The Nepali Mathematical Sciences Report, Vol. 39, No. 1, 2022: 14-21
DOI: https://doi.org/10.3126/nmsr.v39i1.46913

MAXSTATIC SINK LOCATION PROBLEM WITH CAPACITATED SINKS

HARI NANDAN NATH

hari672@gmail.com
Tribhuvan University, Bhaktapur Multiple Campus, Bhaktapur, Nepal

Abstract: The Maximum static flow problem, in a single-source-single-sink network, deals with finding the maximum amount of flow from a source to a sink. Given a set of possible sinks, identification of the sink that maximizes the amount of the maximum flow is an important optimization problem. In this work, we consider the problem of identification of a sink when the possible sinks have given capacities. We devise a simple network transformation so that algorithms in the uncapacitated case can be used in the capacitated one proving that the problem can be solved with strongly polynomial time complexity. Further, we propose an algorithm whose average-case complexity is better than that of an iterative procedure that iterates over all the possible sinks.

Key Words: Sink location, static flow, network flow, capacited sinks

AMS (MOS) Subject Classification. Primary: 90B10, 90C27, 68Q25 Secondary: 90B06, 90B20.

1. Introduction

Network flow modeling, has been widely used to solve problems in various fields of inquiry, e.g. engineering, management, applied mathematics, and computer science [1]. Evacuation planning is one of the active research areas in which network flow modeling has been widely used. The survey of such models can be found in [2, 3]. One of current research interests is identification of facility locations using network flow approach [4, 6]. Hamacher et al. [2] consider the problem of identification of facility locations on arcs so that the reduction in the flow is minimum because of the placement the facilities. Dhungana and Dhamala [5] extend their problem to the case where the direction of arcs can be reversed. Nath et al. [6] consider a similar problem in which the increase in the egress time of a given amount of flow, because placement of facilities, is minimum. Likewise, Pyakurel et al.[7, 8] consider saving of capacities of road-segments in emergency evacuations, so that they can be used for some facilities. Nath et al. [9, 10] consider problems of saving a path for facility movements in the opposite direction of the evacuee flow.

One of the interesting problems in network flow approach is the shelter (sink) location problem in which appropriate sinks are chosen from among a given set of possible shelters [11, 12]. In their work, Heßler and Hamacher [11] present a model that choose shelters

Received: May 27, 2022 Accepted: June 25, 2022 Published Online: June 30, 2022.

so as to minimize the opening cost of the shelter. Nath and Dhamala [12] present sink location models based on static and dynamic network flows to choose a single sink with the objectives of maximizing the flow or minimizing the time of evacuation.

In this work, we consider the problem of identification of a sink from among a set of sinks of given capacities so that the value of the maximum flow is the greatest and modify the algorithm given in [12]. The outline of the paper is as follows: Section 2 gives the basic ideas about the maximum static flow problem and MaxStatic sink location problem, Section 3 extends the ideas to the capacitated sinks case and presents algorithms to solve it, and Section 4 concludes the paper.

2. Preliminaries

2.1. **Maximum static flow problem.** In network flow modeling, a network is a directed graph in which something called flow moves from some nodes called sources to some other nodes called sinks via arcs or edges with non-negative capacities.

A network $N = (V, A)$ consists of a set V of nodes, and a set $A \subseteq V \times V$ of directed arcs. We assume that $|V| = m$ and $|A| = n$. For each $i \in V$, we define

$$V_a(i) = \{j \in V : (i, j) \in A\}$$

which denotes the set of nodes that succeed i (the nodes after i). and

$$V_b(i) = \{j \in V : (j, i) \in A\}$$

which denotes the set of nodes that precede i (the nodes before i). Associated with each arc (i, j) is a capacity $u_{ij} \geq 0$ denoting the maximum amount of flow that can flow on (i, j). Given two special nodes: a source node s and a sink node t, a vector $x = \{x_{ij}\}$ satisfying the mass balance constraints

$$\sum_{j \in V_a(i)} x_{ij} - \sum_{j \in V_b(i)} x_{ji} = 0 \ \forall i \in V - \{s, t\} \tag{2.1}$$

and the capacity constraints

$$0 \leq x_{ij} \leq u_{ij} \ \forall (i, j) \in A \tag{2.2}$$

is known as a *static s-t flow*. The value of x is defined by

$$v(x) = \sum_{j \in V_a(s)} x_{sj} - \sum_{j \in V_b(s)} x_{js} = - \sum_{j \in V_a(t)} x_{tj} + \sum_{j \in V_b(t)} x_{jt} \tag{2.3}$$

The flow x, that maximizes $v(x)$ is known as a *maximum static flow*.

