GANIT

Available online at https://ganitjournal.bdmathsociety.org/

GANIT: Journal of Bangladesh Mathematical Society

GANIT *J. Bangladesh Math. Soc.* 42.1 (2022) 010–024

DOI: https://doi.org/10.3329/ganit.v42i1.60997

Saving a Path and Maximizing Dynamic Contraflow: A Bilevel Programming Approach

Hari Nandan Nath[a], Urmila Pyakurel*[b], Stephan Dempe[c], and Tanka Nath Dhamala[b]

[a] *Tribhuvan University, Bhaktapur Multiple Campus, Bhaktapur, Nepal*
[b] *Tribhuvan University, Central Department of Mathematics, Kathmandu, Nepal*
[c] TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 09596 Freiberg, Germany

ABSTRACT

An important aspect of evacuation planning is to save human population by sending the people in dangerous areas (sources) to the safe places (sinks). Optimization models for evacuation planning focus on regulating the traffic flow in urban road networks so as to maximize the number of evacuees reaching the safe places or to minimize the evacuation time of evacuees. Recent studies show that reversing the direction of the usual traffic flow in necessary road segments increases the traffic flow and decreases the evacuation time significantly. However, this may block the flow of some necessary support towards the source. Based on network flow models, we consider the problem of saving a path from a given node to the source, in a single-source-single-sink network, as a bilevel program. With an objective to minimize a function depending on flow rate and the path chosen, the upper level selects a path from a given node to the source, and the lower level maximizes the dynamic flow allowing arc reversals in the resulting network within a given time horizon. We discuss a solution strategy based on replacing the lower level problem by the corresponding KKT conditions.

Received: June 04, 2021 **Accepted:** June 29, 2022 **Published Online:** July 07, 2022

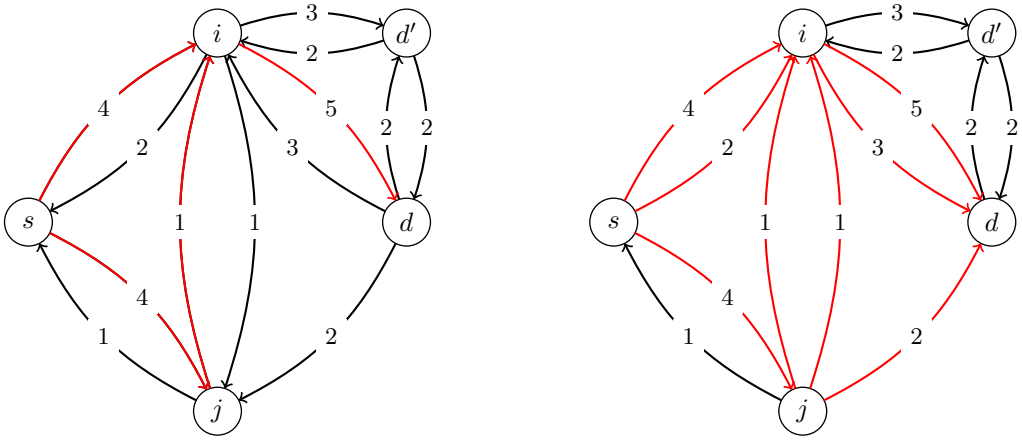
Keywords: Evacuation planning; contraflow; bilevel optimization; maximum dynamic flow; network flow
AMS Subject Classifications 2020: Primary: 90B10, 90C27, 68Q25 Secondary: 90B06, 90B20.

1 Introduction

Saving one's life is a basic instinct of any living organism. Despite the development in science and technology, one of the challenges to save human life, in masses, is from various type of disasters. An important aspect of saving human life from disasters is evacuating human population from disastrous areas to safe places using some means of transportation through the complex urban road network.

*Corresponding author. *E-mail address:* urmilapyakurel@gmail.com

Figure 1.1 Arc occupancy for maximum flow (a) before arc reversal, and (b) after arc reversal (arc labels denote capacities)



Thus a very effective traffic planning is needed to make the traffic flow smooth and send the population to safe places as quickly as possible, or as much as possible within a given time horizon.

To apply mathematical methods and tools in evacuation planning, network flow approach is an active research area. In this approach, an urban road network is represented by a graph or network (arcs representing the road segments, and vertices representing the road intersections or junctions) and various network optimization strategies are used resulting into effective traffic planning, not only in emergency but also in normal situations as well. For a recent survey on such approaches, we refer to Dhamala et al. [1].

In urban road networks, the rule that traffic can move in a particular direction in a particular segment may be contra productive in evacuation scenarios. This is so because during an evacuation, the traffic flow is mostly directed towards the safe area causing the road segments on the direction of paths towards the danger areas (sources) to be less occupied while the segments in the opposite direction become congested. So, an important strategy to optimize flow is to reverse the usual direction of traffic flow wherever necessary. In network optimization, contraflow problems focus on addressing such situations, mathematically, to optimize traffic flow with the ideal direction of arcs in the transportation network. The problem is NP-hard in general and heuristic solutions are available in literature [2, 3]. For specific problems, there are available analytic solutions as well [4, 5, 6, 7].

Using the arc reversal strategy, maximum flow reaching the safe places may be enhanced and the time of evacuation may be reduced. But it may block the paths from a particular node to the source node representing the hazardous area. This obstructs the movement of facilities, if necessary, towards the source. Example 1 illustrates this fact.

