International Journal of
***O*perations *R*esearch**

International Journal of Operations Research Vol. 15, No. 4, 175-185 (2018)

Network Flow Approach for Locating Optimal Sink in Evacuation Planning

Hari Nandan Nath[1*], Tanka Nath Dhamala[2]

[1]Tribhuvan University, Bhaktapur Multiple Campus, Bhaktapur, Nepal

[2]Tribhuvan University, Central Department of Mathematics, Kathmandu, Nepal

Abstract: Network flow models have been widely applied for evacuation planning, which involves moving people from risk areas (sources) to safe places (sinks) using some means of transportation, to optimize traffic flow in urban road networks. The decisions related to the locations of the sinks are also important to maximize the number of evacuees or minimize time for the evacuees to reach the safe places. In this work, we consider the problems of identifying the optimal sink node out of a given set of possible sink-nodes in a single source network to maximize the flow value, and that to minimize the time to transfer a given flow value to the sink in minimum time. Designing efficient computational procedures to solve the problems, we prove that the problems can be solved with strongly polynomial time complexity. Corresponding optimal sink location problems along with identification of ideal direction of the flow based on contraflow approach are also solved in strongly polynomial time. Our results are substantiated by a case illustration based on Kathmandu road network.

Keyword — Evacuation planning, sink location, network flow, dynamic flow, maximum flow, quickest flow, contraflow

1. INTRODUCTION

Evacuation planning decisions involve the decisions related to the traffic flow on urban road networks to transfer people from danger areas to safe places or shelters. Choosing optimal shelters out of a given number of shelters is an important optimization problem, and is a growing research area. Likewise, reversing the direction of usual traffic flow so as to increase the flow towards the shelters is also one of the active research problems in evacuation planning. We discuss, briefly, some of the works done in these directions.

Sherali, Carter, and Hobeika (1991) develop a location-allocation model to choose a set of shelter locations from among feasible locations. Their approach is a discrete minisum or median location approach, in which they minimize the total time spent by evacuees. They use the congestion related travel time developed by U.S. Bureau of Public Roads (BPR) (1964). Noting that the problem is NP-hard, they develop a fast heuristic which performs well in practice and also present exact enumeration algorithms with running time increase rapidly with the growing size of the problem.

Kongsomsaksakul, Chen, and Yang (2005) propose a bilevel model in which the upper level (representing the planner) chooses the number of shelters and their locations from among a given set of potential locations with the objective to minimize the total evacuation time. The lower level (representing evacuees) is a combined distribution and assignment (CDA) which is formulated with a principle that evacuees try to travel to safe shelter with the least travel time. They also use the travel time developed by BPR. Realizing the difficulty of solving the bilevel programming problems analytically, they develop a genetic algorithm (GA) to solve the problem. A similar approach can be found in Ng, Park, and Waller (2010) in which the lower level problem is a deterministic user equilibrium (DUE) (Sheffi (1985)) in which the evacuees find their shortest routes to the shelters they are assigned to.

For the evacuees who depend on transit vehicles, Goerigk, Grün, and Heßler (2014) develop an integer programming model to find locations for optimal shelters along with the schedule of buses from pick-up locations to shelters. Realizing the NP-completeness of the problem, they develop a branch-and-price approach to solve the problem. Considering the movement of transit-dependent evacuees and evacuees having individual vehicles, Goerigk, Deghdak, and Heßler (2014) present a comprehensive evacuation plan. They formulate the problem as a multicommodity flow, multicriteria mixed-integer programming problem with the objectives to minimize the number of used shelters, evacuation time, and the total risk exposure of evacuees. Given a risk value to each arc, the total risk value is the sum the product

*Corresponding author's e-mail: hari672@gmail.com

http://doi.org/10.6886/IJOR.201812_15(4).0003

of number of people moving on the arc and risk value of the arc. They propose a heuristic solution procedure based on Genetic Algorithms (GA) because the mixed-integer programming problem is not likely to be solvable in sufficient time for real-world instances.

Based on network flow approach, Heßler and Hamacher (2016) present a mixed integer programming model in which given a supply at each node of a network, and cost of opening a shelter at a node, they minimize the total cost of the opened shelter. They present exact algorithms for choosing single/multiple sink(s), in cases when flows originating at different nodes do not add up on edges, and sinks are uncapacitated, approximation algorithms for the cases when flows originating at different nodes add up on edges and on the chosen sinks as well.

In case of emergency evacuation, since the traffic flow is directed towards the safe areas, the road segments on the direction of paths towards the danger areas (sources) are little occupied or empty while the segments in the opposite direction are over-occupied. Contraflow problems focus on addressing such situations, mathematically, to optimize traffic flow with the ideal direction of arcs in the transportation network.

To solve a multi-source, multi-sink contraflow configuration problem, Kim, Shekhar, and Min (2008) present algorithms based on greedy heuristic and bottleneck relief heuristic. Wang, Wang, Zhang, Ip, and Furuta (2013) considered a multi-objective optimization model to minimize the evacuation time of different categories of evacuees from a single source to appropriate (multiple) shelters and to minimize the traffic set-up time to reverse traffic flow. Their algorithm uses discrete version of particle swarm optimization (PSO) metaheuristic. Zhao, Feng, Li, and Bernard (2016) propose a bi-level model in which a tabu search algorithm is applied to find an optimal lane reversal plan in the upper-level, and the lower-level utilizes a simulated annealing algorithm for lane-based route plans with intersection crossing conflict elimination.

