Maximum Flow and its Applications: Part 2

Saladi Rahul, Sept. 7th 2023

1 (s,t)-cuts in a graph

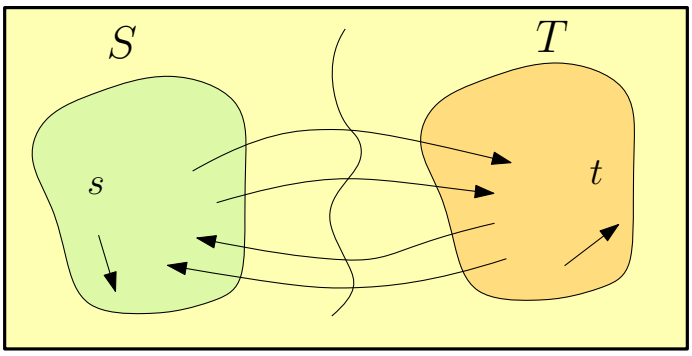


Figure 1: An (s,t)-cut.

An (s,t)-cut in a graph $G=(V,E)$ is a partition of V into two subsets S and T with the only constraints that $s \in S$ and $t \in T$. See Figure 2 for an example. For an edge $e=(u,v)$, let u be the *tail* of e and v be the *head* of e. The edges in G can be classified into four categories. Two of them are less interesting for us: the edges which have both its vertices in S and the edges which have both its vertices in T. The more interesting edges are:

1. the edges in E whose tail is in S and head is in T. Denote them by $E(S,T)$, and
2. the edges in E whose tail is in T and head is in S. Denote them by $E(T,S)$.

We will be interested in computing the *capacity* of an (s,t)-cut (S,T), which is defined as:

$$c(S,T) = \sum_{e \in E(S,T)} c_e.$$

Note that in the definition of capacity we have included the edges $E(S,T)$, but not the edges $E(T,S)$. Why is that? Think about it. To get an intuition, lets see Figure **??**. All the edges in this graph except one edge has a capacity of 100 units. Therefore, one would be tempted that a large amount of flow can be pushed on this graph. However, the maximum flow in this graph is *only* 10 units. At the same time, consider the cut (S,T) where $S=\{s,a,b,c\}$ and $T=\{d,e,f,t\}$. The capacity of the cut (S,T) is 10. So, there seems to be some connection between the capacity of a cut and the flow through the graph! We will formally establish the connection next. In that process, we will also establish the fact we wanted to prove from the previous lecture that the Ford-Fulkerson algorithm produces the maximum flow.

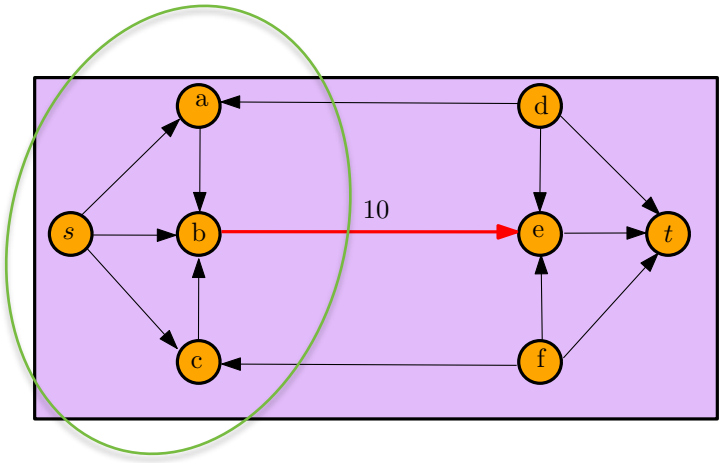


Figure 2: An extreme example where all the edges have capacity 100 units *except* for the edge (b, e) whose capacity is 10 units. As a result, the maximum flow in this graph is only 10 units.

2 The max-flow min-cut theorem

Theorem 1. *Let G be a graph with flow f. Then the following three statements are equivalent (i.e., either all the statements are true or none of them are true):*

1. *f is a maximum flow in G,*
2. *There is an (s, t)-cut in G whose capacity is equal to the value of f.*
3. *There is no path from source to sink in the residual graph G_f such that all edges on the path have positive residual capacity.*

Proof. We will prove the theorem in a cyclic manner. First, we will establish that $(2) \implies (1)$, then establish $(1) \implies (3)$, and finally establish $(3) \implies (2)$.