Example 1. *Let us consider a network as shown in Figure 1.1(a) with each arc labeled with its capacity. The maximum flow from s to d without using arc reversal is 5 (4 via s–i–d, 1 via s–j–i–d) and that with allowing arc reversal is 10 (6 via s–i–d, 2 via s–j–i–d, 2 via s–j–d), arcs (i, s), (i, j), (d, i), and (d, j) being reversed (see Figure 1.1(b)). Considering the flow, there are no paths available from d' to s after allowing arc reversals.*

To support evacuation, movement of facilities, e.g. ambulance, fire-brigade, etc. towards the sources may be necessary. Hamacher et al.[8], and Nath et al.[9] present models and algorithms which combine location decisions with network flow. From a given set of arcs and a given set of facilities, their approach finds an optimal allocation of the facilities to arcs. In a recent work, Pyakurel et al. [10] focus on identification of the road segments which are not fully occupied, allowing arc reversal, by traffic in evacuation situations so that they can be used for other purposes. Nath et al. [11] model the problem of saving a path as a bicriteria model to minimize the length of the saved path and maximize

the value of the flow allowing arc reversals in the network except the saved path.

The present work proposes a bilevel model to identify a path from a given node to the source, in which the usual direction of the traffic flow has not to be reversed allowing reversals in the remaining segments, for emergency vehicular movement towards the source. In Section 2, we give the basic mathematical notions needed for our modeling. In Section 3, we present the mathematical modeling of the problem as a bilevel optimization problem. Section 4 is devoted to the solution techniques, and Section 5 concludes the paper.

2 Basic Ideas

Let $N = (V, A)$ be a directed network with number of nodes $|V| = n$, and the number of arcs $|A| = m$. For each $i \in V$, we define the set of arcs incoming to i as

$$A_i^- = \{e \in A : e = (j, i) \text{ for some } j \in V\}$$

and the set of outgoing arcs from i as

$$A_i^+ = \{e \in A : e = (i, j) \text{ for some } j \in V\}.$$

Associated with each arc $e \in A$, we consider capacity $u_e \geq 0$, and travel time $\tau_e \geq 0$.

2.1 Static Flow

For two vertices s (called source) and d (called sink) in V with $s \neq d$, a static (s, d)-flow is a function $x : A \to \mathbb{R}_{\geq 0}$ such that

$$\sum_{e \in A_i^+} x_e - \sum_{e \in A_i^-} x_e \begin{cases} \geq 0 & \text{if } i = s \\ = 0 & \text{if } i \in V \setminus \{s, d\} \\ \leq 0 & \text{if } i = d \end{cases} \tag{2.1a}$$

where $x_e = x(e)$, the value of the flow on arc e. The value of the flow x is:

$$\text{val}(x) = \sum_{e \in A_d^-} x_e - \sum_{e \in A_d^+} x_e$$

the amount of the flow reaching the sink d. The flow x is called feasible if the flow on e does not exceed the capacity u_e of arc e, i.e.

$$0 \leq x_e \leq u_e, \forall e \in A \tag{2.1b}$$

For $b : A \to \mathbb{R}$, a b-flow $x : A \to \mathbb{R}$ is defined, so as to satisfy

$$\sum_{e \in A_i^-} x_e - \sum_{e \in A_i^+} x_e = b(i), \forall i \in V \tag{2.2a}$$

A b-flow x is called a circulation if $b(i) = 0, \forall i \in V$. If l_e denotes the lower capacity of an arc $e \in A$, then x is feasible if

$$l_e \leq x(e) \leq u_e, \forall e \in A \tag{2.2b}$$

Any static (s, d)-flow x is equivalent to a b-flow with $b(s) = -\text{val}(x), b(d) = \text{val}(x)$ and $b(i) = 0, \forall i \in V \setminus \{s, d\}$. One can easily realize (e.g. by adding all equations in the system (2.2a)) that the necessary condition for the existence of a b-flow is

$$\sum_{i \in V} b(i) = 0 \tag{2.2c}$$

2.2 Flow-Decomposition

Let P be a simple path in N with a starting vertex $s(P)$ and end vertex $d(P)$, a flow x_P of value δ on P is a $(s(P), d(P))$-flow such that

$$x_P(e) = \begin{cases} \delta & \text{if } e \in P \\ 0 & \text{otherwise} \end{cases}$$

If C is a simple cycle in N, we can similarly define a flow on x_C of value δ on C as follows:

$$x_C(e) = \begin{cases} \delta & \text{if } e \in C \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{P}$ be the set of all simple paths and $\mathcal{C}$ be the set of all cycles in N, then a function $x : A \to \mathbb{R}$ defined by

$$x(e) = \sum_{P \in \mathcal{P}: e \in P} x_P(e) + \sum_{C \in \mathcal{C}: e \in C} x_C(e)$$

is a b-flow for some b. Conversely, any b-flow can be decomposed into at most $m + n$ flows on paths and cycles such that:

(i) Every directed path with positive flow connects a vertex i with $b(i) < 0$ to a vertex j with $b(j) > 0$.

(ii) There are at most m circulations in the decomposition.

2.3 Minimum-Cost Flow

Let c_e be a cost associated with $e \in A$, then the cost of the b-flow x is $\sum_{e \in A} c_e x_e$. The minimum cost flow problem seeks to find the feasible b-flow that minimizes the cost of the flow, and has the following formulation

$$\min \quad \sum_{e \in A} c_e x_e \tag{2.3a}$$

$$\text{s.t.} \quad \sum_{e \in A_i^-} x_e - \sum_{e \in A_i^+} x_e = b(i) \tag{2.3b}$$

$$l_e \ \leq \ x_e \leq u_e, \forall e \in A \tag{2.3c}$$

If $b(i) = 0$ for each $i \in V$, then the corresponding problem is known as minimum-cost circulation problem.

2.4 Dynamic Flow or Flow Over Time

Given a time horizon $T \geq 0$, a flow over time χ (also known as a dynamic flow) consists of a Lebesgue measurable function $\chi_e : [0, T) \to \mathbb{R}_{\geq 0}$ for each $e \in A$ (representing the rate of flow entering e at time θ), and

$$\chi_e(\theta) = 0 \text{ for } \theta \geq T - \tau_e \tag{2.4a}$$

Considering a flow, physically, as something consisting of flow particles, the particles entering arc $e = (i, j)$ at its tail i at time θ arrive at its head j exactly τ_e time units later at time $\theta + \tau_e$ so that the outflow rate at the head of arc e at time θ equals $\chi_e(\theta - \tau_e)$.