So, the maximum static flow problem can be stated as [1]

$$\max \ v \tag{2.4}$$

subject to

$$\sum_{j \in V_a(i)} x_{ij} - \sum_{j \in V_b(i)} x_{ji} = \begin{cases} v, & i = s \\ 0, & i \in V \setminus \{s, t\} \\ -v, & i = t \end{cases} \tag{2.5}$$

The maximum static flow problem is a well-studied problem. Following the labeling algorithm with complexity $O(mnU)$, where $U = \max\{u_{ij} : (i,j) \in A\}$, by Ford and Fulkerson [13], there are several improvements of the algorithm. For example, the capacity scaling algorithm augments flows along paths with sufficiently large residual capacity and runs in $O(mn \log U)$ time, the shortest augmenting path algorithm, which augments the flows along shortest augmenting paths, runs in $O(mn^2)$ time [14]. There are preflow-push algorithms which relax the flow conservation constraints at the intermediate steps, seek out the shortest paths, but send flows on individual arcs from active nodes (nodes with positive excess flow) rather than on an s–t paths. The first preflow-push algorithm with running time $O(n^3)$ is due to Karzanov [15]. A generic pre-flow push algorithm runs in $O(mn^2)$ time [17]. Among the several specific implementations of the algorithm, FIFO (first in first out) preflow-push algorithm [16] runs in $O(n^3)$ time, highest-label perflow push algorithm [17] runs in $O(n^2\sqrt{m})$ time, and the excess scaling algorithm [18], runs in $O(mn + n^2 \log U)$ time. Among the various improvements in the algorithms to solve maximum static flow problem, the algorithm by Orlin [19], solves the maximum flow problem in $O(mn)$ time if $m < n^{1.06}$.

2.2. **MaxStatic Sink Location Problem.** If there are more than one possible sinks, and one sink is taken at a time, the value of the maximum flow will depend on the choice of the sink. If we have to choose only one sink, a natural idea is to choose the one that maximizes the maximum flow.

Definition 2.1 (MaxStatic sink [12])**.** Consider a set of possible sinks $T \subset V$ such that $s \notin T$. Let v_t be the value of a maximum static s-t flow. The MaxStatic sink is defined as $\arg\max_{t \in T}\{v_t\}$.

A simple iterative procedure given in Algorithm 1 can be used to identify the MaxStatic sink. This makes easy to realize that the MaxStatic sink problem can be solved in a strongly polynomial time.

Algorithm 1 Identifying the MaxStatic sink [12]

Input: Network $N = (V, A)$ with a set of possible sinks $T \subset V$.
Output: MaxStatic sink

```
max_f = −1
for t ∈ T do
    Calculate the maximum flow value v_t
    if v_t > max_f then
        MaxStatic sink = t
        max_f = v_t
    end if
end for
return MaxStatic sink
```

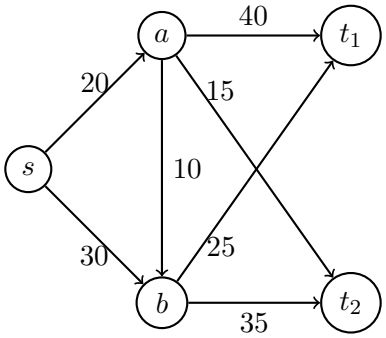


FIGURE 1. Example Network

3. MaxStatic Sink Location Problem with Capacitated Sinks

In the problem considered in Section 2.2, the possible sinks are considered to have infinite capacity (i.e. each sink has the capacity to hold any amount of flow that can reach the sink).