Apart from heuristic techniques, there are available exact analytical algorithms also. Rebennack, Arulselvan, Elefteriadou, and Pardalos (2010) introduce efficient algorithms to solve the maximum dynamic contraflow and the quickest contraflow problems in a single source single sink network. T. Dhamala and Pyakurel (2013) present strongly polynomial time algorithm for the earliest arrival (also known as universally maximum) contraflow problem on a series parallel network. Algorithms to solve the lexicographically maximum contraflow problem on multi-source-multi-sink network and the earliest arrival contraflow problem on multi-source-single-sink network are presented in Pyakurel and Dhamala (2015). Pyakurel and Dhamala (2017b) present a pseudo-polynomial time algorithm to solve the problem on two terminal general network. These problems are solved for discrete time set up. The discrete time algorithms are converted to continuous time algorithms by Pyakurel and Dhamala (2016, 2017a) using the idea of natural transformations in Fleischer and Tardos (1998). Algorithms for contraflow approach for quickest flow with constant and load dependent time on arcs can be found in Pyakurel, Nath, and Dhamala (2018a). Realizing the need of saving capacities of arcs for other facilities during evacuation, Pyakurel, Nath, and Dhamala (2018b) apply partial contraflow technique in a network with path reversal capabilities.

The paper is organized as follows. In Section 2, we present the necessary network flow models on which our approach is based. We introduce optimal sink models and solution procedures in Section 3. Section 4 deals with the identification of optimal sinks allowing arc reversals, and Section 6 concludes the paper.

2. BASIC IDEAS

In this section, we present basic mathematical tools used for developing models and solution algorithms in this work. For modeling, we consider road segments as arcs and their intersections as nodes of a directed network. Anything that moves on arcs is referred to as flow. The amount of flow that can enter a road segment per unit time is the capacity of the arc and the time the flow takes to travel from one end to the other end of the road segment represents the travel time on the corresponding arc.

2.1 Static and dynamic flows

Let $N = (V, A)$ be a directed network where V is the set of nodes and A is the set of arcs with $|V| = n$ and $|A| = m$. For a node $i \in V$, we denote the set of outgoing arcs from i by $A_i^{\text{out}} = \{e \in A : e = (i, j) \text{ for some } j \in V\}$, and the set of incoming arcs to i by $A_i^{\text{in}} = \{e \in A : e = (j, i) \text{ for some } j \in V\}$. For each $e \in A$, b_e, τ_e denote the upper capacity (or capacity), and travel time on e. A flow travels from a special $S \subset V$, called source nodes to another $D \subset V$, called sink nodes. We represent such a network by $N = (V, A, b, \tau, S, D)$. If $S = \{s\}, D = \{d\}$, we write $N = (V, A, b, \tau, s, d)$.

A static s-d flow $x : A \to \mathbb{R}_{\geq 0}$ is a function of non-negative values such that $x(e) = x_e$ satisfying the following.

$$\sum_{e \in A_i^{\text{out}}} x_e - \sum_{e \in A_i^{\text{in}}} x_e = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{if } i \in V \setminus \{s, d\} \end{cases} \tag{1}$$

$$0 \leq x_e \leq b_e \tag{2}$$

The quantity v is the value of the static flow x which is the flow that goes out of the source s and reaches the sink d. The third set of constraints in (1) states that whatever flow comes to a node other than the source or the sink goes out of it. The constraints set (2) restricts the flow in any arc exceed its capacity. The maximum static flow problem seeks to maximize v under the constraints (1) - (2) for the solution of which there are available highly efficient algorithms because of the special structure of the linear program stated above. For more details, see T. N. Dhamala, Pyakurel, and Dempe (2018).

A dynamic flow x^{dyn} with time horizon T consists of a Lebesgue- integrable functions $x_e^{\text{dyn}} : [0, T) \to \mathbb{R}_{\geq 0}$ for each arc $e \in A$ with $x_e^{\text{dyn}}(t) = 0$ for $t \geq T - \tau(e)$ and satisfies the following.

$$\sum_{e \in A_i^{\text{in}}} \int_0^{t-\tau(e)} x_e^{\text{dyn}}(\theta)d\theta - \sum_{e \in A_i^{\text{out}}} \int_0^{t} x_e^{\text{dyn}}(\theta)d\theta \quad \geq \quad 0, \ \forall t \in [0, T), \forall i \in V \setminus \{s, d\} \tag{3}$$

$$\sum_{e \in A_i^{\text{in}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta)d\theta - \sum_{e \in A_i^{\text{out}}} \int_0^{T} x_e^{\text{dyn}}(\theta)d\theta \quad = \quad 0, \ \forall i \in V \setminus \{s, d\} \tag{4}$$

$$0 \leq x_e^{\text{dyn}}(t) \quad \leq \quad b_e, \ \forall e \in A, \forall t \in [0, T). \tag{5}$$

