

Lecture 17

*Lecturer: Andre Wibisono**Scribe: Matthew Zhang*

1 Overview

This lecture continues discussing max flow. Specifically, these lecture notes cover:

- Ford-Fulkerson Algorithm
- Flows and Cuts
- Max-Flow Min-Cut Inequality

2 Flow Network Problem

First, recall the flow network we have been working with in the past few lectures.

- Directed graph $G = (V, E)$
 - source vertex $s \in V$
 - sink vertex $t \in V$
 - Assumption: no edges go into s , no edges go out of t , and in between s and t any edge is allowed
 - edge capacities $c_e > 0$ for each $e \in E$.
- We want to find a flow $f = (f(e))_{e \in E}$ that satisfies the following constraints.
 - [capacity]: $0 \leq f(e) \leq c_e$
 - [conservation]: $\forall v \in V \setminus \{s, t\}$

$$f^{in}(v) = f^{out}(v),$$

$$\text{where } f^{in}(v) = \sum_{e=(u,v) \in E} f(e) \text{ and } f^{out}(v) = \sum_{e'=(v,w) \in E} f(e').$$

The value of a flow f is defined to be

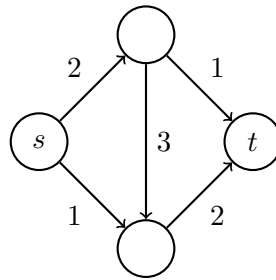
$$val(f) := f^{out}(s).$$

By the flow property, this is also equal to

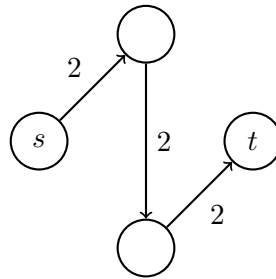
$$val(f) = f^{in}(t).$$

Our goal is to find the flow with maximum value (which we call a max flow).

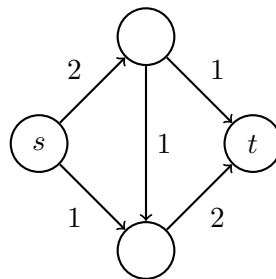
Consider the following example.



The following flow with value 2 would not be a max flow.



The following flow with value 3 would be the max flow.



3 Ford-Fulkerson Algorithm

Last time, we discussed the Ford-Fulkerson algorithm for finding the max flow.

Algorithm 1 Ford-Fulkerson(G, c)

```
1: function FORDFULKERSON( $G, c$ )
2:   init  $f = 0$ 
3:   residual graph  $G_f$ 
4:   while  $\exists s \rightarrow t$  path in  $G_f$  do
5:     choose an augmenting path  $P$  (a simple  $s \rightarrow t$  path in  $G_f$ )
6:      $f' = \text{Augment}(f, P)$ 
7:      $f \leftarrow f'$ 
8:      $G_f \leftarrow G_{f'}$ 
9:   end while
10:  return  $f$ 
11: end function
```

Properties of Augment:

- $f' = \text{Augment}(f, P)$

Then f' is a flow and we have that

$$\text{val}(f') = \text{val}(f) + \beta > \text{val}(f),$$

where $\beta = \text{bottleneck}(P) = \min_{e \in P} c_f(e) > 0$

- If all capacities c_e and flow values $f(e)$ are integers, then β is a positive integer and is at least 1. Then in each while loop, this means that

$$\text{val}(f') \geq \text{val}(f) + 1$$

Properties of FF: Suppose all $c_e \in \mathbb{N}$ for all edges $e \in E$.

- Then FF can be implemented in $O(m \cdot C)$ time, where C is the value of max flow.
- This is because there are at most C iterations of the while loop, and in each iteration, we can use DFS/BFS to find an augmenting path in $O(m)$ time.
- The running times of **Augment** and constructing G_f are also $O(m)$.
- Therefore, the running time of FF is $O(m \cdot C)$.

3.1 Optimality of Ford-Fulkerson

Question: How can we show that $f = \text{FF}(G, c)$ is a max-flow?

Answer: via the Max Flow - Min Cut theorem

4 Cuts

Let $G = (V, E)$ be a flow network with capacity $c_e > 0$.

Define: an $s - t$ cut is a partition of the vertices:

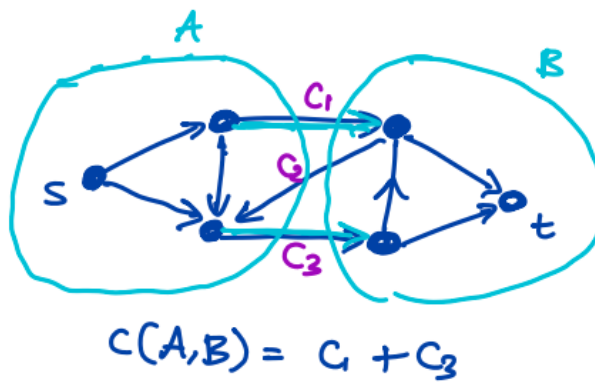
$$V = A \cup B$$

such that $s \in A, t \in B$, and $A \cap B = \emptyset$.

