow, in the time-expanded network corresponds to a discrete dynamic s-d-flow in $\mathcal{N}$. After computing the static maximum s_0-d'-flow, we set $l'(d_T, d')$ as well as $u'(d_T, d')$ to the value of the maximum s_0-d'-flow. Afterwards, we introduce an arc from v'_i to d' of infinite capacity, where v_i is the vertex with the next highest priority. Then, we compute again a static maximum s_0-d'-flow in the time-expanded network. This procedure is iteratively repeated (similar to Algorithm 1) such that we obtain a discrete dynamic lexicographically maximum flow, see Algorithm 2.

Corollary 3.1

Given a network $\mathcal{N} = (G, l, u, \tau, T, k)$, source s and terminal set $\mathcal{S} = \{v_1,\ldots,v_k\} \subset V$ with $d = v_1 \succ \cdots \succ v_k$ and $l(a) = 0$ for all $a \in A$. Then, Algorithm 2 computes a discrete dynamic lexicographical maximum flow in $\mathcal{N}$ in pseudo-polynomial time.

Proof

The correctness follows from Theorem 2.1. The pseudo-polynomial running time follows from the fact that the size of the time-expanded graph is pseudo-polynomial in the input size.

Temporally Repeated Flows

An optimal solution to minimum cost circulation, flow problem obtained by interpreting transit times as cost coefficients for each arc $a \in A$, can be transformed into a maximal discrete dynamic flow for single-source-single-sink case using the notion of Temporally Repeated Flows (TRFs) (Ford and Fulkerson, 1958). This technique computes feasible optimal flow for $v_1 = d$ as sink for the problem. However, while considering remaining vertices $v_i \in \mathcal{S}$ as the sink, flow computed by TRFs may exceed fixed vertex capacities. Moreover, TRFs may not induce optimal flows for these vertices as sinks due to non-uniqueness of path decomposition carried out for TRFs. These hurdles occur due to the fixed vertex capacities at intermediate vertices. In the following we consider the problem on network without vertex capacities.

Let $\mathcal{N} = (G, l, u, \tau, T)$ be a dynamic network without vertex capacities. Let $\mathcal{S} = \{v_1,\ldots,v_k\}$ be a terminal set with $v_1 \succ \cdots \succ v_k$ prioritized from higher to lower priority and $l(a) = 0$ for all $a \in A$. We aim to solve our problem in polynomial time without vertex capacities by using the lexicographically maximum dynamic flow algorithm proposed in (Hoppe and Tardos, 2000). Their algorithm is summarized in Algorithm 3. Note that $\mathcal{N}^i_{g^{i+1}}$ refers to the residual dynamic network with respect to flow g^{i+1} with vertex set V and arc set A^i.

Algorithm 3 lexicographically maximizes the amount of flow leaving the terminals in $\mathcal{S}$ in the given order, i.e., the algorithm lexicographically maximizes the vector $(-ex_f(v_1, T), \ldots, -ex_f(v_k, T)^\top$. However, we aim at lexicographically maximizing $(ex_f(v_1, T), \ldots, ex_f(v_k, T)^\top$. Therefore, we adapt our problem in the following way such that we can use Algorithm 3 to solve it:

1) $V \leftarrow V \cup \{v'_i\}$, $A \leftarrow A \cup \{(v_i, v'_i)\}$ for all $v_i \in \mathcal{S}$
2) $u(v_i, v'_i) \leftarrow \infty$ and $\tau(v_i, v'_i) \leftarrow 0$
3) $\mathcal{S} = \{v'_1, \ldots, v'_k, s\}$ with $\mathcal{S}^+ \leftarrow \mathcal{S} \setminus \{s\}$ and $\mathcal{S}^- \leftarrow \{s\}$
4) Take the inverse network of $\mathcal{N}$, i.e., $\mathcal{N}^{inv}$, where all arcs are reversed
5) Apply Algorithm 3 on $\mathcal{N}^{inv}$

Algorithm 3 Lexicographically maximum dynamic flow algorithm (Hoppe and Tardos, 2000)

Input: A dynamic network $\mathcal{N} = (G, u, \tau, T)$, $\mathcal{S} = \{v_1,\ldots,v_k\} = \mathcal{S}^+ \dot\cup \mathcal{S}^-$ with $v_1 \succ \cdots \succ v_k$, where $\mathcal{S}^+$ and $\mathcal{S}^-$ refer to the sources and sinks of $\mathcal{S}$, respectively

Output: A lexicographically maximum dynamic flow

1: $V \leftarrow V \cup \{s^*\}$ ▷ Introduce super source s^*
2: $A^{k+1} \leftarrow A \cup \{(s^*, s) \mid s \in \mathscr{S}^*\}$, $u(s^*, s) \leftarrow \infty$, $t(s^*, s) \leftarrow 0$
3: $g^{k+1} \leftarrow 0$ ▷ zero flow
4: $\Gamma^{k+1} \leftarrow \emptyset$ ▷ path decomposition
5: **for** $i = k,\ldots,1$ **do**
6: $A^i \leftarrow A^{i+1}$
7: **if** $v_i \in \mathscr{S}^-$ **then**
8: $A^i \leftarrow A^i \cup \{(v_i, s^*)\}$, $u(v_i, s^*) \leftarrow \infty$, $t(v_i, s^*) \leftarrow -(T+1)$
9: $f^i \leftarrow$ min cost circulation in $\mathscr{N}^i_{g^{i+1}}$ with t as arc costs
10: **if** $v_i \in \mathscr{S}^+$ **then**
11: $A^i \leftarrow A^i \setminus \{(s^*, v_i)\}$
12: $f^i \leftarrow$ min. cost max. s^*-v_i-flow in $\mathscr{N}^i_{g^{i+1}}$ with τ as arc costs
13: $g^i \leftarrow g^{i+1} + f^i$
14: $P^i \leftarrow$ path decomposition of f^i
15: $\Gamma^i \leftarrow \Gamma^{i+1} \cup P^i$
16: return Γ^1