$x_e^{\text{dyn}}(\theta)$ is the rate of flow that enters arc e at time θ. The value of the dynamic flow x^{dyn} for the time horizon T is:

$$\begin{aligned} \text{val}_T(x^{\text{dyn}}) &= \sum_{e \in A_d^{\text{in}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta)d\theta - \sum_{e \in A_d^{\text{out}}} \int_0^{T} x_e^{\text{dyn}}(\theta)d\theta \\ &= \sum_{e \in A_s^{\text{out}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta)d\theta - \sum_{e \in A_s^{\text{in}}} \int_0^{T} x_e^{\text{dyn}}(\theta)d\theta \end{aligned} \tag{6}$$

For a given T, the maximum dynamic flow problem seeks to maximize $\text{val}_T(x^{\text{dyn}})$ under the constraints (3) - (5). The constraints (3) state that there can be a holdover of the flow in the intermediate nodes which ultimately gets cleared by the time horizon T as stated in (4). The constraints in (5) do not allow the flow value to exceed the upper capacity at any time. For more details, we refer to Skutella (2009). If we replace each of the integral by the corresponding summation over the discretized set of times $\{0, \ldots, T\}$, we get the discrete-time version of the problem.

3. IDENTIFICATION OF THE OPTIMAL SINK

In a network with a single source, given a set of feasible sink nodes, we take a straightforward approach for the choice of a single sink depending on the objective of the evacuation problem. If the objective is to send as much flow as possible, we choose the sink which maximizes flow value. Likewise, when a given flow value is given, we choose the sink so as to minimize the time for the flow to reach the sink.

3.1 Optimal sink to maximize the flow value

Definition 1 (MaxStatic sink, MaxDynamic sink)**.** Let $N = (V, A, b, \tau, s)$ be a network with a set of feasible sinks $D \subset V \setminus \{s\}$ and let $v_{stat}(d), v_{dyna}^T(d)$ denote the value of maximum static flow, and maximum dynamic flow with time horizon T, respectively, from s to $d \in D$. We call the node $d_{stat} = \arg\max_{d \in D}\{v_{stat}(d)\}$, the MaxStatic sink and $d_{dyna}^T = \arg\max_{d \in D}\{v_{dyna}^T(d)\}$, the MaxDynamic sink.

Example 1. We consider an evacuation network in Figure 1. Let the source node be s and the set of feasible sinks $D = \{d_1, d_2, d_3\}$. If node d_1 is taken as the sink, the maximum static flow value is 6 (4 via path $s - d_1$, 1 each via $s - d_2 - d_1$, and $s - d_2 - d_3 - d_1$). The dynamic flow value with 2 as the sink and discrete time horizon $T = 3$ is 2 (1 via $s - d_2 - d_1$ twice). The values corresponding to other sinks and time horizons are listed in the following table. So, one can easily conclude that $d_{stat} = 4$ and $d_{dyna}^3 = 3, d_{dyna}^8 = 2, d_{dyna}^{12} = 4$.

d	$v_{stat}(d)$	$v_{dyna}^3(d)$	$v_{dyna}^8(d)$	$v_{dyna}^{12}(d)$
d_1	6	2	30	54
d_2	4	9	28	44
d_3	7	1	28	56

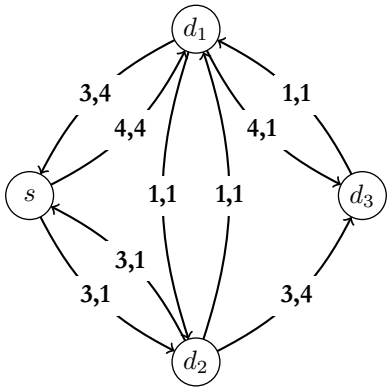


Figure 1: Evacuation Network with arc labels capacity, travel time

Now, we present mathematical programming formulation to identify the MaxStatic sink. We consider a network $N = (V, A, b, \tau, s)$ with a set of feasible sinks D. For the modeling purpose, we consider that there is only one arc incoming to each $d \in D$. This does not restrict the general case because we can always add a node d' to V, and (d, d') to A such that $\tau(d, d') = 0, b(d, d') = \infty$ and replace $d \in D$ by d' so that d is identified with d'. Practically, infinite capacity of (d, d') can be replaced by $\sum_{e \in A_d^{\text{in}}} b_e$. Figure 2 shows such a network transformation in which D becomes $\{d'_1, d'_2, d'_3\}$.

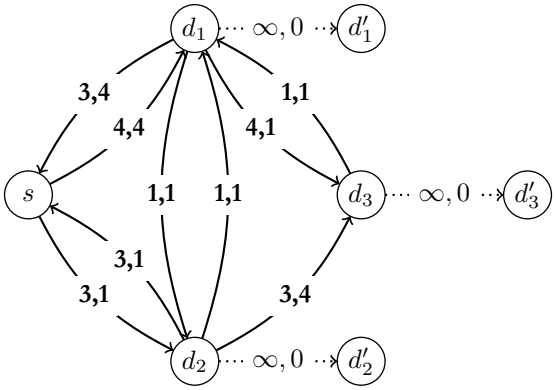


Figure 2: Transformation of the network in Figure 1

Let $A' = \{(i, d) \in A : d \in D\}$. We present the problem of finding MaxStatic sink as a mixed binary integer formulation as:

