Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 13, Issue 1 (June 2018), pp. 508 - 515

Flow Maximization Problem as Linear Programming Problem with Capacity Constraints

[1]Sushil Chandra Dimri and [2*]Mangey Ram

[1]Department of Computer Applications
[2]Department of Mathematics, Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
[1]dimri.sushil2@gmail.com; [2*]drmrswami@yahoo.com

*Corresponding author

Received: October 2, 2017; Accepted: April 19, 2018

Abstract

Flow maximization is a fundamental problem in mathematics; there are several algorithms available to solve this problem, but these algorithms have some limitations. This paper presents the flow maximization problem as a Linear Programming Problem (L.P.P.). The solution given by L.P.P. formulation of the problem and provided by Ford Fulkerson algorithm is same. This paper also compares the single path flow and k-splitting of the flow and suggests that *k*-splitting of flow is better than single path flow.

Keywords: Residual Capacity; Splittable Flow; Augmenting Path; Optimal Flow

2010 MSC No.: 90C08, 90C05

1. Introduction

Flow Network and Maximization Problem: A flow network is a graph G (V, E) a connected weighted digraph, the weight of each edge is a positive integer, which gives the capacity of the edge. There are two special vertexes S and D known as source and sink, the in the degree of the source is zero and the out degree of the sink is zero.

The maximization flow problem is to determine the maximum amount of flow flowing per unit of time from source S to sink D in a given flow network.

The best example of flow network is sending water from one reservoir to other using different paths, the sending reservoir is source S and receiving reservoir is destination *D*. Each path is made of pipe lines of different diameters (capacity), here assumption is that the source generates infinite amount of water. Now the problem is how to send the maximum amount of water from source *S* to destination *D* per unit of time.

Here, there is no intermediate storage, so flow conservation principle will follow; that is, at any intermediate node flow in is equal to flow out. In addition, flow in a pipe is either less or equal to its capacity (capacity constraints) (Cormen et al., 2001; Ford and Fulkerson, 1958; Evin, 1979).

There is famous theorem Max Flow –Min Cut theorem, which states that the amount of maximum flow from source *S* to destination *D* is equal to the capacity of minimum cut.

There are many algorithms of different complexities are available to solve the flow maximization problem. These are Ford – Fulkerson algorithm, Edmonds, Dinic's blocking flow algorithm, General push-relabel maximum flow algorithm etc.

The most popular algorithm is Ford – Fulkerson algorithm which first constructs the residual network for the given flow network. The residual network for the flow network is constructed by assuming the flow 0 in each edge and then the residual network is constructed and then augmented paths are selected, and a flow of capacity of the augmented path has been assigned to the path, and again algorithm construct the residual network. This process continues until no augmented path remains in the residual network. (Ford and Fulkerson, 1958; Gal, 1959; Kleinberg et al., 1996).

2. Ford-Fulkerson Algorithm

A flow in the network is an integer-valued function f defined on the edges of *G* satisfying

$$0 \le f(i, j) \le c(i, j), \text{ for every edge } (i, j) \text{ in } E.$$

Conservation Condition
For every vertex *j* in *V*, where *j* is not the source *S* or the destination *D*, the sum of the flow into *j* equals the sum of the flow out of *j*.

A flow that satisfies the conservation condition is called a feasible flow.

Let *f* be a feasible flow in a network *G*. The flow of the network, denoted by *f*(*G*) is the sum of flows coming out of the source *S*.

The Ford-Fulkerson algorithm is a popular method to find the solution of maximization problem. It simply follows these steps:

If an augmenting path exists in the residual network,

- Assign the flow of amount equal to the capacity of augmenting path to the augmenting path.
- Again, construct the residual network and repeat the same process until no augmenting path remains.

- The algorithm finds the augmenting path by using graph traversal methods like DFS or BFS on the flow network.

It is quite difficult to compute the actual time complexity of Ford –Fulkerson algorithm since it is not possible to determine exactly that how many augmenting paths exist in the arbitrary flow network. In the worst case of the algorithm, the search for an augmenting path takes time $O(|E|)$ time, where $|E|$ is the number of edges flow network $G(V, E)$. So worst-case, the complexity of the algorithm is $O(|f||E|)$., where $|f|$ is the amount of maximal flow in the network. (Cormen et al., 2001; Ford and Fulkerson, 1958; Ahuja et.al. 1993).