This procedure yields a dynamic flow in $\mathscr{N}^{inv}$ lexicographically maximizing the amount of flow leaving the terminals in $\mathscr{S}$ in the given order. By translating the obtained dynamic flow back to the network $\mathscr{N}$, we obtain the desired dynamic flow lexicographically maximizing the amount of flow entering the terminals in the given order. The correctness of the procedure follows immediately from the correctness of the lexicographically maximum dynamic flow algorithm, see Hoppe and Tardos (2000). Since the overhead of this procedure is determined by the computation of the k minimum cost flows, the polynomial runtime follows.

Conclusion

In this article, we have introduced a maximum flow evacuation planning problem, where one aims to lexicographically maximize the amount of flow entering the vertices of a given prioritized terminal set with respect to vertex capacities. We showed how to solve this problem in polynomial time for the static case and in pseudo-polynomial time for the dynamic case in the time-expanded network. By neglecting the vertex capacities, we provided a procedure to solve that problem in polynomial time in a dynamic network. This work identifies optimal routes to the prioritized vertices besides the safe zone (sink). This allows to send more evacuees out from the disastrous zone (source) to the relatively safe places at least for some time during the period of response in emergency mitigation.

The main shortcoming of this work is one cannot repeatedly send the evacuees at those vertices where vertex capacity is fixed. In the future, it would be interesting to see how to solve that problem in the dynamic case by using temporally repeated flows and satisfying given vertex capacities.

Acknowledgment

First author would like to thank University Grants Commission, Nepal for PhD Fellowship Award 2016. First and second authors are also grateful to GraThO project between TU Kaiserslautern, Germany; TU, Nepal and MSU, Philippines supported by DAAD for providing a suitable atmosphere to conduct this research work. This work was partially supported by the Bundesministerium fur Bildung und Forschung (BMBF) under Grant No. 13N14561. Authors are grateful to anonymous referees for their insightful comments that significantly improved the paper.

Author's Contributions

Phanindra Prasad Bhandari: Prepared the initial manuscript with solution in static and dynamic over the time expanded graph.

Shree Ram Khadka: Initiated the problem, developed the model and polished.

Stefan Ruzika: Improved the model and solution procedure and polished.

Luca E. Schäfer: Contributed in dynamic case with temporally repeated approach and improved the manuscript.

Ethics

There is no non-ethical issues involved in the article. It is original and contains unpublished materials.

References

Borradaile, G., Klein, P. N., Mozes, S., Nussbaum, Y., & Wulff-Nilsen, C. (2017). Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. SIAM Journal on Computing, 46(4), 1280-1303.

Borrmann, A., Kneidl, A., Köster, G., Ruzika, S., & Thiemann, M. (2012). Bidirectional coupling of macroscopic and microscopic pedestrian evacuation models. Safety science, 50(8), 1695-1703.

Cherkassky, B. V., & Goldberg, A. V. (1997). On implementing the push-relabel method for the maximum flow problem. Algorithmica, 19(4), 390-410.

Dhamala, T. N. (2015). A survey on models and algorithms for discrete evacuation planning network problems. Journal of Industrial & Management Optimization, 11(1), 265.

Ford, L. R., & Fulkerson, D. R. (1956). «Maximal Flow through a Network», Canadian Journal of Mathematics.

Ford, L. R., & Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. Operations research, 6(3), 419-433.

Ford, L. R., & Fulkerson, D. R. (1962). Flows in networks princeton university press. Princeton, New Jersey, 276, 22.

Goldberg, A. V., & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. Journal of the ACM (JACM), 35(4), 921-940.

Göttlich, S., Kühn, S., Ohst, J. P., Ruzika, S., & Thiemann, M. (2011). Evacuation dynamics influenced by spreading hazardous material. Networks & Heterogeneous Media, 6(3), 443.

Hamacher, H. W., Heller, S., Klein, W., Köster, G., & Ruzika, S. (2011). A sandwich approach for evacuation time bounds. In Pedestrian and Evacuation Dynamics (pp. 503-513). Springer, Boston, MA.

Hamacher, H. W., & Tjandra, S. A. (2001). Mathematical modelling of evacuation problems: A state of art.

Hoppe, B., & Tardos, É. (1994, January). Polynomial Time Algorithms for Some Evacuation Problems. In SODA (Vol. 94, pp. 433-441).

Hoppe, B., & Tardos, É. (2000). The quickest transshipment problem. Mathematics of Operations Research, 25(1), 36-62.

Khadka, S. R., & Bhandari, P. P. (2017). Dynamic network contraflow evacuation planning problem with continuous time approach. International Journal of Operations Research, 14(1), 27-34.

Megiddo, N. (1974). Optimal flows in networks with multiple sources and sinks. Mathematical Programming, 7(1), 97-107.

Megiddo, N. (1977). A good algorithm for lexicographically optimal flows in multi-terminal networks. American Mathematical Society, 83(3).

Minieka, E. (1973). Maximal, lexicographic and dynamic network flows. Operations Research, 21(2):517–527.

Pyakurel, U., Dhamala, T. N., & Dempe, S. (2017). Efficient continuous contraflow algorithms for evacuation planning problems. Annals of Operations Research, 254(1-2), 335-364.

Rebennack, S., Arulselvan, A., Elefteriadou, L., & Pardalos, P. M. (2010). Complexity analysis for maximum flow problems with arc reversals. Journal of Combinatorial Optimization, 19(2), 200-216.