Lexicographically Maximum Flows under an Arc Interdiction

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

[1] Science and Humanities Department, Khwopa Engineering Colleg, Bhaktapur, Nepal
[2] Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal

Correspondence to: Shree Ram Khadka, Email: shree.khadka@cdmath.tu.edu.np

Abstract: *Network interdiction problem arises when an unwanted agent attacks the network system to deteriorate its transshipment efficiency. Literature is flourished with models and solution approaches for the problem. This paper considers a single commodity lexicographic maximum flow problem on a directed network with capacitated vertices to study two network flow problems under an arc interdiction. In the first, the objective is to find an arc on input network to be destroyed so that the residual lexicographically maximum flow is lexicographically minimum. The second problem aims to find a flow pattern resulting lexicographically maximum flow on the input network so that the total residual flow, if an arc is destroyed, is maximum. The paper proposes strongly polynomial time solution procedures for these problems.*

Keywords: Network flow, Capacitated vertices, Lexicographically maximum flows, Network interdiction problem

DOI: https://doi.org/10.3126/jnms.v4i2.41459

1 Introduction

An interdiction problem arises when a supplier (e.g., a smuggler) has a plan to supply a commodity (drug packets) from one place to another through a transportation network, and, in the mean time, an interdictor (anti-drug agency) gets information about the plan. The interdictor has limitation that the plan cannot be destroyed completely but a desired link in the network be destroyed to seize the commodity passing through it. Thus, the interdictor aims to interdict a link through which a maximum packets of commodity can pass. Supplier is aware of possible loss of commodity from interdictor. So, s/he aims to dispatch the packets of commodity in such a way that, if any link is interdicted, the number of lost packets is minimum. Here, both, supplier's problem and interdictor's problem, can be interpreted as network interdiction problem with different flavors, e.g., [2, 11, 13, 14]. First problem is *supplier's network interdiction (SNI) problem* that wishes for a minimum loss of flow (alternatively, to maximize the residual flow) and second is *interdictor's network interdiction (INI) problem* that wishes for a maximum loss of flow (alternatively, to minimize the residual maximum flow), if interdiction on network with a limited resources occurs. Optimal flow pattern obtained as a solution of SNI problem is also helpful to interdictor for necessary actions, anti-drug efforts for example, and utilizing limited resources. Likewise, network interdiction problem arises in emergency evacuations, in the management of communication system, power grid, logistic supply and in controlling communicable diseases when terrorist attacks, looting, vandalism or some action of an adversary occurs.

Network interdiction problem arises as an extension of the maximum flow problem [6]. The network interdiction problem was first introduced in [13]. Since then numbers of models for the problem with different applications have been taken into consideration and solutions are proposed. Some of models, solution approaches and their complexities are discussed in [5, 9, 12]. The scenario at which interdictor injects some adversarial traffic flows on the network to reduce its throughput flow is modeled in [7]. Authors in [10] gave the network interdiction model in which dynamic interaction between the strategies and responses of the interdictor and supplier based on the evolving state of network resulting from their immediate decisions occurs. Network interdiction problem is modeled with transportation network that consists vertices and directed arcs. The source and the sink are two specified vertices that represent the place from where the flow units (drugs, goods, data, etc.) are dispatched and the place at which the dispatched units are

supposed to arrive, respectively. Remaining vertices are intermediate vertices that serve as transshipment vertices as well as temporary storage. Each vertex is connected to one or more other vertices by directed arcs representing the links between two places showing to and fro direction. Each arc has its capacity that bounds the flow units traveling through it from above. Then the objective of a SNI problem is to find a maximum flow pattern on input network in such a way that, if interdiction occurs, the residual flow is maximum. On the other hand, the objective of INI problem is to find interdiction spots (arcs and/or vertices) on input network to destroy so that the residual maximum flow is minimum, provided that resources are limited. These two problems are quite different in nature [2].

It is not always easy to reorient the flow traveling through an arc if it is interdicted (destroyed). Therefore, flow along this arc is lost. In such situation, besides sending the maximum flow units of goods to the destination, a supplier may also wish to send to the intermediate spots of given priority order maximally for reasons– potential threat of damaging, looting or seize of the goods at source; storage capacity and high rental rates at source, etc. The prioritization depends upon the status of the spot: potential threats from interdictors, storage capacity, rental rate, etc. Stored units of goods can be transshipped to the destination later. Motivated from these the paper considers SNI problem and INI problem studied in [2] with some modifications in model. Modified model is based on the network flow model with weak flow conservation constraints where intermediate vertices may have fixed storage capacities and are prioritized for storage purpose. In the modified SNI problem it is aimed to find a lexicographic maximum flow pattern in such a way that the total residual flow is maximum, if an arc interdiction is allowed. The modified SNI problem is termed as *lexicographic maximum flow interdiction (LexMaxFI) problem* in the paper. The network flow problem with this later objective without any interdiction is termed as *lexicographic maximum flow (LexMaxF) problem* and has been studied in [4] (cf. [3]) with its model and polynomial time solution. Likewise, in INI problem the objective is to find such an arc on the network to interdict so that the residual lexicographic maximum flow is lexicographically minimum. The INI problem is termed as *lexicographic min-max flow interdiction (LexMinMaxFI) problem.* The later problem can also be viewed as an extension of *most vital arc problem* [1].