2.1. For a Single Shortest Path Flow

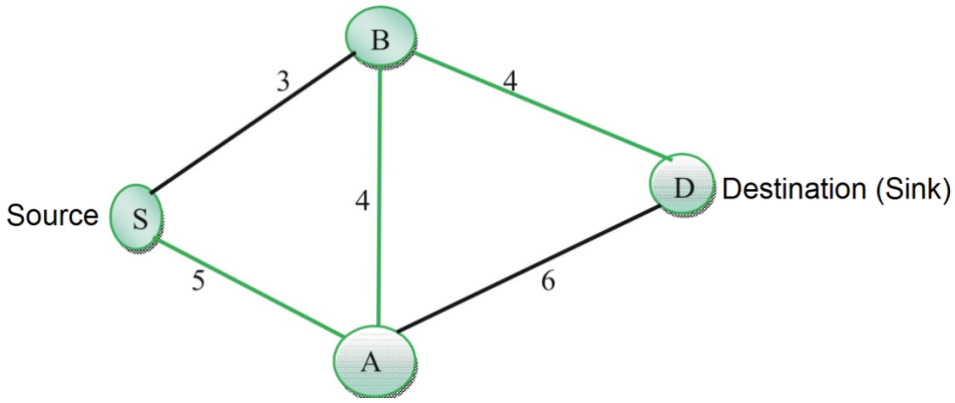

Figure 1. A capacitated flow network

The single path flow or unsplittable flow from source S to destination D in a flow network uses a single path from source S to destination D. One of the single paths from source S to destination D is shown in Figure 1 with green color. The capacity of the path p in flow network is equal to $\min\{ C_e, \ \ where \ \ e \in p \}$ where C_e is the capacity of edge e.

Of course, per unit of time maximum flow in single path flow is equal to the capacity of the path. For Figure 1, the capacity of path S-A-B-D = $\min\{5, 4, 4\} = 4$ (Sharma, 2004; Kleinberg, 1996).

2.2. *k*-Splittable Flow

A k- splittable flow is a generalization of unsplittable flow problem in which to send the data from source S to destination D by using at most k paths, these paths may or may not be distinct. Splitting the flow along many paths leads towards flow maximization and better usages of networks. But, it is difficult to deal with so many paths, especially when we need flow in a particular order (Baier et al., 2002; Martens and Skutella, 2006; Skutella, 2000).

3. Formulation of Flow Maximization Problem as an L.P.P.

Let S is the source and D is the target in the directed or undirected graph $G = (V, E)$ and the path under consideration for flow between S and D are k, these paths are not necessarily all disjointed.

P_1, P_2, P_3…, P_k are the paths between S and D and flow along these paths are F_1, F_2, F_3, …, F_k, respectively.

It is possible to transform the flow maximization problem in to a linear programming problem with the objective of maximization of total flow between S and D with the restriction of the edges capacities that is the flow value in an edge cannot exceed the capacity of the edge and the total flow cost cannot be higher than the given budget. (Dinitz et al, 1999; Dimri and Pant, 2008; Horowitz et al., 1997; Kalavathy, 2013)

The problem as L.P.P. can be formulated as

$$Max\, Z = F_1 + F_2 + F_3 + ... + F_k, \tag{1}$$

subject to

$$\sum_{i \in \{1, \ldots, k\},} F_i \leq u_{e,} \quad \forall e \in E(G), where\ e \in P_i,$$

the capacity constraint, where u_e is the capacity of edge e. (2)

The constraints indicate that the sum of flow on an edge e must be bounded by u_e (capacity of edge e), Sharma (2004) and Dimri and Pant (2008).

3.1. Numerical Problem Example

Maximization of splittable flow under above mentioned constraints where the flow is splitted along 3 paths (Figure 2).

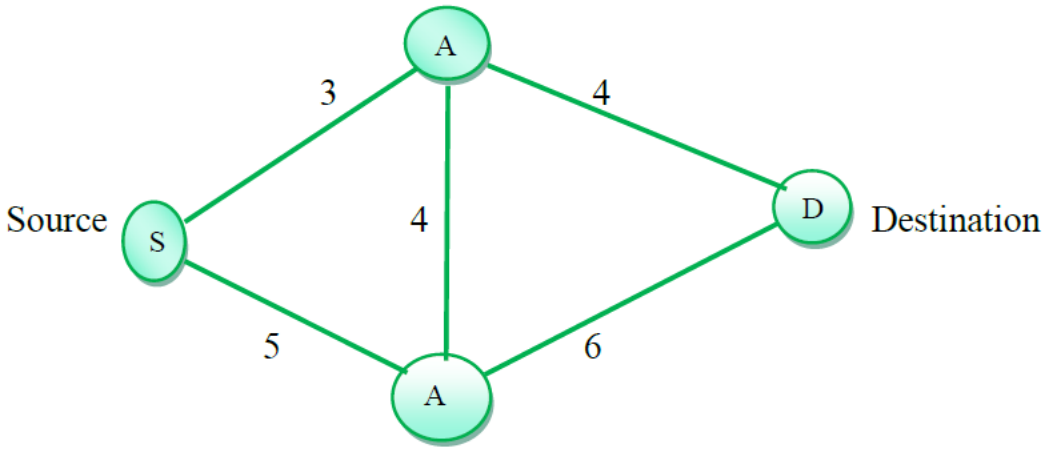

Figure 2. A capacitated flow network

Paths considered between S and D are

$$P_1 = S - A - D,$$
$$P_2 = S - B - A - D,$$
$$P_3 = S - B - D,$$
$$| \mathrm{E}(G) | = 5,$$

F_1 flow along P_1,

F_2 flow along P_2,

F_3 flow along P_3,

and max flow with $k = 3$.

