Flow Networks

Abderrahman Laoufi, Yanis Merzouki

April 12, 2022

This is the augmented transcript of a lecture given by Luc Devroye on the 12th of April 2022 for the Honours Data Structures and Algorithms class (COMP 252). The subject was flow networks.

Flow Network

Definition 1. A **flow network** $G = (V, E)$ is a directed graph whose edges $(u, v) \in E$ have a non negative capacity $c(u, v) \geq 0$. It is also distinguished by the presence of the vertices s (source) and t (target or sink).

In fact, for all $v \in V$, a flow network contains a path $s \rightsquigarrow v \rightsquigarrow t$, i.e., there exists a path from s to t which goes by v.

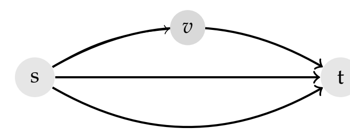


Figure 1: Every arbitrary node $v \in V$ between s and t is reachable.

Definition 2. A **flow**[1] is a function $f : V \times V \to \mathbb{R}$ that satisfies:

[1] One can compare a flow f with the current in an electrical network.

1. **Capacity rule**: $\forall u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$
2. **Skew symmetry**: $\forall u, v \in V$, $f(u, v) = -f(v, u)$
3. **Conservation (Kirchhoff's law[2])**: $\forall u \in V/\{s, t\}$, we have:

$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u).$$

[2] Arshad [2010]

Definition 3. The **value**[3] $Val(f)$ of a flow network f is defined by the following formula:

[3] Our objective is to maximize the value of the flow.

$$\begin{aligned} Val(f) &= \sum_{v \in V} f(s, v) \qquad \text{(leaving } s\text{)} \\ &= \sum_{v \in V} f(v, t) \qquad \text{(arriving at } t\text{).} \end{aligned}$$

Example 4. A flow network with a value $Val(f) = 19$.

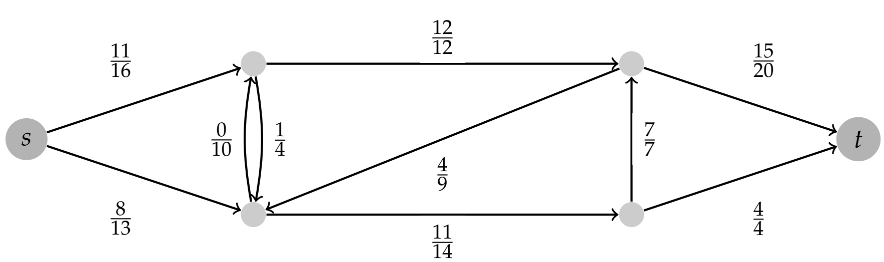


Figure 2: Example of a flow network. Only positive flows are shown. For each edge, the numerator represents the flow, and the denominator represents the capacity.

Some Properties

Definition 5. If we define $f(A, B)$ as follows:

$$f(A, B) = \sum_{x \in A} \sum_{y \in B} f(x, y)$$

then, the following properties hold:

1. $\forall u \notin \{s, t\} : f(u, V - \{u\}) = f(u, V) = 0$
   $f(V, u) = f(V - \{u\}, u) = 0$
2. $f(s, v) = f(s, V - \{s\})$
3. $f(A, A) = 0$
4. $f(A, B) = -f(B, A)$
5. $A \cap B = \emptyset \Rightarrow f(A \cup B, C) = f(A, C) + f(B, C)$

The Ford-Fulkerson Method

The Ford-Fulkerson[4] method's goal is to increase the flow's value iteratively. We set $f(u, v) = 0 \;\; \forall u, v \in V$ at the start of the method. Each loop iteration increases the flow value in G by finding an "augmenting path" in what we call a "residual network" G_f.[5]

[4] Ford and Fulkerson [1956]

[5] Residual networks are explained below

Remark 6. Note that although the value of the flow f increases, the flow of a specific edge (u, v) could increase or decrease.

Ford-Fulkerson Method

```
flow f ≡ 0
while ∃ an augmenting path p
// On which we can send more flow
      augment flow along p
return f
```

To be able to efficiently analyze the Ford-Fulkerson method, the notion of residual network needs to be introduced.

The Residual Network

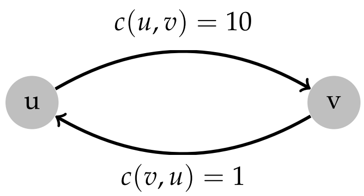
Definition 7. The **residual network** defined on a flow network G, denoted by G_f, is a group of edges whose capacities $c_f(u, v)$[6] are altered depending on a flow f of G.

[6] $c_f(u, v)$ is called **residual capacity**

Definition 8.

$$c_f(u, v) = c(u, v) - f(u, v)$$

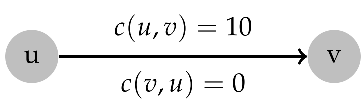
Example 9. Example of a residual capacity computation with two edges between two nodes



$f(u,v) = 5$

$$c_f(u,v) = 10 - 5 = 5$$
$$c_f(v,u) = 1 - (-5) = 6$$

Example 10. Example of a residual capacity computation with one edge between two nodes



$f(u,v) = 7$

$$c_f(u,v) = 10 - 7 = 3$$
$$c_f(v,u) = 0 - (-7) = 7$$

Example 11. An example of a residual network G_f with an augmenting path p colored in red.

