CSE 347 Lecture 11

Max-Flow II: Max-Flow = Min-Cut

1. Running time of Ford-Fulkerson
2. Flows vs. cuts
3. Correctness of Ford-Fulkerson and corollaries

Last lecture, we introduced the Max-Flow problem: given a directed graph $G = (V, E)$ such that each edge $e \in E$ has a positive integer capacity $c(e)$, and a pair of vertices $s$ and $t$ in $G$, we wished to find a flow, an assignment $f : E \to \mathbb{R}^+$ that satisfied (1) $\forall e \ 0 \leq f(e) \leq c(e)$ (the "capacity constraint") and (2) $\forall v \ f^{in}(v) = f^{out}(v)$ where $f^{in}(v) = \sum_{e \text{ into } v} f(e)$ and $f^{out}(v) = \sum_{e \text{ out of } v} f(e)$ (the "conservation constraint"). In particular, we wanted a flow that maximized $f^{out}(s)$. This captured the problem of finding the most efficient utilization of the network to route vehicles/goods/packets/etc. from $s$ to $t$. (We also suggested that this problem captured the "difficult part" of many other problems, something we'll return to next time.)

We arrived at the following extension of the natural greedy algorithm for the problem:

Ford-Fulkerson($G$)

1. Initialize $f(e) \leftarrow 0$ for all $e$, $G_f \leftarrow G$
2. While there is a simple $s$-$t$ path $P$ in $G_f$
   a. Define $f^{in}(s) = f^{out}(t) = 0$ and $f^{in}(v) = f^{out}(v) = 0$ for all $v \not\in P$
   b. For each edge $e$ in $P$, if $e \in E$, let $f(e) = \min\{c(e) - f^{in}(u) + f^{out}(u) : (u, v) = e\}$
   c. For each edge $e$ in $P$, if $e \not\in E$, let $f(e) = \min\{f^{in}(u) - f^{out}(u) : (u, v) = e\}$
   d. Update $G_f$ by inflating the capacities of edges in $P$
3. Return $G_f$
\[ \text{Put } f \leftarrow \text{augment}(f, P) \]
\[ \text{Put } G_f \leftarrow \text{the residual graph of } G \text{ for } f \]
\[ \text{Return } f. \]

Where the residual graph \( G_f \) of \( G \) for \( f \) is a graph with the same vertex set as \( G \), capacity \( c(e) - f(e) \) for each edge of \( G \) (and the edge \( e \) whenever this is non-zero), and a new backwards edge in the opposite direction of \( e \) with capacity \( f(e) \).

The backwards edges represent our ability to "undo" assignments of flow in \( f \) by redirecting it using \( \text{augment}(f, P) \), which takes bottleneck \((P, f)\) - the minimum capacity of any edge of \( G_f \) traversed by \( P \) - and (1) adds this amount to \( f(e) \) for each forwards edge \( e \) and (2) subtracts this amount from \( f(e) \) for each edge of \( P \) that is the backwards edge corresponding to \( e \). We proved

\underline{Lemma 1}: \( \text{augment}(f, P) \) returns a flow \( f' \) in \( G \)

\underline{Lemma 2}: At every iteration, both \( f \) and the capacities of \( G_f \) are integers.

\underline{Lemma 3}: For a flow \( f \) and simple \( s-t \) path \( P \) in \( G_f \), \( f' = \text{augment}(f, P) \) has value \( v(f') = v(f) + \text{bottleneck}(P, f) \)

and finally, for \( C = \sum_{e \text{ out of } s} c(e) \),

\underline{Lemma 4}: The while loop in Ford-Fulkerson runs for at most \( C \) iterations.

**Proof** \( v(f) = f^{\text{out}}(s) = \sum_{e \text{ out of } s} f(e) \). Since \( f(e) \leq c(e) \), \( v(f) \leq C \).
Therefore, since by Lemma 3 \( v(f) \geq \# \text{iterations} \), we can conclude that

Let's now assume further that, with no real loss in generality, every vertex \( v \in V \) has at least one edge incident to it. (We can discard isolated vertices)

Theorem: Ford-Fulkerson can be implemented to run in \( O(\sqrt{E} \cdot C) \) time.

Proof: Lemma 4 bounds the number of iterations of the loop by \( C \), so it suffices to show how each iteration can be implemented to run in time \( O(\sqrt{E}) \).