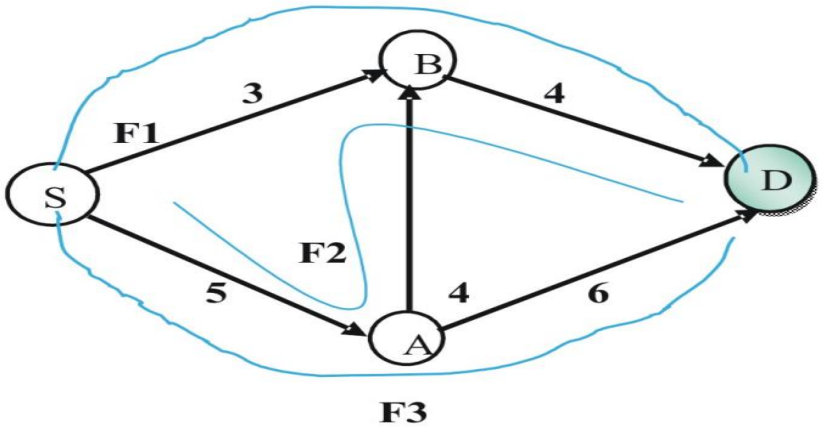

Figure 3. A capacitated flow network with possible paths

The flow constraints are

(1) $F_1 \leq 3,$

(2) $F_1 + F_2 \leq 4,$

(3) $F_2 + F_3 \leq 5,$

(4) $F_2 \leq 4,$

(5) $F_3 \leq 5.$

The L.P.P. for the given flow networks is

$$Max\ Z = F_1 + F_2 + F_3,$$

subject to flow constraints 1, 2, 3, 4 and 5.

Using slack variables F_4, F_5, F_6, F_7, F_8, the L.P.P. is

$$max\ Z = F_1 + F_2 + F_3 + 0F_4 + 0F_5 + 0F_6 + 0F_7 + 0F_8,$$

$$\begin{aligned}
&F_1 + 0F_2 + 0F_3 + 1F4 + 0F_5 + 0F_6 + 0F_7 + 0F_8 = 3,\\
&F_1 + F_2 + 0F_3 + 0F_4 + 1F_5 + 0F_6 + 0F_7 + 0F_8 = 4,\\
&0F_1 + F_2 + F_3 + 0F_4 + 0F_5 + 1F_6 + 0F7 + 0F_8 = 5,\\
&0F_1 + F_2 + 0F_3 + 0F_4 + 0F_5 + 0F_6 + 1\,F_7 + 0F_8 = 4,\\
&0F_1 + 0F_2 + F_3 + 0F_4 + 0F_5 + 0F_6 + 0F_7 + F_8 = 5,
\end{aligned}$$

where $F_1,\ F_2,\ F_3,\ F_4,\ F_5,\ F_6,\ F_7,\ F_8 \geq 0.$

The initial simplex table for the problem is

		C_j	1	1	1	0	0	0	0	0
C_B	X_B	B	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
0	F_4	3	1	0	0	1	0	0	0	0
0	F_5	4	1	1	0	0	1	0	0	0
0	F_6	5	0	1	1	0	0	1	0	0
0	F_7	4	0	1	0	0	0	0	1	0
0	F_8	5	0	0	1	0	0	0	0	1
	$\triangle_j$		-1	-1	-1	0	0	0	0	0
C_B	X_B	B	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	F_1	3	1	0	0	1	0	0	0	0
0	F_5	1	0	1	0	-1	1	0	0	0
0	F_6	5	0	1	1	0	0	1	0	0
0	F_7	4	0	1	0	0	0	0	1	0
0	F_8	5	0	0	1	0	0	0	0	1
	$\triangle_j$		0	-1	-1	1	0	0	0	0

	C_j		1	1	1	0	0	0	0	0
1	F_1	3	1	0	0	1	0	0	0	0
1	F_2	1	0	1	0	-1	1	0	0	0
0	F_6	4	0	0	1	1	-1	1	0	0
0	F_7	3	0	0	0	1	-1	0	1	0
0	F_8	5	0	0	1	0	0	0	0	1
	$\triangle_j$		0	0	-1	0	1	0	0	0
	C_j		1	1	1	0	0	0	0	0
1	F_1	3	1	0	0	1	0	0	0	0
1	F_2	1	0	1	0	-1	1	0	0	0
0	F_3	4	0	0	1	1	-1	1	0	0
0	F_7	3	0	0	0	1	-1	0	1	0
0	F_8	1	0	0	0	-1	1	-1	0	1
	$\triangle_j$		0	0	0	1	0	1	0	0

