

## Network flow problems

The topic „network flow problems“ considers a network, that is represented by a directed and weighted graph  $G$  with vertices  $v \in V$  and Edges  $(u,v) \in E$  and a weight  $c(u,v)$  called the capacity. In this network  $G$  a flow  $f$  of lets say goods from a specific source  $s$  to a particular sink  $t$  is being created and the problem, called the maximum flow problem, will be to find the greatest of all valid flows, whereas valid means that a flow has to serve specific properties. The motivation for this is the frequent occurrence of the max-flow problem in reality, whenever a maximum amount of units (information, goods, liquid, ...) has to be shipped from one place to another over a network of connections which can carry only a limited amount of units per time. Just think of a derrick as a source of petrol that has to be pumped over a system of oilpipes to a sink, e. g. a port. These pipes are of different size and are connected with special junctions where gates regulate the direction of the flow.

Formally this flow in  $G$  is a real valued function  $f: V \times V \rightarrow \mathbb{R}$  with 3 properties:

capacity constraint:  $\forall u,v \in V$ , we require  $f(u,v) \leq c(u,v)$  (1)

skew symmetry:  $\forall u,v \in V$ , we require  $f(u,v) = -f(v,u)$  (2)

Flow conservation:  $\forall u,v \in V \setminus \{s,t\}$ , we require  $\sum_{v \in V} f(u,v) = 0$  (3)

The expression  $f(u,v)$  means here the net or actual flow from  $u$  to  $v$ , which implies the notion that positive flows can cancel out each other if they have opposite direction.

For example:  $f(u,v) = 8$  and  $f(v,u) = 3$  result to a net flow of  $f(u,v) = 5$  and  $f(v,u) = -5$  in the reverse direction according to (2).

The flow properties (1) and (2) remain self-explanatory, to (3) I want to add that in the vertices, except  $s$  and  $t$ , flow cannot increase or disappear, that is that the (positive) net flow entering the vertex must equal the (positive) net flow leaving the vertex.

Above all that the value of a flow  $f$  in  $G$  is defined as  $|f| = \sum_{v \in V} f(s,v)$ , which sums up all the flows going out from the source.

### The Ford-Fulkerson Algorithm

Ford and Fulkerson found a method how to solve the max-flow problem, here its pseudo code:

- 1) initialise flow  $f$  to 0
- 2) while there exists an augmenting path  $p$
- 3)     do augment flow  $f$  along  $p$
- 4) return  $f$

In order to understand that method 3 different notions have to be introduced and explained: residual networks, augmenting paths and cuts of flow networks.

The residual network  $G_f$  of a given flow network  $G$  with a valid flow  $f$  consists of the same vertices as in  $G$  which are linked with residual edges  $E_f$  that can admit more strictly positive net flow.

The residual capacity  $c_f$  represents the weight of each edge  $E_f$  and is the amount of additional net flow  $f(u,v)$  before exceeding the capacity  $c(u,v)$

$$c_f(u,v) = c(u,v) - f(u,v) \quad (4)$$

An augmenting path  $p$  is a simple (free of any cycle) path from  $s$  to  $t$  in the residual network  $G_f$  and has got the following residual capacity:

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\} \quad (5)$$

We define now a flow  $f_p: V \times V \rightarrow \mathbb{R}$  along the augmenting path in  $G_f$  such as:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -c_f(p) & \text{if } (v,u) \text{ is on } p \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

This flow is valid, because it doesn't violate skew symmetry (2) by the definition of (6) and with (5) also the capacity constraint is fulfilled and finally flow conservation (3) is obeyed according to the definition of the path  $p$  going from  $s$  to  $t$ . Our idea is now to add this new flow  $f_p$  to the flow  $f$  in the original network which will result in a new flow  $f'$  which must be strictly greater than  $f$ , as  $f_p$  is strictly greater than 0 according to the definition.

$$f': V \times V \rightarrow R : f' = f + f_p \quad (7)$$

At this point I will give a formal proof that  $f'$  is a valid flow and therefore obeys the tree flow properties:

(A) capacity constraint:

We have  $f_p(u, v) \leq c_f(u, v)$  according to (6) and (5). By addition of  $f(u, v)$  at both sides of the equation and the following equality of the left side according to (4)  $c_f(u, v) + f(u, v) = c(u, v) - f(u, v) + f(u, v) = c(u, v)$  we finally get:

$$(f + f_p)(u, v) = f(u, v) + f_p(u, v) \leq c(u, v)$$

(B) skew symmetry:

$$(f + f_p)(u, v) = f(u, v) + f_p(u, v) = -f(v, u) - f_p(v, u) = -(f(v, u) + f_p(v, u)) = -(f + f_p)(v, u)$$

(C) flow conservation:

$$u \in V - \{s, t\} \Rightarrow \sum_{v \in V} (f + f_p)(u, v) = \sum_{v \in V} (f(u, v) + f_p(u, v)) = \sum_{v \in V} f(u, v) + \sum_{v \in V} f_p(u, v) = 0 + 0 = 0$$

As we have found out that the new flow  $f'$  is a valid flow that is greater than the original flow  $f$ , we know now how to augment a flow if we find an augmenting path  $p$  in the residual network  $G_f$ , but what happens if we cannot find any more an augmenting path, will the flow be maximum then?