$$\max \quad \sum_{e \in A'} x_e \tag{7a}$$

$$\sum_{e \in A_i^{\text{out}}} x_e - \sum_{e \in A_i^{\text{in}}} x_e = 0, \ \forall i \in V \setminus (\{s\} \cup D) \tag{7b}$$

$$x_e \leq b_e, \ \forall e \in A \setminus A' \tag{7c}$$

$$x_e \leq b_e y_e, \ \forall e \in A' \tag{7d}$$

$$\sum_{e \in A'} y_e = 1 \tag{7e}$$

$$y_e \in \{0, 1\} \tag{7f}$$

The objective (7a) maximizes the flow entering to the sinks. Because of (7d)-(7f), the flow will be directed towards only one sink. Constraints (7b) are mass balance constraints in the intermediate nodes. (7c),(7d) are capacity constraints. The constraint (7e) chooses only one sink out of feasible sinks.

In the constraint set (7d), since b_e may not be in $\{-1, 0, +1\}$, the matrix associated with the constraints is not totally unimodular. Thus the linear programming relaxation of the above integer programming may not give the integral solution. However, if $y_e, e \in A'$ is bound to be binary, we have the following observation.

Theorem 1. If $b_e \in \mathbb{Z}$, the mixed integer programming (7a)-(7f) has all integral solutions.

Proof. Let $I^{|D|}$ denote the set of the column vectors of the $|D| \times |D|$ identity matrix. Because of our assumption, indegree of each $d \in D$ is 1, and hence the solution set of the constraints (7e) and (7f) is $I^{|D|}$. If $y = [y_e : e \in A']$, then the column vector y has exactly $|D|$ components. If a fix $y \in I^{|D|}$ is chosen, the mixed integer programming problem (7) becomes a linear programming problem. Since $|I^{|D|}| = |D|$, MIP (7) can be solved by solving linear program (7a-7d) at most $|D|$ times.

Further, let the matrix equation $Mx = 0$ represent the constraints (7b). If e has both of its ends in $V \setminus (\{s\} \cup D)$, then the column of M corresponding to x_e will have exactly two entries $+1, -1$. Each of the other columns of M corresponding to x_e with an end of e in $\{s\} \cup D$ will have exactly one entry either 1 or -1. This shows that M is totally unimodular. Since the polyhedron $\{x \in \mathbb{R}^n : Mx = b, 0 \leq x \leq u\}$ with totally unimodular M and integer u is an integral polyhedron, we can get all integer solutions of the mixed integer programming (7) if $b_e \in \mathbb{Z}, \forall e \in A$. □

The above is particularly important, because most of the network flow algorithms use integrality of flow to develop efficient algorithms. We present a straight-forward procedure to identify MaxStatic sink in Algorithm 1 which iteratively chooses an element $d \in D$, finds the maximum static s-d flow value, and selects d as the MaxStaic sink if the flow-value is improved. When the iteration ends, the algorithm returns the MaxStatic sink and the corresponding static flow.

Algorithm 1: Locating the MaxStatic sink

Input : Directed network $N = (V, A, b, s)$, the set of possible sink locations D
Output: Optimal sink d^*, the corresponding static flow x

```
1  curr_max_v = −1
2  for d ∈ D do
3      new_max_v = v_stat(d)
4      if new_max_v > curr_max_v then
5          d* = d
6          curr_max_v = new_max_v
7          x = corresponding static flow
8      end
9  end
10 return d*, x
```

The practical running time of Algorithm 1 can be improved by finding $v_{stat}(d)$ in Line 3 only if $\sum_{e \in A_d^{\text{in}}} b_e > curr\_max\_v$ in the network without the transformation mentioned in Figure 2, and exiting the **for** loop and returning d as d^* if $v_{stat}(d) = \sum_{e \in A_s^{\text{out}}} b_e$.

Algorithm 1 leads to the following important implication.

Theorem 2. The problem of identifying MaxStatic sink can be solved with strongly polynomial time complexity of $O(nm|D|)$.

Proof. Let C be the complexity of a maximum static flow calculation. Since Algorithm 1 terminates after $|D|$ iterations and Line 3 calculates a maximum static flow value in each iteration, the complexity of the algorithm is $O(C|D|)$. Using the algorithm of Orlin (2013), the maximum static flow calculation can be done in $O(nm)$ time, where n, m denote the number of nodes and number of arcs in N. Hence, Algorithm 1 solves the problem in $O(nm|D|)$ time. This proves the assertion. □

Given a time horizon T, Ford and Fulkerson (1962) showed that for a dynamic maximum flow problem, the maximum flow can be obtained by temporally repeating the static flow. 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 (Fleischer and Tardos (1998); Skutella (2009))**.** 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 $Tv - \sum_{e \in A} \tau_e x_e$.

Using the result in Lemma 1, we can replace the objective (7a) by $\sum_{e \in A'} Tx_e - \sum_{e \in A} \tau_e x_e$ to obtain the MaxDynamic sink, in continuous time setting. In the discrete time setting, we can just replace T by $T + 1$. The result analogous to that in Theorem 1 is valid in this case also.

Replacing $v_{stat}(d)$ by $v_{dyna}^T(d)$ in Line 3, we can adapt Algorithm 1 to identify the MaxDynamic sink.