The flow over time χ is called feasible if

$$\chi_e(\theta) \leq u_e, \forall e \in A, \theta \in [0, T) \tag{2.4b}$$

The excess of the node i at time θ is defined as

$$\text{ex}_\chi(i, \theta) = \sum_{e \in A_i^-} \int_0^{\theta - \tau_e} \chi_e(\xi) d\xi - \sum_{e \in A_i^+} \int_0^{\theta} \chi_e(\xi) d\xi \tag{2.4c}$$

which is the net amount of flow that enters the node i up to time θ. An (s,d)-flow over time is the flow over time which satisfies:

$$\begin{aligned} \mathrm{ex}_\chi(i,\theta) &\geq 0, \forall i \in V \setminus \{s\}, \theta \in [0,T) && \text{(2.4d)} \\ \mathrm{ex}_\chi(i,T) &= 0, \forall i \in V \setminus \{s,d\} && \text{(2.4e)} \end{aligned}$$

The value of the (s,d)-flow over time is:

$$\mathrm{val}_T(\chi) = \mathrm{ex}_\chi(d,T).$$

Constraints (2.4d) and (2.4e) allow to store flow at intermediate nodes for some time as long as it has left the node again before the time horizon is over.

Temporally repeated flow. Given a feasible static flow x and a time horizon T, a flow decomposition on x gives a set of paths $\mathcal{P}$ with flow x_P for each $P \in \mathcal{P}$. If a flow is sent along P at a constant rate x_P from the source during the time interval $[0, T-\tau(P))$, we can obtain a flow over time with time horizon T, where $\tau(P) = \sum_{e\in P} \tau(e)$, is the travel time on path P. For a node i in an s-d path P, let $P_{s,i}, P_{i,d}$ denote s-i path and i-d path on P, and

$$\mathcal{P}_e(\theta) = \{P \in \mathcal{P} : e \in P \text{ and } \tau(P_{s,i}) \leq \theta \text{ and } \tau(P_{i,d}) < T - \theta\}.$$

Then a temporally repeated flow χ is obtained as described in the following equation

$$\chi_e(\theta) = \sum_{P\in\mathcal{P}_e(\theta)} x_P, \ \forall e = (i,j) \in A, \theta \in [0,T) \tag{2.5}$$

The temporally repeated flow described in (2.5) is a feasible (s,d)-flow over time, with strict equality in the constraints (2.4d), i.e. it does not allow waiting in the intermediate nodes (see Skutella, [12]).

2.5 Contraflow Problem

A contraflow problem seeks to optimize the flow allowing arc reversal, i.e. replacing $e = (i,j) \in A$ with (j,i) if necessary. An idea to solve a contraflow problem is to solve the corresponding problem in what is known as an auxiliary network. Given a network $N = (V, A)$, the auxiliary network $\tilde{N} = (V, \tilde{A})$ consists of the same node set V and the arc set

$$\tilde{A} = \{(i,j) : (i,j) \in A \text{ or } (j,i) \in A\}$$

if $e = (i,j)$, we denote its reverse counterpart (j,i) by $\overleftarrow{e}$. The capacity $\tilde{u}_e$ for $e \in \tilde{A}$ is defined as:

$$\tilde{u}_e = u_e + u_{\overleftarrow{e}}$$

where $u_e = 0$ if $e \notin A$. The travel time $\tilde{\tau}_e$ for $e \in \tilde{A}$ is defined as:

$$\tilde{\tau}_e = \begin{cases} \tau_e & \text{if } e \in A \\ \tau_{\overleftarrow{e}} & \text{otherwise} \end{cases}$$

It is generally considered that $\tau_e = \tau_{\overleftarrow{e}}$ [4]. After solving the problem in $\tilde{N}$, a flow decomposition into paths and cycles is performed and the cycle-flows are removed. The flow, thus obtained in $\tilde{N}$, gives the optimal flow with arc reversal and decision to reverse an arc or not in N.

3 Model Formulation

In this section, we develop a bilevel programming model of the problem considered in this work, formulating maximum dynamic contraflow problem with path reversal (which serves as the lower level of the problem) mathematically in a network with positive travel time on arcs.

As seen in Example 1, the flow allowing arc reversals may block arcs so that paths from a particular node to another node, especially, a source node are obstructed. In this section, we consider a problem of identification of a path from a particular node to the source node to optimize the flow with a given objective. Example 2 considers a problem of identification of such a path so as to hamper the maximum static flow the least.

Example 2. *Let us consider a network in Example 1 again. Suppose that s is the source node, d is the sink node and d' is a depot where supplies which are to move from d' to s are located. If the path $d'-i-j-s$ is chosen not to be reversed, then the maximum static flow value in this configuration becomes 9. The maximum flow values after not reversing the different paths from d' to s are listed in Table 3.1. If either $d'-i-j-s$ or $d'-d-i-j-s$ is chosen not to be reversed, the flow value is maximum, i.e. the flow value is decreased the least from the maximum contraflow value 10. If the shortest path with the least decrease in the flow value is to be chosen, it is the path $d'-i-j-s$.*

Table 3.1: Maximum contraflow values without reversing a particular d'-s path (Cf. Figure 1.1)

SN	Path not reversed	Maximum flow
1	$d'-i-s$	8
2	$d'-d-i-s$	8
3	$d'-i-j-s$	9
4	$d'-d-j-s$	8
5	$d'-d-i-j-s$	9
6	$d'-i-d-j-s$	6
7	$d'-d-j-i-s$	5

In the above example, we have not considered the temporal dimension of the flow. Time plays an important role, particularly, in evacuation problems. To incorporate the time aspect, we consider the problem of optimizing the flow over time.