To proof this we need to understand the notion of a cut of a flow network

A cut  $(S, T)$  of a flow network  $G=(V, E)$  is a partition of  $V$  into  $S$  and  $T = V \setminus S$  such that  $s \in S$  and  $t \in T$ . Note that  $S$  and  $T$  are sets of vertices and  $s$  and  $t$  are particular vertices. Our functions above used up to now only vertices as arguments, in order to work also with sets as arguments of these functions we shall use an implicit summation notation that computes the flow  $f(S, T)$  over a cut and the capacity of a cut  $c(S, T)$  as the sum of all possible ways of replacing the sets  $S$  and  $T$  by their members. It follows:

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) \quad \text{and} \quad c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$$

Together with this new notation and some lemmas derived from it, that I do not want to mention in my brief overview, it can be proofed that the flow  $f(S, T)$  over any cut  $(S, T)$  in the form stated above is equal to the value of the flow  $f$  in  $G$

$$f(S, T) = |f| \quad (8)$$

Moreover the value of any flow  $f$  in a flow network  $G$  is bounded from above by the capacity of any cut of  $G$

$$|f| \leq c(S, T) \quad (9)$$

Proof:  $|f| = f(S, T) = \sum \sum f(u, v) \leq \sum \sum c(u, v) = c(S, T)$

Now we can proof the question above if the non-existence of an augmenting path implies that the flow is maximum and vice versa which is an essential part of the important maximum-flow min-cut theorem.

The maximum-flow minimum-cut theorem

If  $f$  is a flow in a flow network  $G=(V, E)$  with source  $s$  and sink  $t$ , then the following statements A, B and C are equivalent:

- A.  $f$  is a maximum flow in  $G$
- B. The residual network  $G_f$  contains no augmenting paths
- C.  $|f| = c(S, T)$  for some cut  $(S, T)$  of  $G$  (10)

proof:

(A)  $\Rightarrow$  (B):

We assume for the sake of contradiction that  $f$  is a maximum flow in  $G$  but that there still exists an augmenting path  $p$  in  $G_f$ .

Then as we know from above, we can augment the flow in  $G$  according to (7). That would create a flow  $f'$  that is strictly greater than the former flow  $f$  which is in contradiction to the assumption that  $f$  is a maximum flow.

(B)  $\Rightarrow$  (C)

We suppose that  $G_f$  has no augmenting path according to 2, that is that  $G_f$  contains no path from  $s$  to  $t$ . Now we define a set  $S = \{v \in V \mid \exists \text{ a path } p \text{ from } s \text{ to } v \text{ in } G_f\}$  which contains the source  $s$  and all from  $s$  reachable vertices  $v$  in  $G_f$ .  $T = V \setminus S$  will be the complementary set of  $S$  in  $V$ . Then this partition will be a cut, as  $t$  cannot be an element of  $S$  according to 2.

The  $S$  and  $T$  connecting vertices  $(u, v)$  with  $u \in S$  and  $v \in T$  must all carry a maximum flow, we call these edges critical edges, otherwise  $v$  would be an element of  $S$ .

As all connecting edges are critical edges the flow over this cut must be maximum and therefore equal to the capacity of the cut:  $f(S, T) = c(S, T)$ .

As we know from (8) before the flow in the network equals the flow over any cut and equals here also the capacity of this cut  $|f| = f(S, T) = c(S, T)$ .

(C)  $\Rightarrow$  (A)

as stated before  $|f| = f(S, T) \leq c(S, T)$  (according to (8),(9)) and (C):  $|f| = c(S, T)$  implies (A) that  $f$  is a maximum flow

Now that we have introduced the main ideas of the Ford-Fulkerson algorithm and proved the max-flow min-cut theorem which actually proves the correctness of the algorithm, we can look more closely at this method which solves systematically the max-flow problem from above. The basic structure of the algorithm is the following:

```
1   for each edge  $(u, v) \in E[G]$ 
2       do  $f[u, v] = 0$ 
3        $f[v, u] = 0$ 
4   while there exists a path  $p$  from  $s$  to  $t$  in the residual network  $G_f$ 
5       do  $c_f(p) = \min \{c_f(u, v) \mid (u, v) \text{ is in } p\}$ 
6       for each edge  $(u, v)$  in  $p$ 
7           do  $f[u, v] = f[u, v] + c_f(p)$ 
8            $f[v, u] = -f[u, v]$ 
```

In the first three lines the flow in the network will be initialised to 0 and the while loop in line 4 repeats line 5 to 8 as long as an augmenting path can be found in the residual network. If an augmenting path  $p$  is found in line 4 then the capacity of  $p$  will be computed in line 5, whereby we take for granted that the capacities are constant time functions and if moreover an edge does not exist between two vertices the capacity will be set to 0. Finally in every while loop the flow in the original network will be increased in line 6 to 8 according to the mathematical definitions above (statement (5), (6) and (7)). After finitely increasing the flow the while loop will finally be forced to terminate because of the max-flow min-cut theorem (10) stated above. In the end there are no more augmenting paths in the residual network and we will have created a maximum flow.