Theorem 3. The problem of identifying MaxDynamic sink can be solved with strongly polynomial time complexity of $O((m \log n)(m + n \log n)|D|)$.

Proof. For a given sink $d \in D$, let N' be the network obtained by adding an arc (d, s) to N with $b(d, s) = \infty, \tau(d, s) = -T$ in a continuous-time setting (in the discrete time setting, we take $\tau(d, s) = -(T + 1)$). Assuming that there is no directed path from d to s in N, let x be the the min-cost circulation in N', then as a result of Lemma 1, the restriction of x to N is the temporally repeated static flow, with value v, corresponding to the maximum dynamic flow and $v^T_{dyna}(d) = Tv - \sum_{e\in A} \tau_e x_e$ (in discrete-time setting, $v^T_{dyna}(d) = (T + 1)v - \sum_{e\in A} \tau_e x_e$). As minimum cost flow problem, using enhanced capacity scaling technique, can be solved in $O((m \log n)(m+n \log n))$ time (Orlin (1993)), and the MaxDynamic sink can be identified iterating over elements of D, we get the required result. □

3.2 Optimal sink to minimize the quickest time

Given supply F at s, the quickest flow problem seeks to find the dynamic s-d flow which has the value F within the minimum time horizon. The following result is crucial in this regard.

Theorem 4 (Lin and Jaillet (2015))**.** The quickest flow problem in $N = (V, A, b, \tau, s, d)$ with a given supply at s can be formulated as the following linear programming problem

$$\min \quad \frac{F + \sum_{e\in A} \tau_e \cdot x_e}{v} \tag{8a}$$

$$\sum_{e\in A^{\text{out}}_i} x_e - \sum_{e\in A^{\text{in}}_i} x_e = \begin{cases} v, & \text{if } i = s \\ -v, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \tag{8b}$$

$$0 \le x_e \le b_e \; \forall e \in A \tag{8c}$$

Proof. Let x be a feasible static s-d flow with value v. Then the value of the corresponding dynamic flow with time horizon T is $Tv - \sum_{e\in A} \tau_e x_e$ (see Lemma 1). So, we can formulate the maximum dynamic flow problem with a time horizon T as:

$$\max \quad Tv - \sum_{e\in A} \tau_e x_e \tag{9a}$$

$$\sum_{e\in A^{\text{out}}_i} x_e - \sum_{e\in A^{\text{in}}_i} x_e = \begin{cases} v, & \text{if } i = s \\ -v, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \tag{9b}$$

$$0 \le x_e \le b_e \; \forall e \in A \tag{9c}$$

Let $\psi(T) = \max Tv - \sum_{e\in A} \tau_e x_e$ under constraints (9b)-(9c). Then $\psi(T)$ is non-decreasing function of T (Burkard, Dlaska, and Klinz (1993)). So, the quickest flow problem with a given supply F is equivalent to finding the minimum time T^* such that $\psi(T^*) \ge F$. Hence, the quickest flow problem can be formulated as:

$$\min \quad T \tag{10a}$$

$$Tv - \sum_{e\in A} \tau_e x_e \ge F \tag{10b}$$

along with (9b)-(9c). The constraint (10b) can be reshuffled as:

$$T \ge \frac{F + \sum_{e\in A} \tau_e x_e}{v} \tag{11}$$

in which $v \neq 0$ because if $v = 0$, no flow can be sent from the source to the sink. Since (9b)-(9c) do not involve T, we can replace T in (10a) with $\frac{F+\sum_{e\in A} \tau_e x_e}{v}$ and the result follows. □

Given a source s and a supply F, if a sink is to be chosen from a given set of feasible sets, then we take a point of view to choose the sink which minimizes the quickest time.

Definition 2 (Quickest sink)**.** Let $N = (V, A, b, \tau, s)$ be a network with a set of feasible sinks $D \subset V$ and let $T^F(d)$ denote the quickest time to transport the flow value of F from s to $d \in D$. We call the node $\arg\min_{d \in D}\{T^F(d)\}$, the Quickest sink.

We can adapt the mathematical formulation given in (7), replacing its objective by (8a) where $v = \sum_{e \in A'} x_e$. To linearize the objective (8a), we can introduce a new variable $x'_e = x_e/v$ and adjust the constraints accordingly converting the problem into mixed binary integer linear program.

We present a simple procedure to identify the quickest sink in Algorithm 2. The algorithm iteratively chooses an element $d \in D$, finds the quickest time to send F from s to d, and selects d as the Quickest sink if the quickest time decreases. When the iteration ends, the algorithm returns the Quickest sink and the static flow corresponding to the temporally repeated quickest flow.

Algorithm 2: Locating the Quickest sink

Input : directed network $N = (V, A, b, s, \tau)$, supply F at s, the set of possible sink locations D
Output: optimal sink d^*, the corresponding static flow x

```
curr_quickest_time = ∞
for d ∈ D do
    new_quickest_time = T^F(d)
    if new_quickest_time < curr_quickst_time then
        d* = d
        curr_quickest_time = new_quickest_time
        x = corresponding static flow
    end
end
return d*, x
```

On the basis of Algorithm 2, we have the following result.

Theorem 5. The Quickest sink can be computed in strongly polynomial time complexity of $O(nm^2 \log^2 n)|D|$.

