A Review on Ford Fulkerson Graph Algorithm for Maximum Flow

Srijan Biswas[1], Saswata Sundar Laga[2], Biswajit Paul[3]
2nd year B.Tech, Dept. of CSE, Jadavpur University
Assistant Professor, Dept. of ECE, Camellia Institute of technology

Abstract— In the field of Algorithms in Computer Science, Graphs hold a very important position. Many applications commonly used by us require the application of graphs. For example, we may require the concept of graphs in GPS or Google/Yahoo maps for finding the optimal route between two locations or in case of finding the cheapest airfare between two destinations and many other such instances.

In this paper we have discussed about the design of **maximum flow graph algorithm**. The maximum flow problem is one of the most fundamental problems in network flow theory and has been investigated extensively. The Ford-Fulkerson algorithm is a simple algorithm to solve the maximum flow problem based on the idea of **residual network**, **augmenting path** and **cuts**. But its time complexity is high and it's a pseudo-polynomial time algorithm.

Index Terms— Algorithm, augmenting path, flow network, Ford-Fulkerson, graph, maximum flow, residual network.

1 INTRODUCTION

Just as we can model a road map as a directed graph in order to find the shortest path from one point to another, we can also interpret a directed graph as a "flow network" and use it to answer questions about material flows. Imagine a material coursing through a system from a source, where the material is produced, to a sink, where it is consumed, the source produces the material at some steady rate, and the sink consumes the material at the same rate. The "flow" of the material at any point in the system is intuitively the rate at which the material moves. Flow networks can model many problems, including liquids flowing through pipes, parts through assembly lines, current through electrical networks, and information through communication networks. We can think of each directed edge in a flow network as a conduit for the material. Each conduit has a stated capacity, given as a maximum rate at which the material can flow through the conduit. Vertices are conduit junctions, and other than the source and sink, material flows through the vertices without collecting in them. In other words, the rate at which material enters a vertex must equal the rate at which it leaves the vertex. We call this property **"flow conservation"**, and it is equivalent to Kirchhoff's current law when the material is electrical current. In the maximum-flow problem, we wish to compute the greatest rate at which we can ship material from the source to the sink without violating any capacity constraints.

Graph cut is a well studied concept in Graph Theory. One of the major applications of Graph cuts is in the field of Computer Vision. Many fundamental problems in Computer Vision can be reformulated as a Graph Cut problem and particularly in case of Max-flow min-cut problem.

- *Srijan Biswas is currently pursuing bachelor degree program in Computer Science and Engineering, Jadavpur University, India, PH-9748340965-mail: a.srijanbiswas.b@gmail.com*
- *Saswata Sundar Laga is currently working as Assistant Professor, Electronics and Communication Dept., Camellia Institute of Technology, MAKAUT, India, PH-9477405306, E-Mail: saswata_ece@rediffmail.com*

In this paper, we will study the Ford-Fulkerson algorithm which is based on Max-flow min-cut theorem. Finally, a brief analysis of the time complexity of the algorithm will be presented. According to graph theory, a graph cut is the grouping of nodes in a connected component into two disjoint subsets and the weight of the cut is defined to be equal to the sum of the weights of edges that are present between the disjoint subsets. If we consider the graph as a flow network, then an s − t cut is defined as a graph cut which requires the source and sink nodes to be in different subsets. The weight of an s − t cut is called as the capacity of the cut. For a given graph containing a source and a sink node, there are many possible s − t cuts. Thus a minimum cut is defined as that s − t cut whose capacity is less than or equal to every other s − t cut for the given graph.

2 FORD FULKERSON METHOD

The Ford–Fulkerson method or Ford–Fulkerson algorithm (FFA) is a greedy algorithm that computes the maximum flow in a flow network. It is called a "method" instead of an "algorithm" as the approach to finding augmenting paths in a residual graph is not fully specified or it is specified in several implementations with different running times. It was published in 1956 by L.R. Ford Jr. and Dr. Fulkerson. The name "Ford–Fulkerson" is often also used for the Edmonds-Karp algorithm, which is a specialization of Ford–Fulkerson.

The idea behind the algorithm is as follows: as long as there is a path from the source (start node) to the sink (end node), with available capacity on all edges in the path, we send flow along one of the paths. Then we find another path, and so on. A path with available capacity is called an augmenting path.

The Ford-Fulkerson algorithm particularly has a lot of applications in Image Processing and Computer Vision. Some of them are image segmentation, optical flow estimation, stereo correspondence, etc. where the given problem is transformed into a maximum flow minimum cut problem and then solved using the Ford-Fulkerson algorithm.