Define: the capacity of a cut (A, B) is:

$$c(A, B) = \sum_{\substack{e=(u,v) \in E \\ u \in A, v \in B}} c_e.$$

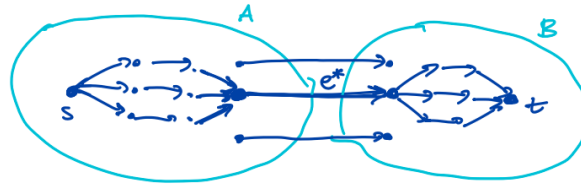
- Only count edges that cross $A \rightarrow B$
- Don't count internal edges ($A \rightarrow A, B \rightarrow B$)
- Don't count opposite edges ($B \rightarrow A$)



4.1 Intuition Behind Cuts

In general, cuts provide an upper bound to flow values. The intuition behind this is as follows.

- Suppose we can find a cut (A, B) such that

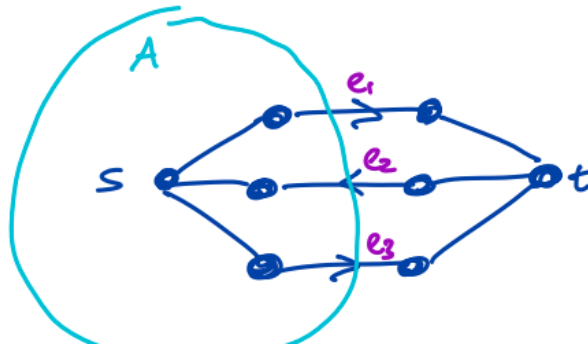


- Any flow from s to t must pass through e^* (or the other edges that cross the cut) from A to B .
- If e^* (along with the other edges) has small capacity $c_{e^*} \approx 0$, then we cannot push too much flow across this edge.
- If e^* (along with the other edges) has large capacity, maybe other cuts have small capacity.

5 Flows and Cuts

Let f be a flow and (A, B) be a cut.

Define: $f^{out}(A) = \sum_{\substack{e=(u,v) \in E \\ u \in A, v \in B}} f(e)$, $f^{in}(A) = \sum_{\substack{e=(u,v) \in E \\ u \in B, v \in A}} f(e)$



In the example above, we have that $f^{out}(A) = f(e_1) + f(e_3)$ and $f^{in}(A) = f(e_2)$. By definition, we also know that $f^{out}(A) = f^{in}(B)$ and $f^{in}(A) = f^{out}(B)$.

5.1 Value of Flows

Lemma: Let f be any flow and (A, B) be any cut. Then we have that

$$val(f) = f^{out}(A) - f^{in}(A) = f^{in}(B) - f^{out}(B).$$

Below are some of the properties of this lemma.

1. Recall that we defined $val(f) = f^{out}(s)$. We claim that the lemma agrees with this definition. Note that this is exactly the case where $(A, B) = (s, V \setminus \{s\})$. Thus, $f^{in}(s) = 0$ and so

$$val(f) = f^{out}(s) - f^{in}(s) = f^{out}(s)$$

as desired.

2. On the other hand, if we take $(A, B) = (V \setminus \{t\}, t)$, then $f^{out}(t) = 0$. Hence, the lemma yields

$$val(f) = f^{in}(t) - f^{out}(t) = f^{in}(t).$$

Now let us return to the proof of the lemma.

Proof. By definition, we know that $val(f) = f^{out}(s) - f^{in}(s)$ and $\forall v \in A, f^{out}(v) - f^{in}(v) = 0$ by the conservation property. Then we have that

$$val(f) = \sum_{v \in A} (f^{out}(v) - f^{in}(v)).$$

Observe that $f^{out}(v)$ can be expressed as $\sum_{e=(v,w)} f(e)$ and $f^{in}(v) = \sum_{e=(u,v)} f(e)$. Consider any edge $e \in E$. We have the following four cases.

1. If e is from A to A ($e = (u, v) : u \in A, v \in A$), then $f^{out}(u)$ has $+f(e)$ while $f^{in}(v)$ has $-f(e)$. This contributes 0 to $val(f)$.



2. If e is from A to B ($e = (u, v) : u \in A, v \in B$), then $f^{out}(u)$ has $+f(e)$ which contributes $f(e)$ to $val(f)$.



3. If e is from B to A ($e = (v, u) : v \in A, u \in B$), then $f^{in}(v)$ has $-f(e)$ which contributes $-f(e)$ to $val(f)$.