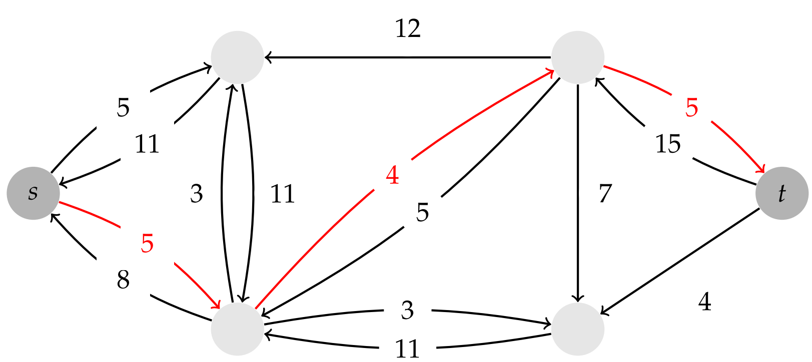
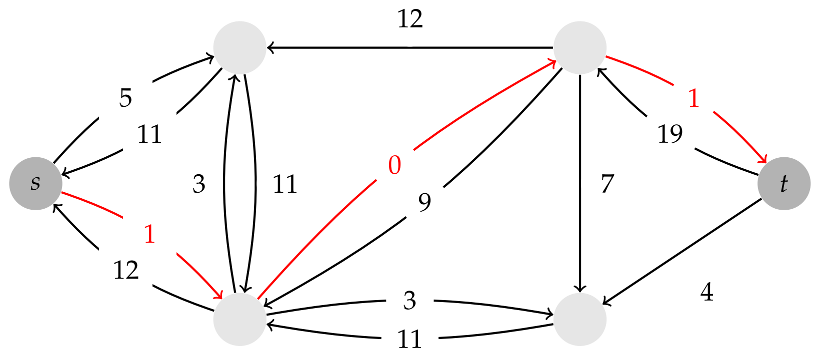


Figure 3: From Figure 2, we can extract the following residual network. The edges' labels represent the new capacity c_f of every edge.

The red path p accepts a flow of 4. If we do push a flow of 4 along this path, then we obtain the following graph:



Where

$$f(u,v) = f(u,v) - \text{flow along the edge}$$

and

$$f(v,u) = f(v,u) + \text{flow along the edge}$$

Then, the vertex t is not reachable anymore from vertex s, as there is no flow that can pass via the middle edge of the augmenting path p, which would make the new value be:

$$Val(f) = 19 + 4 = 23$$

with 19 being the old value $Val(f)$ and 4, the residual flow.

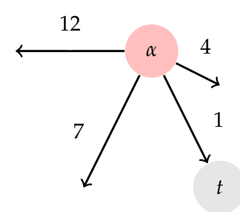


Figure 4: If the edge with a capacity of 1 is the only edge leading to vertex t, and α can't be reached, then there is no more path from t to s.

Finding Augmenting Path p

Let $G_f = (V, E_f)$, with $E_f = \{(u,v) : c_f(u,v) > 0\}$ and $|E_f| \leq 2|E|$. Then, we can perform DFS[7] until t is reached. If t is not reached, then no augmenting path can be found. Otherwise, define p as the path in the DFS tree from s to t. This operation take time $O(|E|)$.
Let $c_f(p) = \min_{(u,v)\in p} c(u,v)$ and define a function $f^* : V^2 \to \mathbb{R}$ such that:

$$f^*(u,v) = \begin{cases} c_f(p) & (u,v) \in p, \\ 0 & \text{otherwise.} \end{cases}$$

If f is a flow on G and f^* is a flow on G_f, then $f + f^*$ is a flow with a value

$$Val(f + f^*) = Val(f) + Val(f^*)$$

for G.

[7] We can also perform Edmonds-Karp's BFS, Edmonds and Karp [1972], or any other traversal in here.

Theorem 12. The following are equivalent:

$$G_f \text{ has no augmenting path}$$
$$\Leftrightarrow$$
$$f \text{ is a maximal flow on } G$$
$$\Leftrightarrow$$
$$Val(f) = c(S,T) \text{ for some cut } (S,T) \text{ of } G, \text{ where } s \in S \text{ and } t \in T.$$

Definition 13. A **cut**[8] (S, T) is a partition of the set of vertices V in G in two disjoint sets S and T, where the source vertex s lies in S and the target vertex t lies in T.

[8] Cormen et al. [2009]

Definition 14.

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

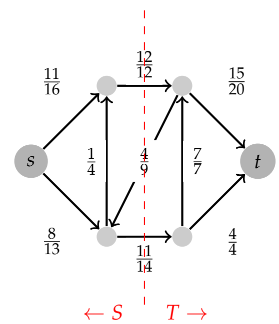


Figure 5: Example of a cut (S, T).