To formulate the problem mathematically, we consider a network $N=(V,A)$ in which, for each $e\in A$ there exists $\overleftarrow{e}=(j,i)\in A$ such that $\tau_e=\tau_{\overleftarrow{e}}>0$. This assumption is non-restrictive because if such $\overleftarrow{e}$ does not exist, we can include one with zero capacity.

We consider a situation which requires to identify a path P from a particular node (representing a depot) to the source node, in which no arc is reversed with a possibility of arc reversals in A which do not form P. Given a time horizon $T\in\mathbb{N}$, we consider a case in which the choice of P is done so as to optimize a function depending on the flow and the choice of P maximizing the flow over time in the network excluding P.

3.1 Maximum Dynamic Flow Problem

Given a time horizon T, the maximum dynamic flow problem seeks to find a feasible flow over time that maximizes its value. In a seminal work, Ford and Fulkerson [13] established that a dynamic maximum flow can be obtained by solving a minimum cost flow problem. In a discrete time set up, the idea is to construct a (d,s) arc with time $-(T+1)$ and infinite capacity and solve the corresponding minimum-cost circulation problem taking time as cost in the resulting network.

The following result, for the continuous time case, connects the dynamic flow with its static counterpart. For the discrete-time version, time T is replaced by $T+1$.

Lemma 1 ([12, 14])**.** *Let x be a feasible static s-d flow with value v then the value of the corresponding temporally repeated dynamic flow is equal to $Tv-\sum_{e\in A}\tau_e x_e$.*

Hence, the maximum dynamic flow problem can be stated as:

$$\min \quad -Tv+\sum_{e\in A}\tau_e x_e \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{e\in A_i^+}x_e-\sum_{e\in A_i^-}x_e=\begin{cases} v & \text{if } i=s\\ 0 & \text{if } i\in V\setminus\{s,d\}\\ -v & \text{if } i=d\end{cases} \tag{3.1b}$$

$$0\ \le\ x_e\le u_e, \forall e\in A \tag{3.1c}$$

The temporally repeated flow with time horizon T, obtained from the static flow x (by solving (3.1)) gives the corresponding dynamic flow with flow value $Tv-\sum_{e\in A}\tau_e x_e$.

3.2 Maximum Dynamic Contraflow Problem

The maximum dynamic contraflow problem seeks to find the maximum amount of flow that can be sent from the source s to the sink d if the direction of arcs can be reversed at time 0. The problem definition and a procedure to solve it can be found in Rebennack et al. [4]. We summarize their procedure in Algorithm 1.

Algorithm 1 Maximum dynamic contraflow algorithm

Input: A network N with arc capacities, travel time on arcs and a time horizon T
Output: Maximum dynamic flow with arc reversal on N

1. Construct the auxiliary network $\tilde{N}$ of N.
2. Calculate a static flow x corresponding to the temporally repeated maximum dynamic flow on $\tilde{N}$.
3. Decompose x into paths and cycles. Remove cycle flows and update x.
4. Arc $\overleftarrow{e} \in A$ is reversed if and only if $e \in A$ and $x_e > u_e$ or $e \notin A$ and $x_e > 0$.
5. Maximum dynamic flow with arc reversal on N = temporally repeated flow generated by x.

Removal of positive flows in cycles in Step 3 ensures the feasibility of the flow in the transformed network after arc reversal. This step can be avoided, if there is no positive flow in cycles. Given a positive travel time on each arc, it can be guaranteed that there is no positive flow in cycles in a solution of the maximum dynamic flow problem.

Lemma 2. *Given $\tau_e > 0, \forall e \in A$, an optimal flow in the solution of the problem (3.1a)-(3.1c) does not have a positive flow in a cycle.*

Proof. Suppose that $D(x) = -Tv + \sum_{e\in A} \tau_e x_e$. Let x^* be the optimal solution of the problem (3.1) with $v = \text{val}(x^*) = v^*$ and suppose the flow decomposition of x^* have a positive flow in cycles. Assume C to be the set of arcs which form a cycle with flow value $\delta > 0$. Define $x^1, x^2 : A \to \mathbb{R}$ by

$$x^1(e) = \begin{cases} x^*(e) & \text{for } e \in A \setminus C \\ x^*(e) - \delta & \text{for } e \in C \end{cases}$$

and x^2 be a flow defined by

$$x^2(e) = \begin{cases} 0 & \text{for } e \in A \setminus C \\ \delta & \text{for } e \in C \end{cases}$$

Then x^1 and x^2 are feasible static flows in N such that $x^1(e) + x^2(e) = x^*(e), \forall e \in A$. Moreover, since a static flow in a cycle does not contribute to the value of the static flow, $\text{val}(x^1) = v^*$. Now,

$$\begin{aligned} D(x^*) &= -Tv^* + \sum_{e\in A} \tau_e x^*(e) \\ &= -Tv^* + \sum_{e\in A} \tau_e x^1(e) + \sum_{e\in A} \tau_e x^2(e) \\ &= -Tv^* + \sum_{e\in A} \tau_e x^1(e) + \delta \sum_{e\in A} \tau_e \\ &> -Tv^* + \sum_{e\in A} \tau_e x^1(e) \\ &= D(x^1) \end{aligned}$$

Since x^1 is a also a feasible static flow with $\text{val}(x^1) = v^*$, this contradicts the optimality of x^*. □

As the capacity of each arc $e \in \tilde{A}$ in the auxiliary network $\tilde{N}$ is the sum of the capacities of e and $\overleftarrow{e}$, and the static flow corresponding to the temporally repeated maximum dynamic flow with time horizon T can be calculated by finding the minimum cost circulation in modified by adding arc (d, s) with infinite capacity and time $-T$ (considering time on arcs as cost), the next result is immediate from Algorithm 1 and Lemma 2.