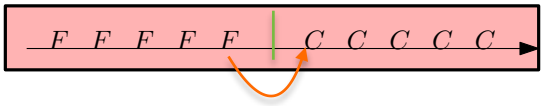


Figure 3: The figure illustrates that any flow in the graph is less than or equal to the capacity of any (s, t)-cut in the graph.

$(2) \implies (1)$. We will actually prove a more stronger result that:

ST: *Any flow in the graph is less than or equal to the capacity of any (s, t)-cut in the graph.*

See Figure 3. Once this fact is established, then it is easy to prove that $(2) \implies (1)$: Assume (2) is true but (1) is not true. Then there exists a flow value $f' > f$ in G. But this also implies that flow f' is strictly greater than the (s, t)-cut whose capacity is f, which is a contradiction. Therefore, $(2) \implies (1)$.

Now we will prove the statement **ST**. Consider any (s, t)-cut. Conceptually, if we remove the edges $E(S, T)$ (see the dotted edges in Figure 4(a)), then notice that exactly *zero* units of flow can be sent from source to sink. Now, if we "re-store" the edges $E(S, T)$, then what is the amount of

flow which can be sent now? See Figure 4(b). The amount of flow can be can be upper-bounded by:

$$\text{flow} \leq \sum_{e \in E(S,T)} c_e = \text{capacity of } (s,t)\text{-cut} = c(S,T).$$

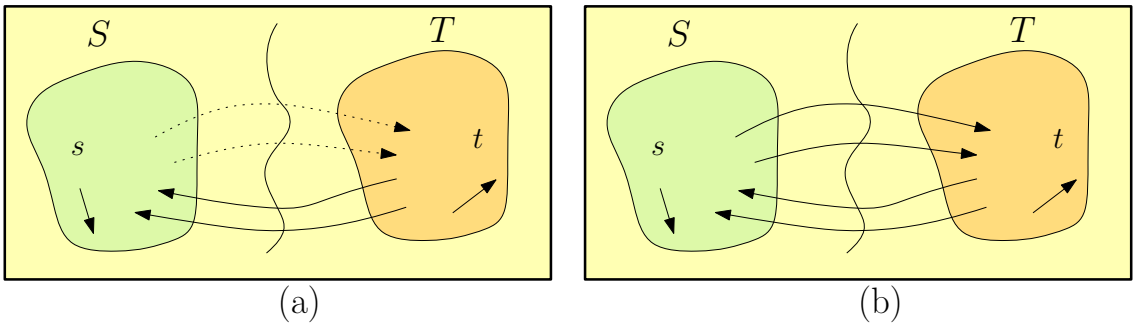


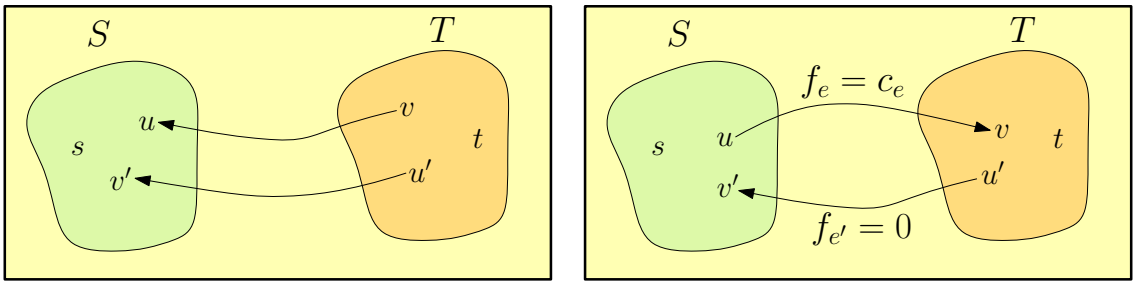
Figure 4: (a) The edges $E(S,T)$ removed conceptually, (b) The edges $E(S,T)$ re-stored.

(1) $\Longrightarrow$ (3). This is the easiest of the three statements to prove. Proving $(1) \implies (3)$ is the same as proving its contrapositive. Therefore, if there is a path from source to sink in the residual graph, then there is at lease one *additional* unit of flow which can be sent from source. This contradicts the fact that f is a maximum flow in G.