4. If e is from B to B ($e = (u, v) : u \in A, v \in B$), then there is no contribution to $val(f)$.



Then

$$\begin{aligned}
 val(f) &= \sum_{v \in A} (f^{out}(v) - f^{in}(v)) \\
 &= \sum_{\substack{e=(u,v) \\ u \in A, v \in B}} f(e) - \sum_{\substack{e=(u,v) \\ u \in B, v \in A}} f(e) \\
 &= f^{out}(A) - f^{in}(A).
 \end{aligned}$$

□

6 Max-Flow Min-Cut Inequality

Lemma: Let f be any flow and (A, B) be any cut. Then

$$val(f) \leq c(A, B).$$

Proof. By the previous lemma, we have that

$$val(f) = f^{out}(A) - f^{in}(A).$$

However, note that our assumption is that all flows are nonnegative; in other words, $f(e) \geq 0$. Then we can say that $val(f) \leq f^{out}(A)$. Furthermore, by the definition of $f^{out}(A)$ and the capacity constraint, we know that

$$val(f) \leq f^{out}(A) = \sum_{\substack{e=(u,v) \\ u \in A, v \in B}} f(e) \leq \sum_{\substack{e=(u,v) \\ u \in A, v \in B}} c_e.$$

Observe that this last summation is exactly defined to be $c(A, B)$, which proves our claim that $val(f) \leq c(A, B)$. \square

Note that this lemma is true for any flow and any cut. In particular, this is also true for the max-flow and the min-cut:

$$\max_{flow f} val(f) \leq \min_{cut(A, B)} c(A, B).$$

Moreover, if $val(f) = c(A, B)$, then f has maximum value and (A, B) has minimum capacity. This is because if there exists another flow f' with $val(f') > val(f) = c(A, B)$, this contradicts the lemma above. The same reasoning holds true for why (A, B) is a min-cut.

6.1 Ford-Fulkerson Max-Flow

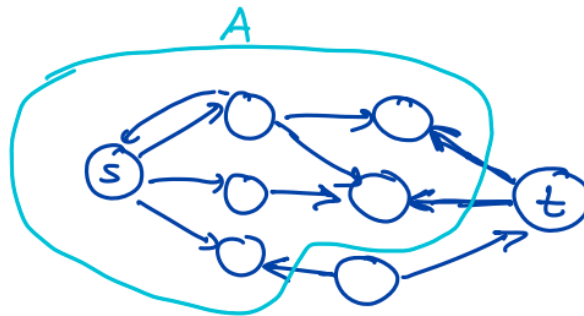
Returning to the Ford-Fulkerson algorithm, we wish to show that the FF algorithm returns a max-flow. This is equivalent to finding a cut (A^*, B^*) such that

$$val(f) = c(A^*, B^*).$$

Then by the max-flow min-cut inequality, f must have maximum value.

Theorem: FF returns a flow f with max value.

Proof. We know that FF terminates when there does not exist an $s \rightarrow t$ path in G_f , the residual graph. An example of this can be found below.



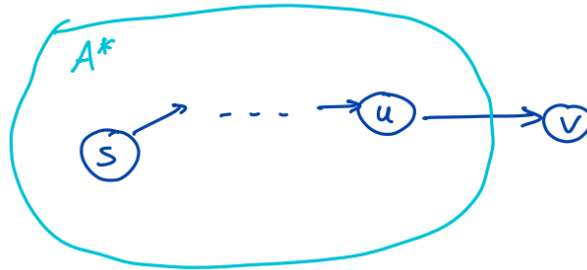
Now define cut (A^*, B^*) such that $A^* = \{v \in V : \exists s \rightarrow v \text{ path in } G_f\}$ and $B^* = V \setminus A^*$. Note that $s \in A^*, t \in B^*$, and $A^* \cap B^* = \emptyset$, which makes this a valid cut. We claim that

$$c(A^*, B^*) = val(f).$$

Proof. 1. All edges out of A^* are saturated by f :

$$\forall e = (u, v) \in E : u \in A^*, v \in B^*,$$

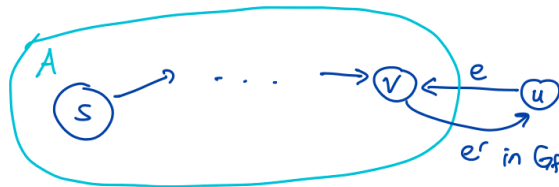
then $f(e) = c(e)$. This is because if $f(e) < c(e)$, then in the residual graph G_f , there would be a forward edge $e = (u, v)$ in G_f with capacity $c_f(e) = c_e - f(e) > 0$. However, this contradicts our definition of A^* because we can then reach $s \rightarrow u \rightarrow v$ in G_f .



2. All edges into A^* are not used:

$$\forall e = (u, v) \in E : u \in B^*, v \in A^*,$$

then $f(e) = 0$. This is because if $f(e) > 0$, then in G_f , there would be a backward edge $e' = (v, u)$ with capacity $f(e') > 0$. Then we can reach $s \rightarrow v \rightarrow u$ in G_f , which once again contradicts our definition of A^* .



Therefore, we know that

$$\begin{aligned} \text{val}(f) &= f^{\text{out}}(A^*) - f^{\text{in}}(A^*) \\ &= \sum_{\substack{e=(u,v) \\ u \in A^*, v \in B^*}} f(e) - \sum_{\substack{e=(v,u) \\ v \in B^*, u \in A^*}} f(e) \\ &= \sum_{\substack{e=(u,v) \\ u \in A^*, v \in B^*}} c_e \\ &= c(A^*, B^*). \end{aligned}$$

□

By the max-flow min-cut inequality, it follows that FF returns the maximum flow. □

This is the end of Lecture 17. The next lecture will continue with the applications of max flow such as bipartite matching.