Proof. There are $|D|$ iterations in Algorithm 2 in each of which it calculates the quickest flow. Saho and Shigeno (2017) adapted cancel-and-tighten algorithm of finding minimum cost flow to compute quickest flow, which has the complexity of $O(nm^2 \log^2 n)$. Hence, Quickest sink can be identified in $O(nm^2 \log^2 n|D|)$ time. □

4. OPTIMAL SINK WITH ARC REVERSAL STRATEGY

4.1 Contraflow approach

The contraflow approach reverses the direction of necessary arcs to optimize the flow in a directed network. The common procedure to do so is to solve the corresponding problem in, what is known as, an auxiliary network. The auxiliary network of $N = (V, A, b, \tau, s, d)$ is $\bar{N} = (V, \bar{A}, \bar{b}, \bar{\tau}, s, d)$ where

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

and for each $(i, j) \in \bar{A}$,

$$\begin{aligned} \bar{b}(i, j) &= b(i, j) + b(j, i) \\ \bar{\tau}(i, j) &= \begin{cases} \tau(i, j) & \text{if } (i, j) \in A \\ \tau(j, i) & \text{otherwise} \end{cases} \end{aligned}$$

in which we consider $b(i, j) = 0$ whenever $(i, j) \notin A$, and vice versa. Figure 3 is the auxiliary network constructed for the network in Figure 1.

4.2 Optimal sink with contraflow

Because of the change in capacity of the arcs, the decisions related to optimal sink also change if we apply contraflow approach. Example 2 illustrates this fact.

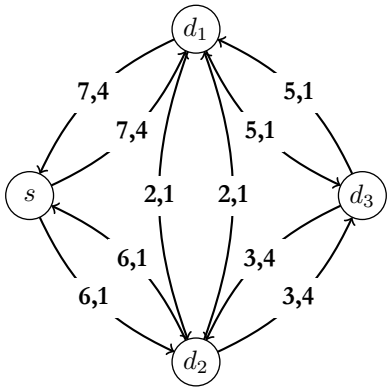


Figure 3: Auxiliary network of the network in Figure 1

Example 2. Consider the network considered in Example 1. To allow arc reversal, we construct the auxiliary network as depicted in Figure 3. Taking s as source, if $d = d_1$, the maximum static flow value is 12 (7 via $s - d_1$, 3 via $s - d_2 - d_3 - d_1$, 2 via $s - d_2 - d_1$). The corresponding values with $d = 3, 4$ are 11 and 8, respectively. Thus MaxStatic sink allowing lane reversals is the node 2 while it is node 4 without allowing arc reversals with maximum static flow value 7 (see Example 1). Considering 2 as the sink, the maximum static flow x allowing arc reversals is given in the following table.

e	(s,d_1)	(s,d_2)	(d_1,s)	(d_1,d_3)	(d_1,d_2)	(d_2,s)	(d_2,d_1)	(d_2,d_3)	(d_3,d_1)	(d_3,d_2)
$x(e)$	7	5	0	0	0	0	2	3	3	0

Since $x(s,d_1) > b(s,d_1), x(s,d_2) > b(s,d_2), x(d_2,d_1) > b(d_2,d_1), x(d_3,d_1) > b(d_3,d_1)$, the arcs to be reversed are: $(d_1,s), (d_2,s), (d_1,d_2)$ and (d_1,d_3).

Given a network $N = (V, A, b, \tau, s)$ and set of feasible sinks D, to identify the MaxStatic sink, MaxDynamic sink, Quickest sink allowing arc reversals, we solve the corresponding problem in $\bar{N}$. We present Algorithm 3 in which the input is $N = (V, A, b, \tau, s)$ and a set of feasible sinks D. For the calculation of the quickest sink, the supply F at the source s is also be given. In Line 1, the the auxiliary network $\bar{N} = (V, \bar{A}, \bar{b}, \bar{\tau}, s)$ is constructed. In Line 2, depending on the objective, MaxStatic sink, MaxDynamic sink, or Quickest sink is identified as described in Section 3. The corresponding static flow x is also calculated. In Line 3, x is decomposed into paths and cycles and cycle flows are removed so that the set of arcs to be reversed in Line 4 is well-defined. After the removal of cycle flows, if an arc in A has flow value more than its original capacity, its opposite arc will be reversed. If the solution shows any positive flow in any arc not in A, then the corresponding opposite arc is also reversed. In this way, Line 4 gives the set of arcs to be reversed.

Algorithm 3: Locating sink with contraflow

Input : directed network $N = (V, A, b, \tau, s)$, the set of possible sink locations D (and supply F at the source s in case of Quickest sink) with arc reversal capability

Output: optimal sink d^*, set of arcs to be reversed, corresponding static flow x

1 Construct the auxiliary network $\bar{N}$.
2 Solve the corresponding problem in $\bar{N}$ to find the optimal sink d^*, and the corresponding static flow x.
3 Decompose x into paths and cycles and remove cycles.
4 $R = \{(j,i) \in A : x(i,j) > b(i,j)$ if $(i,j) \in A$ or $x(i,j) > 0$ if $(i,j) \notin A\}$
5 **return** d^*, R, x.

Theorem 6. The problems of identification of MaxStatic sink, MaxDynamic sink, and Quickest sink allowing arc reversals can be solved in strongly polynomial time with the complexity of the corresponding problems without allowing lane reversals.