Theorem 1. *In the network $N = (V, A)$, with capacity u_e, travel time $\tau_e > 0$ for each $e \in A$, the maximum dynamic contraflow problem can be formulated as the following linear program*

$$\min \quad -T\left(\sum_{e\in A_s^+} x_e - \sum_{e\in A_s^-} x_e\right) + \sum_{e\in A} \tau_e x_e \tag{3.2a}$$

$$s.t. \quad \sum_{e\in A_i^+} x_e - \sum_{e\in A_i^-} x_e = 0, \forall i \in V \setminus \{s, d\} \tag{3.2b}$$

$$0 \leq x_e \leq u_e + u_{\overleftarrow{e}}, \forall e \in A \tag{3.2c}$$

In Theorem 1, we have also used the fact that the value of the static flow x is

$$\sum_{e\in A_s^+} x_e - \sum_{e\in A_s^-} x_e = -\sum_{e\in A_d^+} x_e + \sum_{e\in A_d^-} x_e.$$

Note that the flow obtained in the problem stated in Theorem 1 may be infeasible in N, the infeasibility in $e \in A$ implying that $\overleftarrow{e}$ has to be reversed. However, since the flow is feasible in the auxiliary network, an immediate consequence of Lemma 2 is:

Corollary 1. *An optimal flow in the solution of the problem (3.2a)-(3.2c) does not have a positive flow in a cycle.*

3.3 Bilevel Programming

A bilevel programming problem is an optimization problem in which one optimization problem (called the lower level problem) is among the constraints of the other (called the upper level problem). Partitioning the decision variables into two vectors x and y, let the lower level problem be:

$$\min_x \{f(x, y) : g(x, y) \leq 0, h(x, y) = 0\} \tag{3.3a}$$

where $f : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}$, $g : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^p$, $h : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^q$, $g(x, y) = (g_1(x, y), \cdots g_p(x, y))^\intercal$, $h(x, y) = (h_1(x, y), \cdots, h_q(x, y))^\intercal$. Let $\Psi : \mathbb{R}^{m_2} \to 2^{\mathbb{R}^{m_1}}$ be a set-valued mapping such that $\Psi(y)$ denotes the solution set of problem (3.3a) for a fixed $y \in \mathbb{R}^{m_2}$, then the upper level problem is:

$$\min_{x,y} \{F(x, y) : G(x, y) \leq 0, H(x, y) = 0, x \in \Psi(y)\} \tag{3.3b}$$

where $F : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}$, $G : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^k$, $H : \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^l$, $G(x, y) = (G_1(x, y), \cdots G_k(x, y))^\intercal$, $H(x, y) = (H_1(x, y), \cdots, H_l(x, y))^\intercal$. For details, we refer to Dempe [15].

In game theoretic terms, a bilevel programming problem may be considered as a Stackelberg leader-follower game, where the leader (representing upper level) makes the first move by choosing y, and the follower (representing the lower level) reacts by choosing x optimally. The leader selects, finally, such a y that together with x returned by the follower, the upper level objective $F(x, y)$ is optimized.

3.4 Problem Formulation

Now, we formulate the problem of finding a path not to be reversed from a given node d' to the source node to maximize the dynamic flow allowing arc reversal to fulfill a given objective based on path selection and resulting flow, as a bilevel programming problem. The upper level selects a d'-s

path not to be reversed and lower level maximizes the dynamic flow value on the resulting network allowing arc reversal.

The upper level problem is:

$$\min \quad F(x,y) \tag{3.4a}$$

$$\text{s.t.} \quad \sum_{e\in A_i^+} y_e - \sum_{e\in A_i^-} y_e = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V\setminus\{s,d'\} \\ 1 & \text{if } i = d' \end{cases} \tag{3.4b}$$

$$y_e \le u_e, \forall e \in A \tag{3.4c}$$

$$y_e \in \{0,1\}, \forall e \in A \tag{3.4d}$$

where $x = (x_e)_{e\in A}, y = (y_e)_{e\in A}$, and x is obtained by solving the lower level problem:

$$\min \quad -T\left(\sum_{e\in A_s^+} x_e - \sum_{e\in A_s^-} x_e\right) + \sum_{e\in A} \tau_e x_e \tag{3.4e}$$

$$\text{s.t.} \quad \sum_{e\in A_i^+} x_e - \sum_{e\in A_i^-} x_e = 0, \forall i \in V\setminus\{s,d\} \tag{3.4f}$$

$$0 \le x_e \le (1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}, \forall e \in A \tag{3.4g}$$

The constraints (3.4b)-(3.4d) construct a d'-s path with $y_e = 1$ if e lies on the path. Constraints (3.4c) ensure that arcs with positive capacity can only be a part of the path. The lower level problem is maximum dynamic contraflow problem. Constraints (3.4g) bound the flow on an arc e by $u_e + u_{\overleftarrow{e}}$ if neither e nor $\overleftarrow{e}$ is chosen for the path construction by the upper level. If e is chosen and $\overleftarrow{e}$ is not chosen for path construction, flow on e is bounded by $u_{\overleftarrow{e}}$.

We define the upper level objective function as

$$F(x,y) = \sum_{e\in A: u_e\neq 0} w_e(y_e - x_e/u_e) \tag{3.4h}$$

where the weight $w_e \ge 0$ is the utility assigned to e, as realized by the upper level, so as to motivate flow along the arc.

The constraints (3.4b)-(3.4d) construct a d'-s path which may also contain subtours. Unlike in the case of the shortest path problem with non-negative arc costs (lengths), where a solution without subtours is always better than the corresponding solution with subtours, a solution with a subtour may be better than than the one without a subtour in our consideration of the problem. Example 3 illustrates this fact.