Rest of the paper is organized as follows. Section 2 presents a mathematical model of lexicographic maximum flow problem [4] and formally defines LexMaxFI problem and LexMinMaxFI problem. Solution procedures for both problems have been discussed in Section 3. Section 4 concludes the paper.

2 Problem Formulation

A directed graph $G = (V, A)$ with vertex set V and arc set A, both to be finite, such that $n := |V|$ and $m := |A|$ is taken for modeling the problems. Two specified vertices denoted by s and d represent the source and the sink, respectively. A terminal set $S \subset V$ with $\mathcal{S} := \{v_1, \ldots, v_r\}$ prioritized from higher to lower priority, i.e., $d = v_1 \succ \cdots \succ v_r$, to be given. Then the corresponding two terminal transportation network is represented as $\mathcal{N} = (G, l(a), u(a), k(v), s, d)$. Here, $l : A \to \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ and $u : A \to \mathbb{N}_0$ represent the lower and upper arc capacity function which bounds the number of flow units on each arc $a \in A$ from below and from above, respectively. Similarly, the vertex capacity function $k : \mathcal{S} \to \mathbb{N}_0$ delimits the total number of flow units, which may be held in each of the vertices $v \in S$. A transportation network $\mathcal{N}$ is depicted in Figure 1.

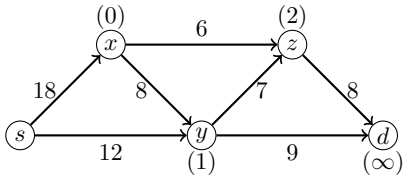

Figure 1: A network $\mathcal{N}$ with arc capacity next to each arc and the vertex capacity inside the parenthesis near by each vertex except the source s.

The non-negative flow variables $f(a)$ defined by $f : A \to \mathbb{N}_0$ specifies the number of flow units entering arc $a \in A$ in the network $\mathcal{N}$. Such flow units are dispatched on $\mathcal{N}$ from the source s. The number of flow units $f(a)$ entering arc a is assumed to be bounded by the corresponding capacity $u(a)$. That is,

$$0 \leq f(a) \leq u(a) \ \forall \ a \in A. \tag{1}$$

The excess flow at vertex $v \in V$, denoted by $ex_f(v)$, is defined as

$$0 \leq ex_f(v) := \sum_{a \in \delta^-(v)} f(a) - \sum_{a \in \delta^+(v)} f(a) \tag{2}$$

where $\delta^-(v) := \{a \in A : a = (w, v) \text{ for some vertex } w \in V\}$ and $\delta^+(v) := \{a \in A : a = (v, w) \text{ for some vertex } w \in V\}$ denote the set of arcs entering and leaving vertex $v \in V$, respectively.

Further, we need to ensure that the excess flow at each vertex $v \in \mathcal{S}$ is to be bounded by the capacity $k(v)$. That is,

$$ex_f(v) \leq k(v) \text{ for all } v \in \mathcal{S}. \tag{3}$$

To this end, the objective function of lexicographic maximum flow problem asks to lexicographically maximize the vector $(ex_f(v_1), \ldots, ex_f(v_r))^\top$ subjected to the constraints (1)–(3).

Furthermore, it is assumed that the flow traveling on an arc cannot be reoriented but is lost totally (partial loss is prohibited), if the arc is destroyed. The model do not allow any loops in the network. For a network with these setup, the objective of INI problem is to lexicographically minimize the lexicographic maximum flow on $\mathcal{N}$, if an arc $(v, w) \in A$ is allowed to destroy. The network flow problem with this objective is phrased as *lexicographic min-max flow interdiction problem* and abbreviated as LexMinMaxFI problem. For the same network setup, the objective of SNI problem is to find a lexicographic maximum flow pattern on $\mathcal{N}$ so that the total residual flow is maximum, if an arc is destroyed. The network flow problem with this objective is phrased as *lexicographic maximum flow interdiction problem* and abbreviated as LexMaxFI problem.

