1. **Cuts**

Also called s-t cuts in the context of flow networks.

Def 1: A **cut** [*S*, *T*] of a flow network $N = (G=(V, E), s, t, c)$ partitions V into two sets S and T of disjoint nodes such that $s \in S$ and $t \in T$.

In other words, [*S*, *T*] is a cut if:

- $S, T \subset V$
- $s \in S, t \in T$
- $S \cup T = V, S \cap T = \emptyset$

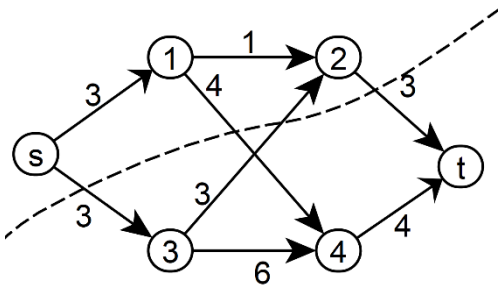


Figure 1

Consider the graph in Figure 1. A cut (represented by the dashed line in the graph) might be $S = \{s,1,2\}$ and $T = \{3,4,t\}$.

To be clear, any partition where s is in one part and t in the other is a valid cut. For example $S = \{s, 2\}$ and $T = \{1,3,4,t\}$ is also a valid cut, shown in Figure 2 by a dashed line.

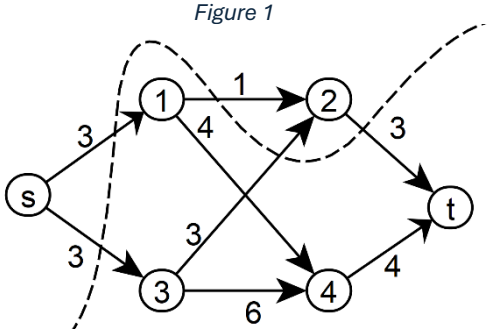


Figure 2

A cut is sometimes defined as the set of edges for which one endpoint is in S and another in T.

For the cut in Figure 1 the edges that define the cut are {(s,3), (1,4), (3,2), (2,t)}.

For the cut in Figure 2, the edges that define the cut are {(s,3), (s,1), (1,2), (3,2), (2,t)}.

Def 2: Let $N = (G, s, t, c)$ be a flow network with a cut [*S*, *T*]. An edge $(v_1, v_2) \in E$ with $v_1 \in S$ and $v_2 \in T$ is called a **forward edge** of the cut [*S*, *T*]. If $v_2 \in S$ and $v_1 \in T$ then it is called a **backwards edge.**

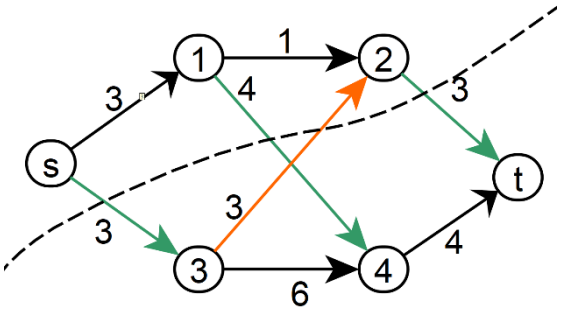


Figure 3

For the cut in Figure 1, the forward edges are (s,3), (1,4) and (2,t), while there is only one backwards edge: (3, 2). In Figure 3 are highlighted in green the forward edges of this cut, and in orange the backward edges.

Def 3: the capacity of a cut [*S*, *T*], is defined as the sum of all capacities of forward edges in the cut:

$$c(S, T) = \sum_{v_1 \in S, v_2 \in T} c(v_1, v_2) .$$

The capacity for the cut in Figure 3 is 10 – we sum only the capacities of forward edges:

$c(s,3) + c(1,4) + c(2,t) = 3 + 4 + 3 = 10.$

Def 4: the **flow across a cut** [*S*, *T*] is the difference between the sum of the flows in forward edges and the sum of the flows in backward edges:

$$f(S,T) = \sum_{v_1 \in S, v_2 \in T} f(v_1, v_2) - \sum_{v_1 \in S, v_2 \in T} f(v_2, v_1)$$

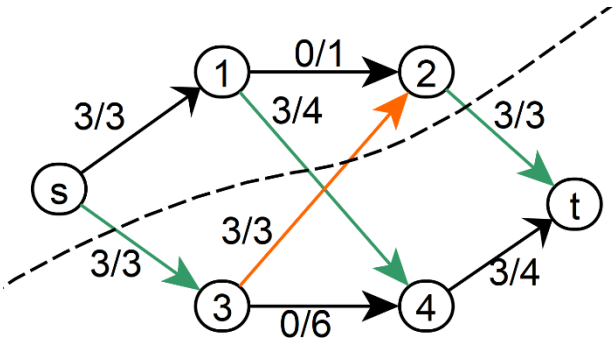


Figure 4

In Figure 4 is given the same cut as Figure 3 with an example flow added in the network. What is the flow across this cut?