Proof. In Algorithm 3, the auxiliary network, in Line 1, can be constructed in $O(m)$ times. Line 2 finds MaxStatic sink, MaxDynammic sink, or Quickest sink in the auxiliary network depending on the problem. So, the complexity of Line 2 is $O(C|D|)$ where C is

(i) $O(mn)$, the complexity of the maximum static flow calculation, in case of the MaxStatic sink

(ii) $O((m \log n)(m + n \log n))$, the complexity of the minimum cost flow calculation, in case of the MaxDynamic sink

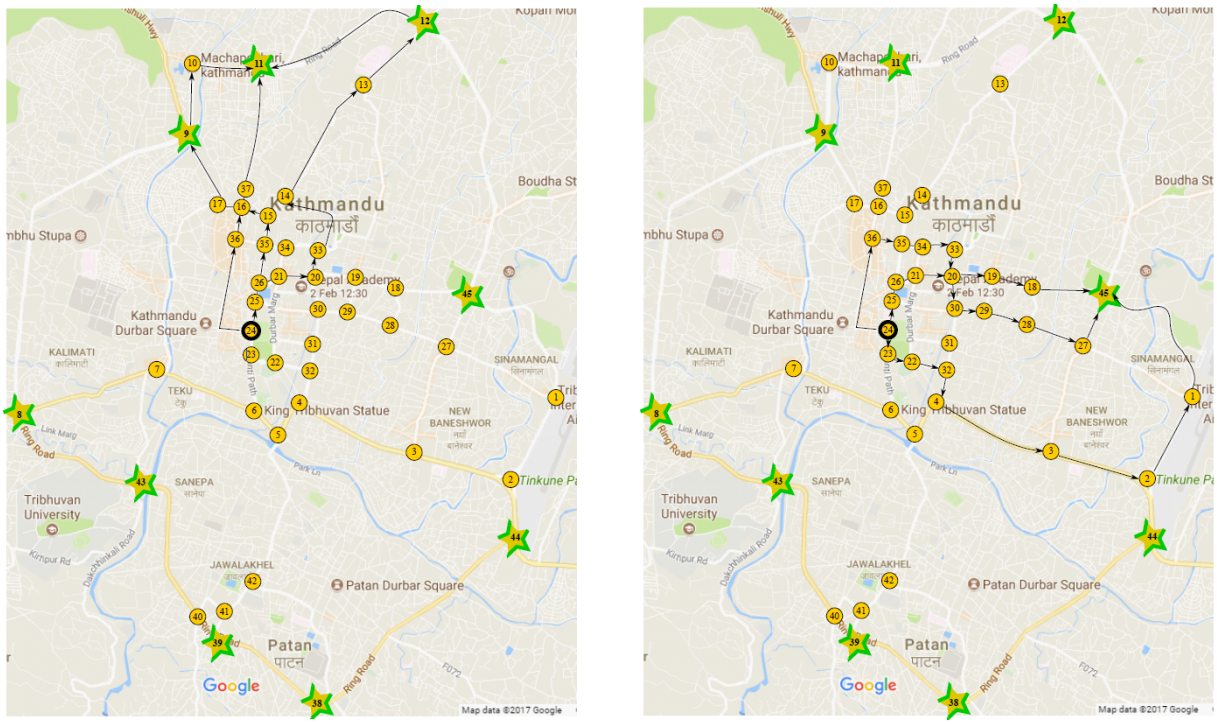
(iii) $O(nm^2 \log^2 n)$, the complexity of quickest flow calculation, in case of the Quickest sink

The decomposition of a static flow into paths and cycles can be done in $O(nm)$ time (see Ahuja, Magnanti, and Orlin (1993)). So the time complexity of Line 3 in Algorithm 3 is $O(mn)$. The construction of R in Line 4 requires $O(m)$ comparisons. Hence, the overall complexity of the algorithm is dominated by the complexity of Line 2, which is $O(C|D|)$. This proves the assertion.

□

5. CASE ILLUSTRATION

As an illustration, we consider Kathmandu road network within Ring Road only with major road segments (cf. Figure 4). We take the node adjoining Tundikhel area (denoted in the figure by 24) as the source and Kalanki (8), Balaju (9), Gongabu (11), Narayan Gopal Chowk (12), Gaushala (45), Koteshwar Jadibuti (44), Satdobato (38), Ekantakuna (39), and Balkhu (43) as possible sinks. To implement auto-based evacuation planning, we take the capacities of arcs between 2 cars per second to 4 cars per second depending on the width of the segment. The direction of the usual traffic flow is taken as the direction of the arc. The time related to an arc is taken using Google Maps.



(a) (b)

Figure 4: Kathmandu network with (a) Node 11 as MaxDynamic sink (b) Node 45 as MaxDynamic sink with contraflow

Considering a time horizon $T = 1$ hour, we find the Maxdynamic sink as Gongabu(11) with the dynamic flow value of 12,120 cars. However, if we allow arc reversal, the Maxdynamic sink is Gaushala(45) with the corresponding flow value 27,360 cars. If we take a time horizon of $T = 2$ hours, the sink locations with and without arc reversal are respectively the same as those of $T = 1$ hour with flow values 33,720 and 70,560.