3 Solution Discussion

At the first part it is considered a LexMinMaxFI problem for the network $\mathcal{N}$ with terminal set $\mathcal{S}$ in $\mathcal{N}$ as described in Section 2 where $k(d) = \infty$ and $k(v)$ to be finite for all $v \in \mathcal{S} \setminus \{d\}$. It is sufficient to find an arc on input network $\mathcal{N}$ to be removed such that its removal optimizes the objective of LexMinMaxFI problem. A straight forward technique [2] to find such an arc is as follows: Compute, for each arc, the lexicographic maximum flow value in the reduced network with removed arc of consideration; and pick the arc up whose removal minimizes the lexicographic maximum flow value. In the case of network without intermediate storage capability, the number of iteration could be reduced by taking only those arcs in to consideration whose capacities are at least the maximum capacity of the arc in min-cut of the network [2]. However, it is not possible in the case of network with intermediate storage capability. The noteworthy step, in finding such arc, is the computation of lexicographic maximum static flow (LexMaxSF) [4] corresponding to each arc $a \in A$.

The computation procedure (Algorithm 1) for solving LexMaxSF problem is as follows. First, the network $\mathcal{N}$ with vertex capacities is transformed into a network $\mathcal{N}'$ without vertex capacities. Then an artificial vertex v_i' for each vertex $v_i \in \mathcal{S}$ is introduced. The artificial vertex $v_1' := d'$ is 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 super-sink 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 $\mathcal{N} = (G, l, u, k)$ is transformed into the network $\mathcal{N}' = (G', l, u)$ with $G' = (V', A')$. Then any of the maximum flow algorithms can be applied on reduced network repeatedly for each intermediate vertex $v_i : k(v_i) > 0$ in given priority order.

Algorithm 1 A lexicographic maximum flow (LexMaxF) algorithm [4]

Input: A 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 $\mathcal{S}$

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

The aim, in the next problem, is to find a lexicographic maximum $s - d$ flow pattern on N such that the total residual $s - d$ flow, if an arc on $\mathcal{N}$ is destroyed, is maximum. To find such flow pattern the total flow value is scattered among as many arcs as possible so that the damage of any arc results minimum loss of flow. That is, it requires to find a number λ, as small as possible, the flow $f(u, v)$ for any arc (u, v) is not greater than λ, and the corresponding flow value is not less than the original maximum $s - d$ flow.

Before finding such λ the input network $\mathcal{N}$ is transformed into new network $\mathcal{N}^+$ by introducing a super-sink D and connecting it to each vertices $v \in \mathcal{S}$ by an artificial arc (v, D) with capacity $u(v, D)$ to be the corresponding vertex capacity $k(v)$. That is, the input network $\mathcal{N}$ is transformed to $\mathcal{N}^+ = (V^+, A^+)$ where $V^+ = V \cup \{D\}$ and $A^+ = A \cup \{(v, D) : v \in V\}$ with $u(v, D) = k(v)$. Now, the transformed network is one without vertex capacities. Transformed network $\mathcal{N}^+$ of network $\mathcal{N}$ depicted in Figure 1 is shown in Figure 2 (left).

To find λ the following steps are performed: At first a maximum $s - D$ flow f^* and min-cut X with value F^* are computed in $\mathcal{N}^+$. Then it is required to initialize λ as $\lambda := max\{u(v, w) : (v, w) \in X\}$, and construct new network $\mathcal{N}^+(\lambda)$ as $\mathcal{N}^+(\lambda) := (V^+, A^+, u(\lambda))$ where $u(\lambda) := min\{u(v, w), \lambda\}$ for all $(v, w) \in A \backslash \{(d, D)\}$. λ is computed, in iterative way until $F(\lambda) = F^*$, as $\lambda := \lambda + (F^* - F(\lambda))/q(\lambda)$. Here, $q(\lambda)$ denotes the number of arcs $(v, w) \in X(\lambda)$ with $u(v, w) = \lambda$. The network $\mathcal{N}^+(\lambda)$ is updated with latest value of λ. The capacity of arc (d, D) remains unaltered in each iteration. Finally, all the artificial arcs are removed from updated $\mathcal{N}^+(\lambda)$ restoring the vertex capacities and Algorithm 1 is applied on it to compute lexicographic maximum static flow. The procedure is summarized in Algorithm 2.

