Flow Networks Maximum Flow Utilizing Ford-Fulkerson Method

José Tobias, UNIFAL - MG

July 2018

1 Introduction

In this report, an implementation of Ford-Fulkerson method will be presented, with an step by step presentation of the various algorithms that composes the method. Source code can be found at: https://bit.ly/2LkLsYi.

The report is divided into three more parts. The first one is an step by step of three main concepts to understand before implementing the Ford-Fulkerson method, the second one is a graphical analysis of the algorithm steps. A brief conclusion goes at the end as the third part.

2 Main concepts

In this section I present the three main concepts which Ford-Fulkerson method is based of: residual networks, augmenting paths and cuts on flow networks. They are the main reason we do not call this procedure an algorithm, since it utilizes various algorithms to build it's foundation.

Deep mathematical theoretical foundation can be found at the book, here I will discuss the most important parts that are really useful for the algorithm implementation and understanding.

2.1 Residual networks

The formal description of a **residual network** goal is given by Cormen: "*Given a flow network G and a flow f, the residual network consists of edges with capacities that represent how we can change the flow on edges of G.*". It means that, given a **flow network**, we need to have a residual network to elucidate how we can increase our flow without breaking important proprieties like "**flow-in flow-out**" one.

So that, residual capacity, c_f, limits the amount of flow we can augment on a certain vertex, being it $c_f(u, v) = c(u, v) - f(u, v)$. Together with that, we have the definition of the augmentation of flow on edges, $(f \uparrow f')(u, v)$, defined by $(f \uparrow f')(u, v) = f(u, v) + f'(u, v) - f'(v, u)$ that will always be equal to $f'(u, v)$

if your algorithm is running under the right proprieties since $f(u, v)$ and $f'(v, u)$ are always equal.

Other important lemmas and proofs are presented by the book, but they are not quite important for the algorithm implementation. All the theoretical base used and understood by me is presented above.

2.2 Augmenting paths

Given a source s and a sink t, a **augmenting path** p is simple a path from s to t in the residual network. Like said before and obviously, we should not have more flow-in than flow-out on a vertex, since then, we should augment this path without breaking this limitation on edge u,v with their capacities $c_f(u, v)$ and $c_f(v, u)$.

By doing that, after each iteration of the method, we should augment paths by $c_f(p)$, being $c_f(p) = min\{c_f(u, v) : (u, v) \in p\}$. Should be clear to see that if we augment the path by the minimum residual flow on it, we should not violate any residual capacity or capacity of vertexes, since $(c_f(u, v) - c_f(p) \leq c(u, v) + f(u, v))$ by the giving past proprieties.

Other important lemmas, corollaries and proofs are presented by Cormen in the book section, but none of them were important to the implementation or to understand the Ford-Fulkerson method.

2.3 Cuts of flow networks

But how does the augmenting path is discovered? And how do we know it's the maximum flow? Those questions can be answered by the **cuts of flow networks**. Like a cut on graph studied before, we have cuts made on the flow network to give us the information like "*where can we augment the path*".

On the start of the iteration we have a set E which contains all the vertexes on a flow network. As we progress walking through the flow network, starting at our source s we mark vertexes added to the augmenting path, forming a S set of those vertexes, while the $T = E - S$ is also formed, such that $s \in S$ and $t \in T$. The cut on a graph permits us to clearly see which edge u, v as $u \in S$ and $v \in T$ will we choose following the **max-flow min-cut** theorem. The algorithm then proceeds to get the **minimum cut**, that means, get the minimum possible flow that connects sets S and T.

The max-flow min-cut theorem is explained and proved by Cormen, but the basic understanding needed is that: if you have no way to augment your path by it's minimum cut, you have a maximum flow now on your network. No further understanding needed for this method implementation.

3 Graphical Analysis

On a flow network given by the input:

origem	destino	capacidade
0	1	16
0	2	13
1	2	12
2	1	4
2	4	14
3	2	9
3	5	20
4	3	7
4	5	4

Table 1: Example of graph input for the given flow network maximum flow problem.

We will have the following graph and path:

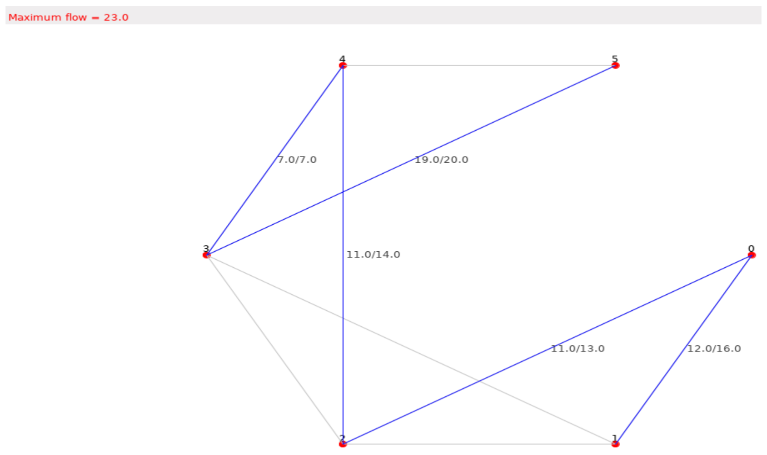

Figure 1: Example of flow network with it's augmenting path and maximum flow of 23.

4 Conclusion

The method itself is not very hard to implement, but it's hard to comprehend since it uses a lot of theorems and concepts already seen in the discipline. The non-trivial parts are hard to be seen, and hard to test, since this method is not guaranteed to find the maximum flow for any s and t chosen.

Cormen gives a lot of theoretical proofs based on his mathematical background, but provides no easy way to understand what is really going on on the algorithm, giving a pseudo-code really hard to translate into high level programming language syntax.