3 ALGORITHM

Let **G(V,E)** be a graph, and for every edge from u to v let **c(u,v)** be the capacity and **f(u,v)** be the flow. We want to find the maximum flow from the source s to sink t. After every step in the algorithm the following is maintained:

- The flow along an edge cannot exceed its capacity.
- The net flow from u to v must be the opposite of the net flow from v to u.
- That is, unless u is s or t, the net flow to a node is zero except for the source which produces the flow and the sink, which consumes flow.
- The flow leaving from s must be equal to the one arriving at t.
- Capacity Constraints: $\forall$ **(u,v) $\in$ f(u,v) <= c(u,v)**
- Skew Symmetry: $\forall$ **(u,v) $\in$ f(u,v) = -f(u,v)**
- Flow Conservation: $\forall$ **u $\in$ V : u≠s and u≠t**
  Σ f (u, w)=0, (where w $\in$ v)
- Value(f): **Σf(s,u) = Σf(v,t) (where s,u,v,t $\in$ E)**

This means that the flow through the network is a legal flow after each round in the algorithm.

We can define the residual network **Gf (V, Ef)** to be the network with capacity **cf (u, v) = c(u,v) - f(u,v)**. Notice that it can happen that a flow from v to u is allowed in the residual network, though not allowed in the original network:

- If **f(u,v)>0 and c(u,v)=0,**
  then **cf (v,u) = c(v,u) - f(v,u) = f(u,v) > 0.**

The basic algorithm is as follows:

- For each edge **(u,v) $\in$ G.E, f(u,v)=0**

While there exists a path p from s to t in the residual network

- **Gf, cf(p)=min{cf(u,v) : (u,v) is in p}**

for each edge in **(u,v) $\in$ p**
f(u,v) = f(u,v) + cf(p) (Send flow along the path)
f(u,v) = f(u,v) - cf(p) (the flow might be returned later)

The path in step 2 can be found with a breadth-first or depth-first search in **Gf(V,Ef)**. When the former is used, the algorithm is called Edmonds-Karp algorithm.

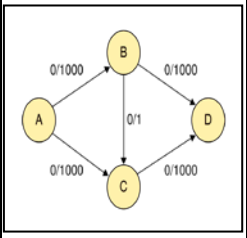
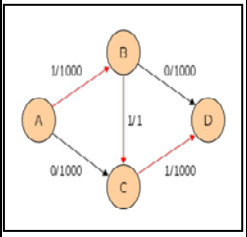
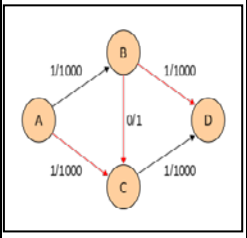
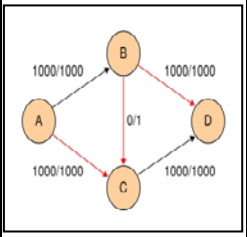
When no more paths in step 2 can be found, s will not be able to reach t in residual network. If S is the set of nodes reachable by s in the residual network, then the total capacity in the original network of edges from S to the remainder of V is on the one hand equal to the total flow we found from s to t, and on the other hand serves as an upper bound for all such flows. This proves that the flow found is maximal.

4 INTEGRAL EXAMPLE

The following example shows the first steps of Ford Fulkerson in a flow network with 4 nodes, source A and sink D.

This example shows the worst-case behaviour of the algorithm. In each step, only a flow of 1 is sent across the network. If breadth-first search were used instead, only two steps would be needed.

Table 1: Relation between path, capacity and Resulting Flow Network

PATH	CAPACITY	RESULTING FLOW NETWORK
Initial flow network		
A,B,C,D	=min(c_f(A,B), c_f(B,C), c_f(C,D)) =min(c(A,B)-f(A,B),c(B,C)–f(B,C),c(C,D) – f(C,D)) =min(1000 – 0, 1 – 0, 1000–0) = 1	
A,C,B,D	=min(cf(A,C), cf(C,B), cf(B,D)) =min(c(A,C)–f(A,C),c(C,B)–f(C,B),c(B,D) – f(B,D)) =min(1000 – 0, 0 – (-1),1000-0)=1	
After 1998 more steps…		
Resulting flow network		

Here we can find that the flow is**"pushed back"** from C to B when finding the path A, C, B, D.

5 NONTERMINATING EXAMPLE

Considering the flow network depicted in the figure below, with source **s**, sink **t**, capacities of edges e_1, e_2 and e_3 respectively **1, r = (√5 – 1)/2** and **1** and the capacity of all other edges some integer **M ≥ 2**. The constant r was chosen so, that $r^2=1-r$.