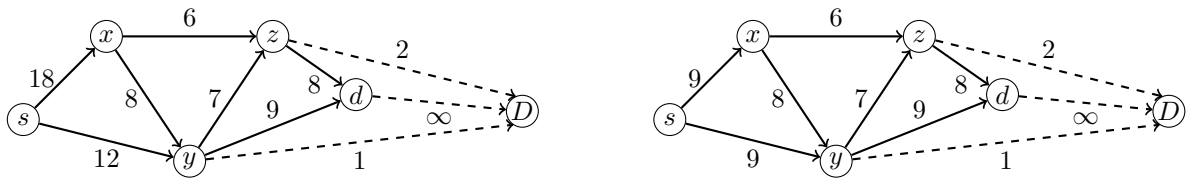

Figure 2: Transformed network $\mathcal{N}^+$ of network $\mathcal{N}$ depicted in Figure 1 with terminal set $\mathcal{S} = \{d, z, y\}$ (left) and network $\mathcal{N}^+(\lambda)$ with $\lambda = 9$ (right).

Example 1: Consider the network $\mathcal{N}$ depicted in Figure 1 with source s and sink d. Consider the terminal set $\mathcal{S} : \{d, z, y\}$ with priority order $d \succ z \succ y$. Transform $\mathcal{N}$ to $\mathcal{N}^+$ as shown in Figure 2 (left).

Algorithm 2 A lexicographic maximum flow interdiction algorithm

Input: A 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 lexicographic maximum flow pattern on $\mathcal{N}$ with maximum residual flow, if an arc on $\mathcal{N}$ is destroyed

1: Transform network $\mathcal{N}$ into $\mathcal{N}^+ = (V^+, A^+)$ where $V^+ = V \cup \{D\}$ and $A^+ = A \cup \{(v, D) : v \in \mathcal{S}\}$ with $u(v, D) = k(v)$
2: Find a maximum flow pattern f^* with D as the sink, and min-cut X with value F^* in $\mathcal{N}^+$
3: Compute λ as in [2]:
    Initialize λ as $\lambda := max\{u(a) : a \in X\}$
    Find $f(\lambda)$, $F(\lambda)$, $X(\lambda)$, $q(\lambda)$ in $\mathcal{N}^+(\lambda)$
    while $F(\lambda) < F^*$ **do**
    $\lambda := \lambda + (F^* - F(\lambda))/q(\lambda)$;
    Find $f(\lambda)$, $F(\lambda)$, $X(\lambda)$, $q(\lambda)$ in $\mathcal{N}^+$ updated with λ;
    end while
4: Modify $\mathcal{N}^+(\lambda)$ by removing all artificial arcs and restoring the vertex capacities
5: Apply Algorithm 1 on modified network $\mathcal{N}^+(\lambda)$

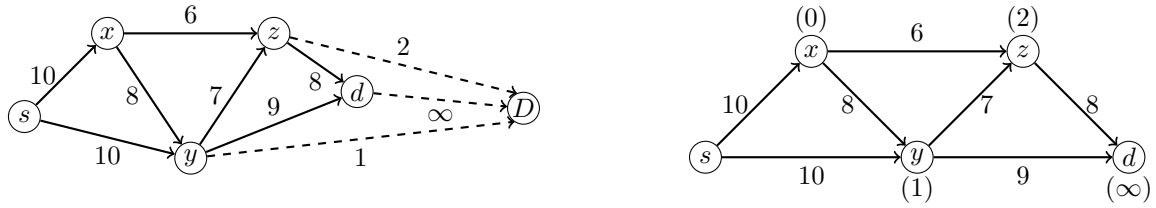

Figure 3: Transformed network $\mathcal{N}^+(\lambda)$ with $\lambda = 10$ (left) and final network after restoring vertex capacities (right).

Application of maximum flow algorithm in $\mathcal{N}^+$ with s as source and D as sink gives maximum flow f^* and min-cut $X = \{(z, D), (z, d), (y, d), (y, D)\}$ with value $F^* = 20$. Here, $\lambda = max\{2, 8, 9, 1\} = 9$. Updating $\mathcal{N}^+$ with current value of λ, Figure 2 (right), and applying maximum flow algorithm on it, the maximum flow is with value $F(\lambda) = 18$ which is less than $F^* = 20$. Also,the min-cut is $\{(s, x), (s, y)\}$. It requires to update λ further. For, $\lambda := \lambda + (F^* - F(\lambda))/q = 9 + (20 - 18)/2 = 10$. Updating $\mathcal{N}^+$ with current value of $\lambda = 10$, Figure 3 (left), and applying maximum flow algorithm on it, the maximum flow is with value $F(\lambda) = 20$ which is equal to F^*. Restoring the vertex capacities on the network $\mathcal{N}^+$ with $\lambda = 10$, Figure 3 (right), and applying Algorithm 1. The destruction that maximizes the loss of flow units is either the arc (s, x) or (s, y) with updated capacities of value 10 for each. Thus, the total residual flow in worst case, when an arc is destroyed, is 10 units.