What is left now is to look more closely at the way how an augmentation path can be found, what will actually determine the running time of Ford-Fulkerson.

A simple bound for the running time can be found if the augmenting path is found arbitrarily and furthermore if all capacities are integers. Implemented like that the algorithm increases the flow at least by one unit per loop of which the number of executions lies within  $O(|f_{\max}|)$ . Obviously the lines 1 to 3 and 6 to 8 will take  $O(|E|)$  time as the flow has to be altered on the edges of the network. The choice of the augmenting path, if it exists, is arbitrarily, but still we

need breadth or depth first search for going through the graph and looking for any augmenting paths which will take us another  $O(|E'|)$  time. As the number of edges in the residual network are roughly the same as the edges in the original network we approximate  $O(|E'|)$  to  $O(|E|)$  and finally we have the simple bound for the running time of  $O(|E| \cdot |f_{\max}|)$ .

If capacities and maximum flow are low the running time is okay, but not if we work with big capacities, that implies also a big maximum flow. In the latter we need a better implementation of Ford-Fulkerson. There are some implementations of which I will explain only one of them briefly: the Edmonds-Karp algorithm.

Here the augmenting path will be found by breadth-first search and moreover it has to be a shortest path from  $s$  to  $t$  in the residual network.

As we work with shortest path now, we have to define that each edge has got unit distance. We will try to find a better bound for the running time. The general assumption with this implementation is that the shortest path distance of a vertex from the source increases monotonically with each flow augmentation:  $df(s,v) \leq df'(s,v)$ . As the proof of this statement is a bit tricky and would blow up the content too much, it will be left unproved.

The above finding helps explain the theorem that the total number of flow augmentations performed by the algorithm is at most  $O(|V| \cdot |E|)$ . In order to understand that the notion of a critical edge has to be introduced. At least one edge  $(u,v)$  on an augmenting path  $p$  must have the same residual capacity as  $p$ :  $c_f(p) = c_f(u,v)$  according to (5), this edge is called critical.

We can observe that this edge  $(u,v)$  disappears from the residual network after the flow was augmented along  $p$ . For this edge  $(u,v)$  to appear again in the residual network and therefore to be able to become critical again, the flow has to be augmented along the opposite direction  $(v,u)$ . As the shortest path distance of  $u$  increases monotonically with every flow augmentation  $(u,v)$  can only become at most  $O(|V|)$  times critical, because the number of vertices restricts the shortest path distance that  $u$  can have from the source. Since only  $O(|E|)$  pairs of vertices can have an edge between them in a residual network, the total number of critical edges is restricted by  $O(|V| \cdot |E|)$ . With the fact that each augmenting path  $p$  can be found with breadth first search in  $O(|E|)$  time and that  $p$  has at least one critical edge we get the bound of the running time of the Edmonds-Karp algorithm as:  $O(|V| \cdot |E|^2)$

Moreover there are some further important variations of the max-flow problem that have to be mentioned in this context, but will not be explained in detail

- The algorithm of Dinic (different implementations reap better running times than Edmonds-Karp): e.g. simultaneous saturation of augmenting paths of the same length  $O(|V|^2 \cdot |E|)$
- A second capacity function  $lim$  for a lower limit of the flow function  $f$  with  $lim(u,v) \leq f(u,v) \leq c(u,v)$
- A cost function cost whereby each edge  $(u,v)$  has a second weight  $cost(u,v)$  and the min-cost max-flow problem
- Networks with multiple sources and sinks where all sources and sinks will be connected with edges of infinite capacity to a supersource or supersink respectively.

## Bibliography

- (1) T. H. Cormen ; C. E. Leiserson ; R. L. Rivest:  
Introduction to Algorithms  
The MIT Press, Cambridge, Mass. u. a. 1990, page 579-600,  
very good detailed and easy description
- (2) M. T. Goodrich, R. Tamassia:  
Data Structures and Algorithms in Java,  
John Wiley & Sons, Inc., 1998. § 10.3  
short overview with definitions and no proofs
- (3) T. Ottmann; P. Widmayer;  
Algorithmen und Datenstrukturen,  
Spektrum Verlag Heidelberg, 1990, page 584-590  
easy overview without mathematic details
- (4) U. Schöning:  
Algorithmen – kurz gefasst, Spektrum,  
Heidelberg u.a., 1997, page 138-147  
short and brief overview, easy to read
- (5) R. Sedgewick: Algorithmen, Addison-Wesley,  
Bonn, 1991, page 551-559  
good explanations, without mathematical proofs, overview
- (6) A. Brandstädt;  
Graphen und Algorithmen, Teubner,  
Stuttgart, 1994, chapter 6  
very detailed, difficult proofs with unusual notations
- (7) Dieter Jungnickel  
Graphs, networks and algorithms, page 155-171  
good overview, with acceptable descriptions
- (8) Ronald Gould  
Graph Theory, page 99-112  
brief overview