(3) $\Longrightarrow$ (2). We will (conceptually) perform a breadth first search on the residual graph starting from s. Let S be the set of vertices which are reachable from s. Trivially, $s \in S$ and since (3) is true, $t \notin S$. Let $T = V \setminus S$. Then (S,T) is a valid (s,t)-cut and enjoys a couple of nice properties:

- Consider any edge $e = (u,v) \in E(S,T)$. We claim that $f_e = c_e$. Why? Because if $f_e < c_e$, then the forward edge (say, e_f) of e in the residual graph would have residual capacity $c_e - f_e > 0$. This implies that the BFS from s could have used edge e_f to reach v, which contradicts the fact that v is not reachable from s. See edge e in Figure 5.
- Consider any edge $e' = (u',v') \in E(T,S)$. We claim that $f_{e'} = 0$. Why? Because if $f_{e'} > 0$, then the reverse edge (say, e'_r) of e' in the residual graph will have residual capacity $f_{e'} > 0$. This implies that the BFS from s could have used edge e'_r to reach u', which contradicts the fact that u' is not reachable from s. See edge e' in Figure 5.

Using these two facts, we will now prove formally that there is a flow f in the graph whose



(a) residual graph (b) original graph

Figure 5:

value is equal to the capacity of the (s,t)-cut (S,T).

$$\begin{aligned}
\text{flow value} &= \sum_{e\in\delta^-(s)} f_e \\
&= \sum_{e\in\delta^-(s)} f_e - \sum_{e\in\delta^+(s)} f_e + \sum_{v\in S\setminus\{s\}} \left(\sum_{e\in\delta^-(v)} f_e - \sum_{e\in\delta^+(v)} f_e \right) \\
&= \sum_{v\in S} \left(\sum_{e\in\delta^-(v)} f_e - \sum_{e\in\delta^+(v)} f_e \right) \\
&= \sum_{e\in E(S,T)} f_e - \sum_{e\in E(T,S)} f_e \quad \text{(Why? This was discussed in the class.)} \\
&= \sum_{e\in E(S,T)} c_e - \sum_{e\in E(T,S)} 0 \quad \text{(From the above two observations.)} \\
&= c(S,T).
\end{aligned}$$

This finishes the proof of the theorem. □

Remark 1. To prove the correctness of the Ford-Fulkerson algorithm, we only need the fact that $(3) \implies (1)$.

Remark 2. From the proof of the above theorem, it follows that the maximum flow in G is equal to the minimum capacity (s,t)-cut in G. This is the famous *max-flow min-cut* theorem.

Exercise. Given the maximum flow, in linear time how can you compute the minimum cut?

3 Running time of ford fulkerson algorithm

Let X be the *input size* to an algorithm $\mathcal{A}$. Then algorithm $\mathcal{A}$ is said to run in *polynomial time* if the its running time is X^c, where c is a constant (say, 1 or 10) independent of n. For the maximum flow problem, let the capacity of each edge be an integer in the range $[1, U]$. Then the input size for any algorithm solving the maximum flow problem is $X = O(m \log U)$, since we have m edges and

it requires $O(\log U)$ bits to describe the capacity of each edge. We want to answer the following question:

Does the Ford-Fulkerson algorithm run in polynomial time?

The answer turns out to be NO! See Figure 6. In the first iteration, if the (s,t)-path chosen is $s{-}v{-}w{-}t$, then only one unit of flow can be sent. As an exercise, draw the residual graph after the first iteration. In the second iteration, if the (s,t)-path chosen is $s{-}w{-}v{-}t$ (note that the residual graph will have this path), then only one unit of flow can be sent. In general, only one unit of flow can be sent in each iteration, if Ford-Fulkerson algorithm chooses the path $s{-}v{-}w{-}t$ in the odd iterations and the path $s{-}w{-}v{-}t$ in the even iterations. Therefore, the worst case running time of the algorithm is $O(|E|f^*)$, where f^* is the maximum flow. As such, Ford-Fulkerson is not a polynomial time algorithm.

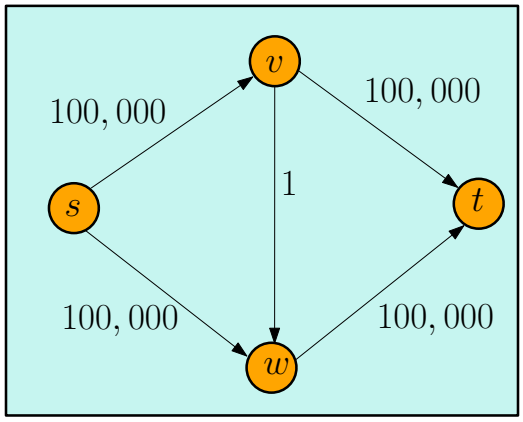


Figure 6: The flow value in this example is $200,000$. It turns out that if the (s,t) paths are not carefully chosen, then the Ford-Fulkerson algorithm will require $200,000$ iterations.