First we sum the flow across the forward edges: f(s,3) + f(1,4) + f(2,t) = 3 + 3 + 3 = 9.

Second we sum the flow across the backward edges: f(3,2) = 3.

The flow across the cut is the difference between these 2 sums: f(*S*, *T*) = 9 – 3 = 6.

Remark 1: The value of flow f across any cut [*S*, *T*] does not exceed the capacity of that cut. Why?

Because all the flow from *s* has to reach to *t*, and for that, all the flow from *s* has to pass through the edges that start in *S* and end in *T*, and for each edge the flow must be smaller than its capacity.

Remark 2: The value of total flow |f| is equal to the value of the flow across any cut [*S*, *T*]. Why is that?

Remember that the total flow is defined as the flow created by *s* or, equivalently, the flow consumed by *t*. If there is no flow in the backward edges, since the flow created by *s* has to go to *t*, then the sum of the flow in all the forward edges must be equal to the total flow. If flow is added to the backward edges, this flow *must* come from the flow passed previously through the forward edges, and it *must* return eventually to *t*. Thus, for any unit of flow added to the backward edges a unit of flow must be added to the forward edges to compensate. So the difference between the flow of the forward edges and the flow of the backward edges gives us the flow through the forward edges if there was no flow in the backward edges. Which, as stated before is equal to the total flow.

Max-flow min-cut theorem: The maximum possible total flow of a graph is equal to the minimum possible capacity of a cut.

Why is that?

Since the flow across a cut is the total flow, that means that the flow across all cuts is the same. If the flow across some cut would be grater than the capacity of the minimum cut, then the flow across the minimum cut breaks the rules of the flow as mentioned by Remark 1 (a flow across a cut cannot be greater than the capacity of the cut). Therefore, the maximum flow cannot be greater than the minimum cut.

On the other hand, if the flow is lesser than the minimum cut it means that it can be augmented, so it is not the maximum flow. So The maximum flow is equal to the capacity of the minimum cut.

2. Max flow of minimum cost:

The problem: having a network on a weighted graph $N = (G=(V, E, w), s, t, c)$. We want to find the maximum flow with the minimum cost.

This problem can be interpreted in (at least) two ways, depending on how we define the cost of a flow:

Problem 1)

We have multiple ways to construct a flow that goes from the source to the sink and we want to construct only the cheapest paths that can carry the maximum flow.

Meaning that the cost of a flow is the sum of the costs of the edges for which the flow is >0:

$$w(f) = \sum_{(v_1,v_2)\in E, f(v_1,v_2)>0} w(v_1, v_2)$$

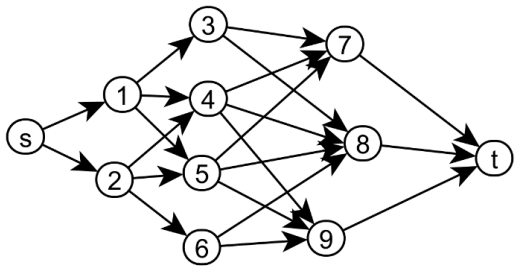


Figure 5

For example, we have a graph like the one in figure 5. It is easy to see that if the total flow is not much greater than the average capacity of an edge, there will be many different maximum flows.

So, the problem is how to find the edges to use for the maximum flow such that the sum of the weight of those edges is minimum.

How do we solve this?

Ford-Fulkerson but using a min-cost walk from *s* to *t*. This will guarantee to find the maximum flow with minimum cost if we set the correct costs of the edges in the residual graph.

What values do the edges in the residual network have?

a) Forward edges. For the forward edges for which the flow is 0 in the original graph we set the weight of the edge to the weight of the edge in the original graph – we want to say that if this edge would be used it will add this much to the cost.
   Forward edges the rest have flow higher than 0 in the original graph we set the weight to 0 – since this edge is already used (i.e. has some flow assigned to it) the cost of using this edge has already been counted, adding any more flow will not change the cost of the total flow.
b) Backwards edges. It is a bit more complicated here because the cost should change depending on how much flow will be adjusted for that edge in the original graph: if the flow will be reduced to 0 then the cost should be the negative weight (since the edge is

not used anymore, the cost should be "returned") but if the flow is not reduced to 0, that means that the edge is still used so the cost of the flow would stay the same – meaning the weight for the backward edge should be 0.
So, we have -weight if the flow is reduced to 0 and 0 otherwise. The problem is that we only know how much the flow is reduced *after* we find the augmenting path, since the cost is reduced by the residual capacity of the path (the minimum of all capacities in the path). And that would mean that normal minimum cost walk algorithms would not work, and we would have to use a brute-force or backtracking algorithm to find the path, which is inefficient.
On the other hand, we could use an heuristic which guarantees to find the maximum flow and finds the maximum cost most of the time, with very few exceptional cases: all backward edges have the cost set to 0, except the backward edges who have the minimum residual capacity of all edges in the residual graph (they are guaranteed then to be equal to the residual capacity of the path) – for which the weight is -weight

Problem 2)

We have a flow to transport from *s* to *t*, and transporting a unit of flow over an edge adds the cost equal to the weight of that edge.