The Quickest sink locations with different values of F are listed in the following table.

F	Without contraflow		With contraflow	
	Quickest sink	Quickest time (sec)	Quickest sink	Quickest time (sec)
1,000	Balaju (9)	1,160	Balaju (9)	910
10,000	Gongabu (11)	3,247	Gaushala(45)	2,150
20,000	Gongabu (11)	4,913	Gaushala(45)	2,987

6. CONCLUSION

In this work, the problem of identification of an optimal sink from among a given set of uncapacitated sinks is considered with and without allowing arc reversals, with different aspects of network flow, viz. maximum static flow, maximum dynamic flow, and quickest flow. Presenting the solution procedures, it is proved that the problems can be solved in strongly polynomial time. The problem considered is particularly important in case of evacuation planning when a single shelter has to be chosen from among given shelters of sufficient capacities. One can use MaxStatic sink, when a maximum number of evacuees are to be sent as one wave, MaxDynamic sink, when the maximum number of evacuees are to be sent within a given time horizon, and Quickest sink when a given number of evacuees are to be evacuated as quickly as possible.

ACKNOWLEDGMENTS

The authors acknowledge partial supports of Alexander von Humboldt Foundation (AvH) and German Academic Exchange Service (DAAD). The first author acknowledges the support of University Grants Commission (UGC) Nepal also for granting PhD fellowship and research support.

REFERENCES

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. Prentice Hall.

Bureau of Public Roads. (1964). *Traffic assignment manual*. Washington D.C.: US Department of Commerce, Urban Planning Division.

Burkard, R. E., Dlaska, K., & Klinz, B. (1993). The quickest flow problem. *ZOR - Meth. & Mod. of OR*, *37*(1), 31–58. Retrieved from `https://doi.org/10.1007/BF01415527`

Dhamala, T., & Pyakurel, U. (2013). Earliest arrival contraflow problem on series-parallel graphs. *International Journal of Operations Research*, *10*(1), 1–13.

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

Fleischer, L., & Tardos, E. (1998). Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, *23*(3), 71–80.

Ford, F., & Fulkerson, D. (1962). *Flows in networks*. New Jersey: Princeton University Press.

Goerigk, M., Deghdak, K., & Heßler, P. (2014). A comprehensive evacuation planning model and genetic soution algorithm. *Transportation Research, Part E*, *71*, 82–97.

Goerigk, M., Grün, B., & Heßler, P. (2014). Combining bus evacuation with location decisions: A branch-and-price approach. *Transportation Research Procedia*, *2*, 783–791.

Heßler, P., & Hamacher, H. (2016). Sink location to find optimal shelters in evacuation planning. *EURO Journal on Computational Optimization*, *4*(3-4), 325–347.

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

Kongsomsaksakul, S., Chen, A., & Yang, C. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, *6*, 4237–4252.

Lin, M., & Jaillet, P. (2015). On the quickest flow problem in dynamic networks: a parametric mincost flow approach. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms* (pp. 1343–1356).

Ng, M., Park, J., & Waller, S. (2010). A hybrid bilevel model for the optimal shelter assignment in emergency evacuations. *Computer-Aided Civil and Infrastructure Engineering*, *25*(8), 547–556.

Orlin, J. (1993). A faster strongly polynomial minimum cost flow algorithm. *Operations research*, *41*(2), 338–350.

Orlin, J. (2013). Max flows in $o(nm)$ time, or better. , 765–774.

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

Pyakurel, U., & Dhamala, T. (2016). Continuous time dynamic contraflow models and algorithms. *Advances in Operations Research*.

Pyakurel, U., & Dhamala, T. (2017a). Continuous dynamic contraflow approach for evacuation planning. *Annals of Operations Research*, *253*(1), 573–598.

Pyakurel, U., & Dhamala, T. (2017b). Evacuation planning by earliest arrival contraflow. *Journal of Industrial & Management Optimization*, *13*(1), 489–503.

Pyakurel, U., Nath, H., & Dhamala, T. (2018a). Efficient contraflow algorithms for quickest evacuation planning. *Science China Mathematics*, *61*(11), 2079–2100.

Pyakurel, U., Nath, H., & Dhamala, T. (2018b). Partial contraflow with path reversals for evacuation planning. *Annals of Operations Research*.

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

Saho, M., & Shigeno, M. (2017). Cancel-and-tighten algorithm for quickest flow problems. *Networks*, *69*(2), 179–188.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Englewood Cliffs, NJ: Prentice-Hall.

Sherali, H., Carter, T., & Hobeika, A. (1991). A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B: Methodological*, *25*(6), 439–452.

Skutella, M. (2009). An introduction to network flows over time. In W. Cook, L. Lovász, & J. Vygen (Eds.), *Research trends in combinatorial optimization* (pp. 451–482). Springer.

Wang, J., Wang, H., Zhang, W., Ip, W., & Furuta, K. (2013). Evacuation planning based on the contraflow technique with consideration of evacuation priorities and traffic setup time. *IEEE Transactions on Intelligent Transportation Systems*, *14*(1), 480–485.

Zhao, X., Feng, Z., Li, Y., & Bernard, A. (2016). Evacuation network optimization model with lane-based reversal and routing. *Mathematical Problems in Engineering*.