Complexity

If capacities are integers, then $c_f(p) \geq 1$. Therefore, the overall time complexity for finding augmenting paths is $O\left(|E| \cdot Val(f)\right)$, where $Val(f)$ is the overall value.

Remark 15. Updating G to G_f takes $O(|E|)$. Also, DFS or BFS takes a similar time $O(|E|)$. As for the number of iterations, it does not exceed the overall value $Val(f)$.

Validity of Some Claims

1. Updating G to G_f yields a flow $f + f^*$.

Proof:

Capacity rule:

$$f(u,v) + f^*(u,v) \leq f(u,v) - \underbrace{(c(u,v) - f(u,v))}_{c_f(u,v)} = c(u,v)$$

Skew symmetry:

$$f(u,v) + f^*(u,v) = -f(v,u) - f^*(v,u) = -\left(f(v,u) + f^*(v,u)\right)$$

Conservation (**Kirchhoff's law**):
If $v \notin \{s,t\}$, then:

$$\sum_{v \in V} \left(f(u,v) + f^*(u,v)\right) = 0 + 0 = 0$$

2. $Val(f+f^*) = Val(f) + Val(f^*)$

Proof:

$$\begin{aligned} Val(f+f^*) &= \sum_{v\in V} (f(s,v) + f^*(s,v)) \\ &= \sum_{v\in V} f(s,v) + \sum_{v\in V} f^*(s,v) = Val(f) + Val(f^*) \end{aligned}$$

Example 16. Let p be an augmenting path, and assume that

$$f^*(u,v) = \begin{cases} c_f(p) & (u,v) \in p, \\ -c_f(p) & (v,u) \in p, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$Val(f^*) = c_f(p)$$

and

$$Val(f+f^*) = Val(f) + c_f(p).$$

The Max Flow Min Cut Theorem

Definition 17. A **minimum cut** of a network is a cut that yields the minimum capacity in the set of all cuts (S,T).

Theorem 18. Let $f(S,T)$ be defined according to DEFINITION 5 above.
Then,

$$Val(f) = f(S,V) = f(S,V) - f(S,S) = f(S,T)$$

and

$$Val(f) = \sum_{u\in S}\sum_{v\in T} f(u,v) \leq \sum_{u\in S}\sum_{v\in T} c(u,v) \stackrel{\text{def}}{=} c(S,T).$$

Thus, we get that[9]

$$\max_{\text{flows } f} Val(f) \leq \min_{\text{cuts } (S,T)} c(S,T).$$

[9] The theorem states that the inequality in the result is actuall an equality.

Proof:

$$\begin{aligned} f \text{ is a maximal flow } &\Rightarrow G_f \text{ has no augmenting path.} \\ &\Rightarrow \text{We get the following image:} \end{aligned}$$

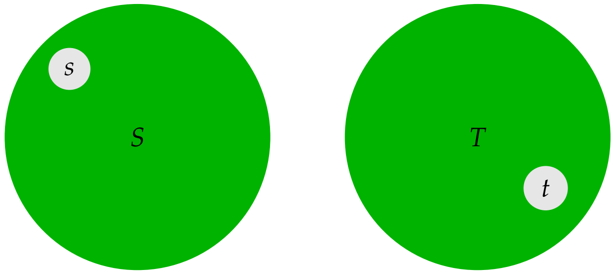


Figure 6: The set S contains all nodes v that can be reached from s in G_f. The set T contains the remaining nodes.

So $\forall u \in S, v \in T : f(u,v) = c(u,v)$, or there would have been an edge between the sets S and T. This implies that:

$$\begin{aligned} Val(f) &= f(S,T) \\ &= \sum_{u \in S} \sum_{v \in T} c(u,v) \\ &= c(S,T). \end{aligned}$$

Thus,

$$\max_{\text{flows } f} Val(f) \geq c(S,T) \geq \min_{\text{cuts } (S^*,T^*)} c(S^*,T^*).$$

Edmonds-Karp Version

Edmonds and Karp propose a version in which BFS is used to find augmenting paths. They were able to show that there are at most $|E| \cdot |V|$ augmenting path steps[10]. Thus the total complexity of the Edmonds-Karp version is $O\left(|E|^2 \cdot |V|\right)$.

[10] No proofs will be given here. Check the references for more information.

References

M. Arshad. *Network Analysis and Circuits*. Jones and Bartlett Publishers, 2010. ISBN 9780763773786. URL `https://archive.org/embed/networkanalysisc0000arsh`.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009. ISBN 9780262033848.

Jack Edmonds and Richard M. Karp. Theoretical improvements in the algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972. ISSN 0004-5411. DOI: 10.1145/321694.321699. URL `https://doi.org/10.1145/321694.321699`.

Lester R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. DOI: 10.4153/CJM-1956-045-5. URL `https://doi.org/10.4153/CJM-1956-045-5`.