This means that the cost of the flow will be the sum of the costs of the edges multiplied by the amount of flow:

$$\mathrm{w(f)} = \sum_{(v_1,v_2)\in E} \mathrm{w}(v_1, v_2) \times \mathrm{f}(v_1, v_2)$$

So, we want to find the maximum flow and the cheapest way to transport it from *s* to *t*.

To solve this problem, we need to first find a max flow of any cost and then improve it. After obtaining a max flow, we construct the residual network and set the costs as follows:

- The original cost on the forward edges

- The negative of the original cost on the backward edges

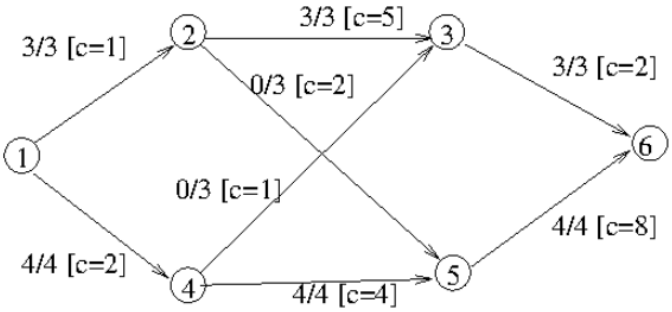
Theorem: A max flow of minimum cost does not have negative cycles in the residual network.

=> We look for a negative cycle. After finding it, we update the flow as if we had found an augmenting path, and then repeat until there are no more negative cycles.

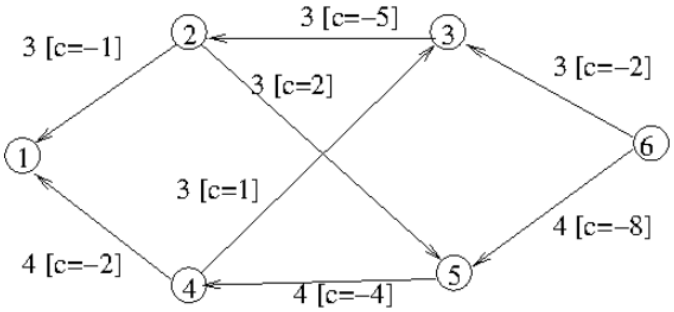
Reminder on how to update a flow:

- Find the minimum residual capacity of the edges in that cycle
- For the forward edges add to the flow that minimum residual capacity
- For the backward edges subtract from the flow that minimum residual capacity

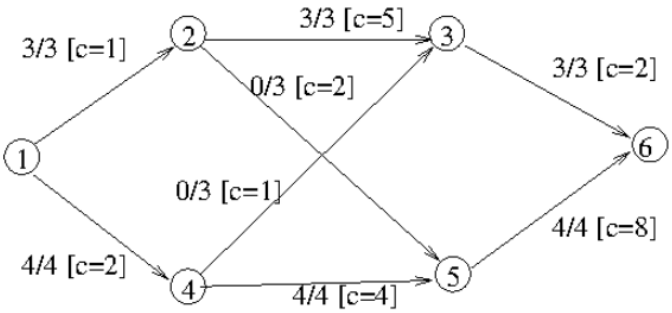
You can find an example of this process in Figure 6.



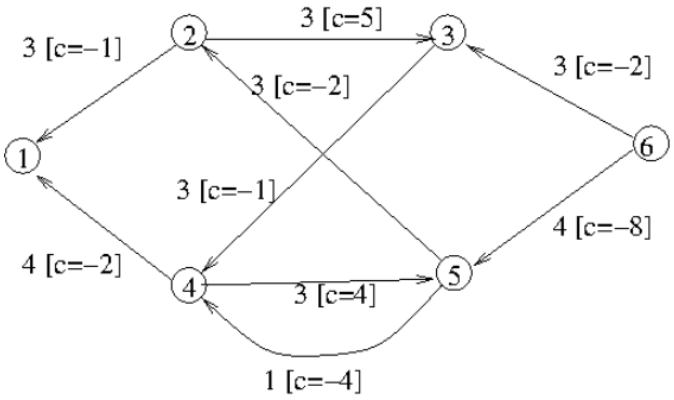
with residual graph



After using negative cost cycle 2, 5, 4, 3, 2 (capacity = 3, cost=-6):



with residual graph



No negative cost cycle can be found. Final flow=7 of cost=80-18=62.

Figure 6