Example 3. *Consider a network shown in Figure 3.1. The only path from d' to s is d'-s. Given a time horizon $T = 5$, without taking any subtour, the static flow which generates the temporally repeated maximum dynamic s-d flow (allowing arc reversal) is given by:*

$$x(s,i) = x(i,j) = x(j,d) = 4, x(i,d) = x(j,i) = x(d',s) = 0$$

and the variables related to the path are

$$y(d',s) = 1, y(s,i) = y(i,j) = y(j,d) = y(i,d) = y(j,i) = 0$$

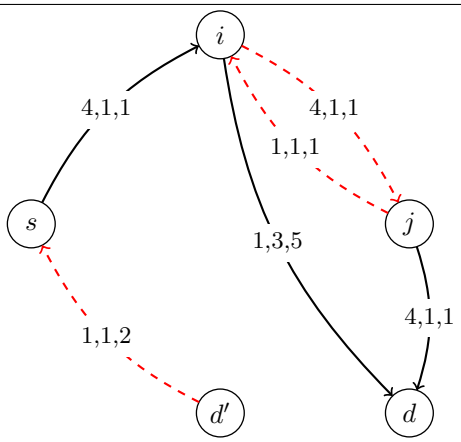
The value of the objective (3.4h) in this case is:

$$F_1 = \sum_{e\in A: u_e\neq 0} w_e y_e - \sum_{e\in A: u_e\neq 0} \frac{w_e x_e}{u_e} = 2 - 3 = -1$$

If the subtour i-j-i is considered along with the path,

$$x(s,i) = x(i,d) = 1$$

Figure 3.1 Path d'-s along with a subtour i-j-i (the arc labels represent capacity, time, weight)



and

$$y(d', s) = y(i, j) = y(j, i) = 1$$

with the objective function value

$$F_2 = \sum_{e \in A: u_e \neq 0} w_e y_e - \sum_{e \in A: u_e \neq 0} \frac{w_e x_e}{u_e} = 4 - (5 + 1/4) = -5/4 < F_1.$$

However, there may be some objectives which can be improved by subtour elimination, e.g. if the second part of the sum in the objective (3.4h) involves the objective of the lower level problem, the subtours may be removed automatically because elimination of a subtour from the path chosen by the upper level, increases the capacity of the arcs for the flow on the lower level.

If subtour elimination is necessary, we can ensure the elementarity of the path (avoid subtours) including the following constraints in the upper level [16, 17, 18]

$$\sum_{e \in A(S)} y_e \leq \sum_{i \in S \setminus \{k\}} \sum_{e \in A_i^+} y_e \ \forall k \in S, \forall S \subseteq V \setminus \{s, d'\}, |S| \geq 2 \tag{3.5a}$$

$$y_e = 0, \forall e \in A_s^-, A_{d'}^+ \tag{3.5b}$$

where $A(S)$ denotes the set of arcs with both the ends in S. Known as generalized cut-set (GCS) inequalities, these inequalities increase the number of inequalities by $O(n2^n)$. Other techniques with polynomial number of additional variables and additional constraints can also be found in Taccari, [18]. For example, replacing the upper level constraints with (3.5b) and the following constraints

$$\pi_i - \pi_j + n y_e \leq n - 1, \forall e = (i, j) \in A \tag{3.6}$$

(see also Bui et al.[17]), the number of additional variables $\pi_i (i = 1, \cdots, n)$ is $O(n)$, and that of additional constraints is $O(m)$. However, it is shown in the literature that the computational performance of GCS inequalities is better than other formulations in solving the shortest elementary path problem.

4 Solution Approaches

We present two approaches to solve the problem. One uses the idea of Stackelberg game along with Algorithm 1, particularly useful when one can get all the paths form d' to s in a desired time, and the other converts the bilevel program to a single-level mixed binary integer linear program so that one can use the whole toolbox to solve a mixed integer programming problems.

4.1 A Naïve Algorithm

A straight-forward procedure to solve the problem is presented in Algorithm 2. The algorithm iteratively chooses a d'-s path and uses Algorithm 1 in the network formed by excluding the path and calculates the objective function value. The output is the path which gives the best objective function value, along with the related flow.

Algorithm 2 Naïve algorithm

Input: directed network $N = (V, A)$, source s, sink d, and a depot node, d', time horizon T

1. $Val_{OBJ} = \infty$
2. LIST = list of simple paths from d' to s without zero capacity arcs
3. For P in LIST
   - Construct the auxiliary network $\bar{N}$ excluding the arcs in P
   - Calculate the static flow x corresponding to the maximum dynamic flow on $\bar{N}$
   - Val_{OBJ}^{new} = value of the objective function with y-values corresponding to P and x
   - If $\text{Val}_{OBJ}^{new} < \text{Val}_{OBJ}$, then $P^* = P, x^* = x$
   - Retain the network
4. Return P^*, x^*

4.2 KKT Approach

A common approach to solve a bilevel programming problem is to transform it into a single level optimization problem replacing the lower level problem by its Karush-Kuhn-Tucker (KKT) conditions. This results into what is known as a mathematical program with equilibrium or complementarity constraints (MPEC or MPCC) (see [19]).

Consider the Lagrangian

$$\begin{aligned} L(x, y, \lambda, \mu) &= -T\left(\sum_{e\in A_s^+} x_e - \sum_{e\in A_s^-} x_e\right) + \sum_{e\in A} \tau_e x_e \\ &\quad + \sum_{i\in V\setminus\{s,d\}} \lambda_i \left(\sum_{e\in A_i^+} x_e - \sum_{e\in A_i^-} x_e\right) \\ &\quad + \sum_{e\in A} \mu_e \left[x_e - (1-y_e)u_e - (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}\right] \end{aligned}$$

The KKT conditions for the lower label problem are (3.4f), (3.4g) along with

$$\lambda_i - \lambda_j + \mu_e \geq -\tau_e, \ \forall e = (i,j) \in A \tag{4.1a}$$

$$\mu_e \left[x_e - (1-y_e)u_e - (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}\right] = 0, \ \forall e \in A \tag{4.1b}$$

$$\mu_e \geq 0, \ \forall e \in A \tag{4.1c}$$

$$\lambda_i \in \mathbb{R}, \ \forall i \in V \setminus \{s, d\} \tag{4.1d}$$

Although there is no λ associated with s, d, the inequalities (4.1a) are valid if we put $\lambda_s = -T$ and $\lambda_d = 0$.