Example 3.1. Consider an example network given in Figure 1. Each arc label represents the capacity of the arc. The node s is the source and $T = \{t_1, t_2\}$ is the set of possible sinks. Suppose that we have to choose one sink out of them. If t_1 is considered as the sink, the value of the maximum flow is 45 (20 via s–a–t_1, and 25 via s–b–t_1). If t_2 is considered as a sink, the value of the maximum flow is 50 (15 via s–a–t_2, and 5 via s–a–b–t_2, and 30 via s-b-t_2). If the maximum flow value is considered, t_2 is has to be chosen as the MaxStatic sink.

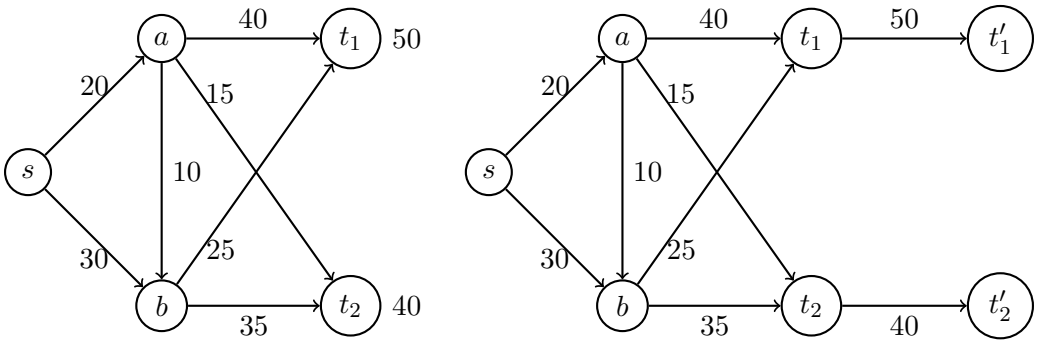
For each $t \in T$, let $c_t > 0$ be assigned its capacity. If the sinks also have the limited capacity, the sink location decision may change. The fact is illustrated in the following example.

Example 3.2. In the network given in Figure 1, let $c_{t_1} = 50, c_{t_2} = 40$ as shown in Figure 2a. If t_1 is considered as the sink, the value of the maximum flow that can reach it is 45, and if t_2 is considered as the sink, the corresponding value is limited to 40 because of the capacity of t_2. So, the MaxStatic sink is t_1.

To find the maximum static flow in the network with a capacitated sink, we can extend the network by adding a node t' and an arc tt' corresponding to the sink t such that $u_{tt'} = c_t$ and find the maximum static s-t' flow. See Figure 2b.

Observation 3.3. Given a network $N = (V, A)$ with a source s and a capacitated sink t with capacity c_t, the maximum s-t flow in network N is equivalent to the maximum s–t' flow in the network $N' = (V', A')$ such that $V' = V \cup \{t'\}$ and $A' = A \cup \{(t, t')\}$ with the capacity $u_{tt'} = c_t$.

To find the MaxStatic sink with possible capacitated sinks, we can use Algorithm 1 in the extended network formed as explained in Observation 3.3. The procedure is given in Algorithm 2.



(a) Network with possible sinks t_1 and t_2 with capacities 50 and 40

(b) Extended network corresponding to (a)

FIGURE 2. Network with capacitated sinks

Algorithm 2 MaxStatic sink with capacitated sinks

Input: Network $N = (V, A)$ with a set of possible sinks $T = \{t_1, \cdots t_k\} \subset V$ with capcity c_t for each $t \in T$

Output: MaxStatic sink

1: Form the extended network $N' = (V', A')$ with $V' = V \cup \{t'_1, \cdots, t'_k\}$, $A' = A \cup \{(t_1, t'_1) \cdots, (t_k, t'_k)\}$ such that $u_{t_i t'_i} = c_{t_i}, 1 \leq i \leq k$ and take $T' = \{t'_1, \cdots, t'_k\}$ as the set of possible sinks.
2: Use Algorithm 1 in N' to find the optimal sink $= t'_{i^*}$.
3: MaxStatic sink $= t_{i^*}$.
4: **return** MaxStatic sink

As the extended network adds $|T| < n$ nodes and $|T|$ arcs to the original network and the MaxStatic sink problem with uncapacitated sinks can be solved in a strongly polynomial time [12], we have the following theorem.