Since all $\triangle_j \geq 0$, the optimality condition satisfied. Hence, the optimal flow value components are given by

$$F_1=3, \quad F_2=1, \quad F_3=4, \text{ total flow} = 8(\text{max flow})$$

$$F_{(max)} = 8(\text{maximum flow}).$$

4. Ford Fulkerson Algorithm (*G, S, D*)

For every edge (*u*, *v*) of *E* (G):

Assign the flow $f[u, v] := 0$ to edge (*u*,*v*)

Also reverse flow in edge (*v*, *u*) i.e. $f[v, u] := 0$.

While there exists an augmenting path in residual network G_f

from source *S* and destination *D* do

$C_f(p) := \min \{C_f(u, v): (u, v) \text{ is in } p\}$

For each edge (*u*, *v*) in *p*

do: $f(u, v) := f(u, v) + C_f(p)$

$f(v, u) := - f(u,v)$

The while loop repeatedly identifies an augmenting path p in residual network G_f and add flow *f* along *p* with residual capacity $C_f(p)$. When no augmenting path remains, the flow *f* is maximum flow.

4.1. Applying the Ford Fulkerson Algorithm on Given Flow Network

Scanning the residual network for augmenting paths of flow network shown Figure 3, the first augmenting path in the residual network is *S-B-D* with capacity 3unit, so 3 unit of flow can be assigned to path *S-B-D*.

With this assignment edge (*S-B*) saturates.

Again, the next augmenting path in the residual network is *S-A-D* with capacity 5. Now sending a flow of 5 units along this path, with this edge *S-A* saturates and then after this no augmenting path remaining and so the max flow in the network is 3+5 = 8 units.

4.2. Comparison between Ford Fulkerson Algorithm and L.P.P. of Flow Maximization Problem

Ford Fulkerson algorithm only maximize the flow from source *S* to destination *D* subject to capacity constraints only while the L.P.P. formulation of flow maximization problem can deal with many constraints like cost, budget, capacity constraints etc.

In L.P.P. formulation though the calculation is expensive but this method can deal with many constraints like budget, cost except the edge constraints and the result given by L.P.P. is same as given by Ford Fulkerson algorithm.

5. Conclusion

This paper presents the flow maximization problem as a linear programming problem and the solution given by L.P.P. is same as it is given by Ford-Fulkerson algorithm. Important is that the flow maximization problem solutions algorithms are only limited to the maximization of flow with capacity constraints, while the L.P.P. formulation of flow maximization problem can deal with many types of constraints like time and cost budget constraints. Further splitting of the flow in to k-paths is a good approach to reduce the delay, to maximize the amount of flow and for optimum network resource utilization.

Acknowledgement:

The authors would like to thank the referees for their valuable suggestions and comments for improving this paper.

REFERENCES

Ahuja, R. K., Magnante, T. L., Orlin, J. B. (1993). Network flows. Prentice Hall, 1-846.

Baier, G., Köhler, E., & Skutella, M. (2002, September). On the k-splittable flow problem. In European Symposium on Algorithms (pp. 101-113). Springer, Berlin, Heidelberg.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). Introduction to algorithms second edition, 1-6.

Dimri, S. C. & Pant, D. (2008). Maximization of K-splittable flow under a given budget with edge cost and edge capacity constraints in flow network, Acta Ciencia Indica, Vol XXXIV M, No 3.

Dinitz, Y., Garg, N., & Goemans, M. X. (1999). On the single-source unsplittable flow problem. *Combinatorica*, Vol. 19, No. 1, pp. 17-41.

Evin. S. (1979). Graph algorithms. Computer Science Press, Rotomac MD

Ford Jr, L. R., & Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, Vol. 6, No. 3, pp. 419-433.

Gal, D. (1959). Transient flow in networks. *The Michigan Mathematical Journal*, *6*(1), 59-63.

Horowitz, E., Sahni, S., & Rajasekaran, S. (1997). Computer algorithms. Galgotia Publications. *Computer Science Press*, 1-759.

Kalavathy, S. (2013). Operations research. Vikas Publishing House Pvt. Ltd, 1-41

Kleinberg, J. M. (1996, October). Single-source unsplittable flow. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on* (pp. 68-77). IEEE.

Martens, M., & Skutella, M. (2006). Length-bounded and dynamic k-splittable flows. *Operations Research Proceedings*, 297-302.

Sharma J. K. (2004). *Operation research theory and applications*. MacMillan India limited.

Skutella, M. (2000). Approximating the single source m split-table num-cost flow problem. *Mathematical Programming*, 1-22.