In what follows, we write the upper level constraints as ULC. Thus the single-level problem corresponding to the bilevel problem is the problem (3.4a) subject to ULC, (3.4f), (3.4g), (4.1a)-(4.1d), which is a mixed integer non-linear optimization problem. The nonlinearity is because of the constraints (4.1b), and solution of this problem would give local optimal solutions. However, since the

Mangasarian-Fromovitz constraint qualifications are violated at every feasible point of this problem, the nonlinear optimization solvers may fail to obtain a local optimal solution. This can be overcomed by solving the following relaxation of the problem for $\epsilon \downarrow 0$ (Dempe, [20]).

$$\min \quad F(x,y) \tag{4.2a}$$

such that

$$\text{ULC} \tag{4.2b}$$

$$\sum_{e\in A_i^+} x_e - \sum_{e\in A_i^-} x_e = 0, \qquad \forall i \in V \setminus \{s,d\} \tag{4.2c}$$

$$0 \;\le\; x_e \le (1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}, \qquad \forall e \in A \tag{4.2d}$$

$$\lambda_i - \lambda_j + \mu_e \ge -\tau_e, \qquad \forall e=(i,j)\in A, i\ne s, j\ne d \tag{4.2e}$$

$$-\lambda_j + \mu_e \ge T - \tau_e, \qquad \forall e=(s,j)\in A \tag{4.2f}$$

$$\lambda_i + \mu_e \ge -\tau_e, \qquad \forall e=(i,d)\in A, \tag{4.2g}$$

$$\mu_e \ge 0, \qquad \forall e \in A \tag{4.2h}$$

$$\mu_e\,[x_e - (1-y_e)u_e - (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}})] \le \epsilon, \qquad \forall e \in A \tag{4.2i}$$

Solving problem (4.2), we get the local optimal solutions. But in our consideration of the problem, a global solution is desirable. To get a global optimal solution, we can employ, what is popularly known as, a big M method. Choosing large enough M [21, 22], we can replace (4.1b) by

$$-x_e + (1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}} \;\le\; M'z_e, \forall e \in A \tag{4.3a}$$

$$\mu_e \;\le\; (1-z_e)M'', \forall e \in A \tag{4.3b}$$

Let $U = \max\{u_e : e \in A\}$. Then the maximum possible value of $(1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}$ is $2U$. Since the minimum possible value of x_e is zero, we can safely take M' as $2U$. In this way, we can get the global optimal solution of the problem by solving the following formulation (4.4), with large enough M.

$$\min \quad F(x,y) \tag{4.4a}$$

such that

$$\text{ULC} \tag{4.4b}$$

$$\sum_{e\in A_i^+} x_e - \sum_{e\in A_i^-} x_e = 0, \qquad \forall i \in V \setminus \{s,d\} \tag{4.4c}$$

$$0 \;\le\; x_e \le (1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}}, \qquad \forall e \in A \tag{4.4d}$$

$$\lambda_i - \lambda_j + \mu_e \ge -\tau_e, \qquad \forall e=(i,j)\in A, i\ne s, j\ne d \tag{4.4e}$$

$$-\lambda_j + \mu_e \ge T - \tau_e, \qquad \forall e=(s,j)\in A \tag{4.4f}$$

$$\lambda_i + \mu_e \ge -\tau_e, \qquad \forall e=(i,d)\in A, \tag{4.4g}$$

$$-x_e + (1-y_e)u_e + (1-y_{\overleftarrow{e}})u_{\overleftarrow{e}} \le 2Uz_e, \qquad \forall e \in A \tag{4.4h}$$

$$\mu_e \le (1-z_e)M, \qquad \forall e \in A \tag{4.4i}$$

$$\mu_e \ge 0, \qquad \forall e \in A \tag{4.4j}$$

$$z_e \qquad \in \{0,1\} \tag{4.4k}$$

But caution has to be taken choosing a big M when we use a solver software to solve the problem, because very low or very high values of M lead to infeasible, suboptimal, and numerically unstable solutions.

A method described in Pineda et al. [22], to solve linear bilevel optimization problems, may be adapted to tune the values of big M. The idea is to use the local optimal solutions of the relaxed problem (4.2) to adjust the value of M. We present the procedure in Algorithm 3.

The big M in (4.4i) is replaced by the value returned by Algorithm 3, and a mixed integer programming solver is used to solve problem (4.4). We may take $z_e = 0$ if $-x_e+(1-y_e)u_e+(1-y_{\overleftarrow{e}}) > 0$ and $z_e = 1$ if $\mu_e > 0$ to initialize the variable z.

Algorithm 3 Tuning big M

1. Initialize $\epsilon > 0, \delta > 1, H > 1$, and the number of iterations $K \in \mathbb{N}$.
2. Set initial number of iterations $k = 0$.
3. Solve problem (4.2) without (4.2i). Let the value of the upper level variable $y = (y_e)_{e\in A}$ be y_0 and the value of the dual variable $\mu = (\mu_e)_{e\in A}$ be μ_0.
4. Find the optimal flow $x = x_0$ using the solution $y = y_0$ of the upper level.
5. While $k < K$
   - Increase k by 1.
   - Solve problem (4.2) by taking $x = x_{k-1}, y = y_{k-1}, \mu = \mu_{k-1}$ as an initial solution.
   - Repace ϵ by ϵ/δ.
6. $M = H \times \max\{(\mu_k)_e : e \in A\}$.
7. Return M.