Theorem 1. *Algorithm 2 solves LexMaxFI problem optimally in strongly polynomial time.*

Proof. The procedure terminates only when the value of λ is computed in such a way that the maximum flow value $F(\lambda)$ in $\mathcal{N}^+(\lambda)$ is equal to the maximum flow value F^* in $\mathcal{N}^+$. Moreover, the flow value F^* is equivalent to the sum of lexicographic maximum flow values on $\mathcal{N}$. On the other hand, the value of λ is as small as possible that bounds the arc capacities from above in $\mathcal{N}^+(\lambda)$. Thus Algorithm 2 computes a lexicographic maximum flow pattern on $\mathcal{N}$ such that the total residual flow, if an arc interdiction occurs, is maximum.

The time complexity of the algorithm is dominated by the complexity of steps 3 and 5. The initial value of $q(\lambda)$ is bounded by the number of arcs in min-cut X that is further bounded by m, and either the step 3 terminates or the value of $q(\lambda)$ decrease by at least one in each iteration. Thus, if one wishes to apply

preflow push algorithm of Goldberg and Tarjan [8] to compute a maximum flow, Step 3 requires $O(n^2m^2)$ times to execute, [2]. Similarly, Step 5 requires $O(n^3m)$ times to compute lexicographic maximum static flow. Thus, Algorithm 2 runs in $O(n^3m + n^2m^2)$ which is strongly polynomial time. □

4 Conclusion

Application of network interdiction problems are of particular interest for suppliers as well as interdictors. This paper discussed two network interdiction problems– *lexicographic min-max flow interdiction problem* and *lexicographic maximum flow interdiction problem*; with capability of intermediate storage of flow units. For this the transportation network with capacitated arcs and vertices is considered and allowed weak conservation flow constraints in the flow model. Solution algorithms for both problems are strongly polynomial time. Solution to the second problem does not guarantee an integral solution since λ need not to be an integral value. However, one can easily obtain integral solution by using the same value of λ, [2].

Study of the problems with consideration of interdiction cost and dynamic (with respect to time) version of the problem would be further research in the field. Lexicographic maximum flow interdiction problem with vertex interdiction would also be an interesting for researchers.

References

[1] Ahuja, R. K., Magnati, T. L. and Orlin, J. B., 1993, *Network flows: theory, algorithms and applications*, Prentice Hall, Englewood Cliffs, New Jersey.

[2] Aneja, Y. P., Chandrasekaran, R. and Nair, K. P. K., 2001, Maximizing residual flow under an arc destruction, *Networks: An International Journal*, 38, 194–198.

[3] Bhandari, P. P. and Khadka, S. R., 2019, 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). `https://www.researchgate.net/publication/332344781`

[4] Bhandari, P. P., Khadka, S. R., Ruzika, S. and Schäfer, L. E., 2020, Lexicographically maximum dynamic flow with vertex capacities, *Journal of Mathematics and Statistics*, 16, 142–147.

[5] Church, R. L., Scaparra, M. P. and Middleton, R. S., 2004, Identifying critical infrastructure: the median and covering facility interdiction problems, *Annals of the Association of American Geographers*, 94, 491-502.

[6] Ford, L. R. and Fulkerson, D. R., 1956, Maximal flow through a network, *Canadian Journal of Mathematics*, 8, 399–404.

[7] Fu, X. Modiano, E., 2019, Network interdiction using adversarial traffic flows. In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 1765–1773.

[8] Goldberg, A. V. and Tarjan, R. E., 1988, A new approach to the maximum-flow problem, *Journal of the ACM (JACM)*, 35, 921–940.

[9] He, M., Du, G. X., Zhang, X. and Zheng, Z., 2018, A cooperative network interdiction model and its optimization solution algorithm, *International Journal of Computational Intelligence Systems*, 11, 560–572.

[10] Lunday, B. J., and Sherali, H. D., 2010, A dynamic network interdiction problem, *Informatica*, 21(4), 553–574.

[11] McMasters, A. W. and Mustin, T. M., 1970, Optimal interdiction of a supply network, *Naval Research Logistics Quarterly*, 17, 261–268.

[12] Murray, A. T., 2013, An overview of network vulnerability modeling approaches, *GeoJournal*, 78, 209–221.

[13] Wollmer, R., 1964, Removing arcs from a network, *Operations Research*, 12, 934–940.

[14] Wood, R. K., 1993, Deterministic network interdiction, *Mathematical and Computer Modelling*, 17, 1–18.