Theorem 3.4. *MaxStatic sink problem with capacitated sinks can be solved in a strongly polynomial time.*

Algorithm 1 has to iterate over all the nodes in the set of possible sinks T. But if we know the value of the minimum cut corresponding to each sink, the MaxStatic sink is the one with the highest value of the minimum cut. However, the complexity of finding the minimum cut is not less than that of finding the maximum flow, finding minimum cut does not improve the complexity of the algorithm. But the observation that the value of a maximum flow cannot exceed the value of any cut can help improve the average case complexity of the algorithm. For example, for each $t \in T, v_t \leq \sum_{i \in V_b(t)} u_{it}$ and $v_t \leq \sum_{j \in V_a(s)} u_{sj}$.

Let $T = \{t_1, t_2, \cdots, t_k\}$. If U_t is the value of an s–t cut, then the value of the maximum static flow $v_t \leq U_t$ for each $t \in T$. Let $t_1, t_2, \cdots, t_k$ be ordered in such a way that $U_{t_1} \geq U_{t_2} \geq \cdots \geq U_{t_k}$. If $v_{t_i} \geq U_{t_{i+1}}$ for some $i, 1 \leq i < k$, then $v_{t_i} \geq U_{t_j} \geq v_{t_j}$ for all $j \geq i$ showing that MaxStatic sink is in the set $\{t_1, \cdots, t_i\}$. So, we have the following result.

Theorem 3.5. *Let $T = \{t_1, t_2, \cdots, t_k\}$ be the set of possible sinks ordered in the decreasing values of cuts $U_{t_1} \geq U_{t_2} \geq \cdots \geq U_{t_k}$. If $v_{t_i} \geq U_{t_{i+1}}$ for some $i, 1 \leq i < k$, then the MaxStatic sink is* $\arg\max\{v_{t_1}, \cdots, v_{t_i}\}$.

Based on Theorem 3.5, we construct Algorithm 3 which does not necessarily have to iterate over all the nodes in T. It sorts the possible sinks in the order of decreasing values of, easy-to-obtain, cuts and stops if maximum flow in any sink is not less than the value of a cut of the next possible sink. As the sorting can be done in $O(|T| \log |T|)$ time, the average case complexity of the algorithm will be better than that of Algorithm 1.

Algorithm 3 MaxStatic sink (improved)

Input: Network $N = (V, A)$ with a set of possible sinks T

Output: MaxStatic sink

1: For each $t \in T$, $U_t = \min\{\sum_{i \in V_b(t)} u_{it}, \sum_{j \in V_a(s)} u_{sj}\}$.
2: Get the sorted set $T = \{t_1, t_2, \cdots, t_k\}$ so that $U_{t_1} \geq U_{t_2} \geq \cdots \geq U_{t_k}$
3: $U_{t_{k+1}} = 0$
4: $\max\_f = -1$
5: **for** $1 \leq i \leq k$ **do**
6: Calculate the maximum static flow v_{t_i}
7: **if** $v_{t_i} > \max\_f$ **then**
8: MaxStatic sink $= t_i$
9: $\max\_f = v_{t_i}$
10: **end if**
11: **if** $v_{t_i} \geq U_{t_{i+1}}$ **then**
12: Go to Line 15
13: **end if**
14: **end for**
15: **return** Maxstatic sink

Using Algorithm 3 in the extended network as in Algorithm 2, taking

$$U_{t'} = \min\{\sum_{i \in V_b(t)} u_{it}, \sum_{j \in V_a(s)} u_{sj}, c_t\},$$

we can construct an algorithm that works in case of capacitated sinks. The major steps of such an algorithm are illustrated and compared with Algorithm 2 in the following example.

Example 3.6. Major steps in the application Algorithm 2 to identify the optimal sink in the network given in Figure 2 are as follows.

$max\_f = -1$

First Iteration:

$v_{t'_1} = 45 > -1 = \max\_f$

MaxStatic sink $= t'_1$

$\max\_f = 45$

Second Iteration:

$v_{t'_2} = 40 < 45 = \max\_f$

At the termination of the algorithm, MaxStatic sink $= t_1$

Using Algorithm 3 we get $U_{t'_1} = \max\{50, 65, 50\} = 50, U_{t'_2} = \max\{50, 50, 40\} = 40$. So, the sorted sinks are $\{t'_1, t'_2\}$.