4.3 Identification of Reversed Arcs

From the solution of the aforementioned problem, we can also determine the arcs which are to be reversed. This can be done with the help of the optimal flow x^*. As we know that x^* may violate feasibility with respect to the capacities of the network N, i.e. x^*_e may exceed u_e for some e. This implies that the opposite arc $\overleftarrow{e}$ has to be reversed to increase the capacity in the direction of e. The procedure is mentioned in Algorithm 4.

Algorithm 4 Identifying reversed arcs

1. Identify the optimal path P^* and the static flow x^* corresponding to the dynamic flow
2. $R = \{\overleftarrow{e} \in A \setminus P : x^*_e > u_e\}$
3. **return** R

Because of our assumption that $\tau_e > 0$ for each $e \in A$, there will be no positive flow in cycles in the flow decomposition of x^* according to Corollary 1. So, we do not require to decompose x^* into paths and cycles in Algorithm 4.

5 Conclusion

In this study, we proposed a model to identify an optimal path from a particular node to the source node allowing arc reversals in a network as a bilevel programming problem, the upper level selecting a path and lower level maximizing the dynamic flow. We presented a simple algorithm based on an algorithm to compute maximum dynamic contraflow and the idea of Stackelberg game, when the number of simple paths from the particular node to the source is not very large. We also discussed the transformation of the problem to the single level using KKT conditions of the lower level and outlined a solution strategy to overcome nonlinearity using big M method. The problem considered is particularly useful to facilitate vehicular movement towards the source, with a purpose to facilitate evacuees, during emergency evacuation.

Acknowledgements

The first author acknowledges the supports of UGC Nepal, DAAD IPID4all Young GEOMATENUM International program, and TU Bergakademie Freiberg, Germany. The second author acknowledges the support of Alexander von Humboldt (AvH) Foundation for her post doctoral research stay (November 2017 – October 2019) at TU Bergakademie Freiberg and for Remote Cooperation Abroad Fellowship (March 1 – August 30, 2022). The authors thank AvH for the support of the AvH Research Group Linkage Program between TU Bergakademie Freiberg, Germany and Central Department of Mathematics, TU Kathmandu, Nepal.

References

[1] Dhamala, T. N., Pyakurel, U., Dempe, S., A critical survey on the network optimization algorithms for evacuation planning problems. *International Journal of Operations Research* 15(3): 101 – 133 (2018).

[2] Kim, S., Shekhar, S., Min, M., Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering* 20(8): 1115–1129 (2008).

[3] Vogiatzis, C., Walteros, J. L., Pardalos, P. M., Evacuation through clustering techniques, In Models, Algorithms, and Technologies for Network Analysis, Springer (pp. 185–198) (2013).

[4] Rebennack, S., Arulselvan, A., Elefteriadou, L., Pardalos, P. M., Complexity analysis for maximum flow problems with arc reversals,*Journal of Combinatorial Optimization*, 19(2): 200–216 (2010).

[5] Pyakurel, U., Dhamala, T. N., Models and algorithms on contraflow evacuation planning network problems. *International Journal of Operations Research* 12(2): 36–46 (2015).

[6] Pyakurel, U., Nath, H. N., Dhamala, T. N., Efficient contraflow algorithms for quickest evacuation planning, *Science China Mathematics* 61(11): 2079–2100 (2018).

[7] Pyakurel, U., Dempe, S. (2020), Network flow with intermediate storage: models and algorithms, *SN Operations Research Forum* 1(4): 1–23 (2020).

[8] Hamacher, H. W., Heller, S., Rupp, B., Flow location (FlowLoc) problems: dynamic network flow and location models for evacuation planning, *Annals of Operations Research*, 207(1): 161–180 (2013).

[9] Nath, H. N., Pyakurel, U., Dhamala, T. N., Dempe, S., Dynamic network flow location models and algorithms for quickest evacuation planning.*Journal of Industrial and Management Optimization*, 17(5): 2943–2970 (2020).

[10] Pyakurel, U., Nath, H. N., Dhamala, T. N., Partial contraflow with path reversals for evacuation planning, *Annals of Operations Research*, 283(1): 591–612 (2019).

[11] Nath, H. N., Dempe, S., Dhamala, T. N., A bicriteria approach for saving a path maximizing dynamic contraflow, *Asia-Pacific Journal of Operational Research*, https://doi.org/10.1142/S0217595921500275 (2021).

[12] Skutella, M., An introduction to network flows over time. *Research Trends in Combinatorial Optimization*, Springer, 451–482 (2009).

[13] Ford, L. R., Fulkerson, D. R., Flows in Networks, Princeton: Princeton University Press (1962).

[14] Fleischer, L., Tardos, É., Efficient continuous-time dynamic network flow algorithms *Operations Research Letters*, 23(3-5): 71–80 (1998).

[15] Dempe, S., Foundations of Bilevel Programming, Springer Science & Business Media (2002).

[16] Drexl, M. and Irnich, S., Solving elementary shortest-path problems as mixed-integer programs. *OR Spectrum* 36(2): 281–296 (2014).

[17] Bui, Q. T., Deville, Y. Pham, Q. D., Exact methods for solving the elementary shortest and longest path problems. *Annals of Operations Research*, 244(2): 313–348 (2016).

[18] Taccari, L., Integer programming formulations for the elementary shortest path problem. *European Journal of Operations Research*, 252(1): 122–130 (2016).

[19] Luo, Z. Q., Pang, J. S., Ralph, D. Mathematical Programs with Equilibrium Constraints, Cambridge: Cambridge University Press (1996).

[20] Dempe, S., Computing locally optimal solutions of the bilevel optimization problem using the KKT approach, *International Conference on Mathematical Optimization Theory and Operations Research*, Springer, 147–157 (2019).

[21] Fortuny-Amat J., McCarl, B., A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32(9): 783–792 (1981).

[22] Pineda, S., Bylling, H., Morales, J. M., Efficiently solving linear bilevel programming problems using off-the-shelf optimization software. *Optimization and Engineering*, 19(1): 187–211 (2018).