We use augmenting paths according to the following table, where
p_1={s, v_4, v_3, v_2, v_1, t}, p_2={s, v_2, v_3, v_4, t} and p_3={s, v_1, v_2, v_3, t}

Table 2: Relation between Augmenting path & Residual Capacities

STEP	AUGMENTING PATH	SENT FLOW	RESIDUAL CAPACITIES		
			e_1	e_2	e_3
0			$r^0=1$	r	1
1	{s, v_2, v_3, t}	1	r^0	r^1	0
2	p_1	r^1	r^2	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	r^3	r^2
5	p_3	r^2	r^2	r^3	0

Note that after step 1 as well as after step 5, the residual capacities of edges e_1, e_2 and e_3 are in the form r^n, r^{n+1} and **0** respectively, for some **n ∈ N**. This means that we can use augmenting paths p_1, p_2, p_1 and p_3 infinitely many times and residual capacities of these edges will always be in the same form.

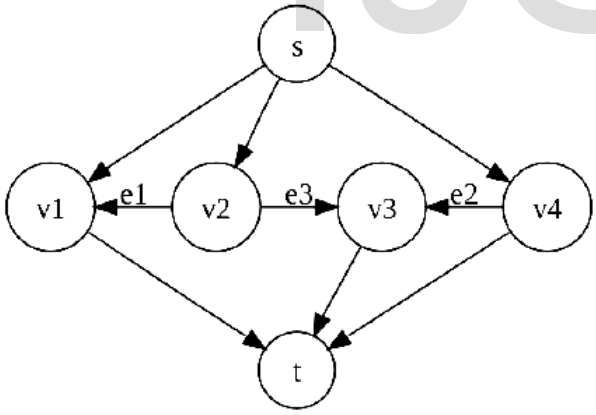


Fig 1. Flow Network

Total flow in the network after step 5 is $1 + 2(r^1 + r^2)$. If we continue to use augmenting paths as above, the total flow converges to $1+2\sum_{i=1}^{\infty} r^i = 3 + 2r$, while the maximum flow is **2M+1**. In this case, the algorithm never terminates and the flow doesn't even converge to the maximum flow.

6 ANALYSIS

The running time of FORD-FULKERSON depends on how we find the augmenting path p in line 3 of the basic algorithm. If we choose it poorly, the algorithm might not even terminate; the value of the flow will increase with successive augmentations, but it need not even converge to the maximum flow value.
In practice, the maximum-flow problem often arises with integral capacities. If the capacities are rational numbers, we can apply an appropriate scaling transformation to make them all integral. If **f*** denotes a maximum flow in the transformed network, then a straightforward implementation of FORD-FULKERSON executes the while loop of lines 3–8 at most **|f*|** times, since the flow value increases by at least one unit in each iteration.
When the capacities are integers, the runtime of Ford Fulkerson is bounded by **O(Ef)**, where E is the number of edges in the graph and f is the maximum flow in the graph. However if the graph has irrational values, the flow might not converge towards the maximum value and the algorithm runs forever.

7 CONCLUSION

In this paper we have sincerely discussed the Ford-Fulkerson method for maximum flow, analysed its algorithm and looked into an example. We can implement this amazing method in various real world applications. The method has made it easy to solve several critical problems smoothly. The Ford-Fulkerson method is indeed a game changer in the world of graph algorithms.

8 ACKNOWLEDGMENT

The authors would like to take this opportunity to express their profound gratitude and deep regards to The Department of Electronics and Communication Engineering and Electrical Engineering of Camellia Institute of Technology, Madhyagram, Kolkata.

9 REFERENCES

[1] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 26.2: The Ford–Fulkerson method". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 651–664. ISBN 0-262-03293-7.

[2] George T. Heineman; Gary Pollice; Stanley Selkow (2008). "Chapter 8:Network Flow Algorithms". Algorithms in a Nutshell. Oreilly Media. pp. 226–250. ISBN 978-0-596-51624-6.

[3] Jon Kleinberg; Éva Tardos (2006). "Chapter 7:Extensions to the Maximum-Flow Problem". Algorithm Design. Pearson Education. pp. 378–384. ISBN 0-321-29535-8.

[4] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surface via graph cuts. In International Conference on Computer Vision, volume 1, pages 26-33, 2003.

[5] Yuri Boykov and Gareth Funka-Lea. Optimal object extraction via constrained graph-cuts. International Journal of Com-

puter Vision (IJCV), 2004, to appear.

[6] L. Ford and D. Fulkerson. Flows in Networks. Princeton University Press, 1962.