$max\_f = -1$

First Iteration:

$v_{t'_1} = 45 > -1 = \max\_f$

MaxStatic sink $= t'_1$

$\max\_f = 45$

$v_{t'_1} = 45 \geq 40 = U_{t'_2}$

So, the algorithm terminates without going for the second iteration and returns MaxStatic sink $= t'_1$, i.e. t_1.

4. Conclusion

In a network with arc capacities and a set of possible sinks with sink capacities, we considered a problem of identification of a sink (MaxStatic sink) so that the value of the maximum static flow is maximum. We presented a network transformation strategy so that the idea of finding MaxStatic sink in the uncapacitated case can be used to the capacitated one, showing that the problem can be solved in strongly polynomial time. We modify the simple iterative procedure of identification of MaxStatic sink by sorting the possible sinks in the decreasing order of cut-values so that we do not have to iterate over all the possible sinks in average case. One of the application area of the problem can be emergency evacuation where people in the disastrous areas have to be transferred to the safe places (sinks) but the sinks have limited capacities.

References

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network flows: Theory, Algorithms and Applications*, Prentice Hall, 1993.

[2] H. W. Hamacher, S. A. Tjandra, *Mathematical Modeling of Evacuation Problems: A State of Art*, Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer(ITWM), 2001.

[3] T. N. Dhamala, U. Pyakurel, S. Dempe, A critical Survey on the Network Optimization Algorithms for Evacuation Planning Problems, *International Journal of Operations Research*, Vol. 15, pp 101–133, 2018.

[4] H. W. Hamacher, S. Heller, B. Rupp, Flow Location (FlowLoc) Problems: Dynamic Network Flow and Location Models for Evacuation Planning, *Annals of Operations Research*, Vol. 207, pp 161–180, 2013.

[5] R. C. Dhungana, T. N. Dhamala, Maximum FlowLoc problems with network reconfiguration, *International Journal of Operations Research*, Vol. 16, pp 13–26, 2019.

[6] H. N. Nath, U. Pyakurel, T. N. Dhamala, and S. Dempe, Dynamic Network Flow Location Models and Algorithms for Quickest Evacuation Planning, *Journal of Industrial and Management Optimization*, Vol. 17, pp 2925–2941, 2021.

[7] U. Pyakurel, H. N. Nath, T. N. Dhamala, Partial contraflow with path reversals for evacuation planning, *Annals of Operations Research*, Vol. 283, pp 591–612, 2019.

[8] U. Pyakurel, H. N. Nath, S. Dempe, T. N. Dhamala, Efficient Dynamic Flow Algorithms for Evacuation Planning Problems with Partial Lane Reversal, *Mathematics*, Vol. 7, doi: https://doi.org/10.3390/math7100993, 2019.

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

[10] H. N. Nath, T. N. Dhamala, S. Dempe, A Bicriteria Model for Saving a Path Minimizing the Time Horizon of a Dynamic Contraflow, *Computer Sciences & Mathematics Forum*, Vol. 2, doi: 10.3390/IOCA2021-10897, 2022.

[11] P. Heßler, H. W. Hamacher, Sink Location to Find Optimal Shelters in Evacuation Planning. *EURO Journal on Computational Optimization*, Vol. 4, pp 325–347, 2016.

[12] H. N. Nath, T. N. Dhamala, Network Flow Approach for Locating Optimal Sink in Evacuation Planning, *International Journal of Operations Research*, Vol. 15, pp 175–185, 2018.

[13] L. R. Ford, D. R. Fulkerson, *Flows in Networks*,Princeton University Press, New Jersey, 1962.

[14] R. K. Ahuja, J. B. Orlin, Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems, *Naval Research Logistics (NRL)*, Vol. 38, pp 413–430, 1991.

[15] A. V. Karzanov, Determining the maximal flow in a network by the method of preflows, in *Soviet Math. Doklady*, Vol. 15, pp 434–437, 1974.

[16] A. V. Goldberg, *A new max-flow algorithm*, Laboratory for Computer Science, Massachusetts Institute of Technology, 1985.

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

[18] R. K. Ahuja, J. B. Orlin, A fast and simple algorithm for the maximum flow problem, *Operations Research*, Vol. 37, pp 748–759, 1989.

[19] J. B. Orlin, Max flows in $O(nm)$ time, or better, in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp 765–774, 2013.