We'll represent \( G_f \) by storing linked lists of edges in and out (respectively) of each vertex. Note that \( G_f \) has at most \( 2\sqrt{E} \) edges—one for each forwards and backwards version of each edge of \( G \). Thus, we can find a simple path \( P \) from \( s \) to \( t \) in \( G_f \) in time \( O(\sqrt{V} + \sqrt{E}) \) using depth-first search; since we assumed/ensured that each vertex had at least one edge incident to it, \( |V| \leq 2\sqrt{E} \), so this is indeed \( O(\sqrt{E}) \). Likewise, since \( P \) is simple, it has length at most \( |V| \), so scanning it to compute bottleneck and then augment likewise takes time \( O(\sqrt{E}) \). Finally, to generate \( G_f \) from \( G \) and \( f \), we simply need to iterate over the edges of \( G \), and add the corresponding forward and backwards edges to the lists in \( G_f \). This takes time \( O(\sqrt{E}) \) as well.

Flows and cuts
Ultimately, we'll prove the correctness of Ford-Fulkerson by appealing to a structural bound. Specifically, an \( s \)-\( t \) cut is a partition of \( V \) into sets \( A \) and \( B \) such that \( s \in A \) and \( t \in B \). The capacity of the cut \( (A,B) \), which we'll denote by \( c(A,B) \), is \( \sum_{e \in A \to B} c(e) \).

**Lemma 5:** For any flow \( f \) and \( s \)-\( t \) cut \( (A,B) \), \( v(f) = f^{\text{out}}(A) - f^{\text{in}}(A) \).

**Proof:** Recall \( v(f) = \sum_{e \in \text{out of } s} f(e) \), and that the conservation constraints guarantee that for every \( v \in S, t \in S \), \( \sum_{e \in \text{in to } v} f(e) = \sum_{e \in \text{out of } v} f(e) \). So,

\[
v(f) + 0 = \sum_{e \in \text{out of } S} f(e) - \sum_{e \in \text{in to } S} f(e) + \sum_{e \in \text{in to } V \setminus S} f(e) - \sum_{e \in \text{out of } V \setminus S} f(e) - \sum_{e \in \text{empty}} f(e)
\]

Let's rearrange this: we pair up terms with both endpoints in \( A \), and pull out the terms leaving and entering \( A \):

\[
\sum_{e \in \text{out of } A \text{ from } V \setminus A} f(e) - \sum_{e \in \text{in to } A \text{ to } V \setminus A} f(e) + \sum_{e \in \text{in to } V \setminus A \text{ to } V \setminus B} f(e) - \sum_{e \in \text{out of } V \setminus A \text{ from } V \setminus B} f(e) = f^{\text{out}}(A) - f^{\text{in}}(A)
\]

That is, intuitively, the value of a flow is precisely the net flow across any cut that separates \( s \) from \( t \).

Using the fact again that edges leaving \( A \) enter \( B \) and vice versa, we similarly find that \( v(f) = f^{\text{in}}(B) - f^{\text{out}}(B) \).

(Letting \( B = \{t\} \), it also follows that \( v(f) = \sum_{e \in \text{in to } t} f(e) \), which is intuitive and helps justify our choice of objective \( v(f) \).)

Now, we come to the key structural bound.

**Lemma 6:** For any \( s \)-\( t \) cut \( (A,B) \) and any flow \( f \), \( v(f) \leq c(A,B) \).

**Proof:** By Lemma 5, \( v(f) = f^{\text{out}}(A) - f^{\text{in}}(A) \). Since \( f^{\text{in}}(A) \geq 0 \),

\[
v(f) \leq f^{\text{out}}(A) = \sum_{e \in \text{out of } A} f(e) \leq \sum_{e \in \text{out of } A} c(e) \text{ by the capacity constraints.}
\]

But, \( \sum_{e \in \text{out of } A} c(e) = c(A,B) \) by definition
That is: the capacity of every cut bounds the maximum flow.

**Correctness of Ford-Fulkerson and Corollaries**

Again, we’ll prove correctness of the greedy-like Ford-Fulkerson algorithm by showing that the flow it finds has value matching the upper bound of Lemma 6. Actually, since Ford-Fulkerson will then give a construction of flows matching the minimum cut value, this proves

**Theorem** $\max_{\text{flow } f} v(f) = \min_{(A^*, B^*)} c(A^*, B^*)$ ("Max-Flow = Min-Cut")

So, here is the main technical step:

**Lemma 7**: Let $f$ be an $s$-$t$ flow such that there is no path from $s$ to $t$ in the residual graph $G_f$. Then there is a $s$-$t$ cut $(A^*, B^*)$ such that $v(f) = c(A^*, B^*) = \min_{(A^*, B^*)} c(A^*, B^*)$.

**Proof**: Let $A^*$ be the set of all nodes $v \in V$ such that there is a $s$-$v$ path in $G_f$. Notice that, by hypothesis, $t \notin A^*$, so if $B^* = V - A^*$, $(A^*, B^*)$ is indeed a $s$-$t$ cut.

**Claim**: For each edge $e$ from $A^*$ to $B^*$, $f(e) = c(e)$.

**Proof of claim**: By contrapositive: if $f(e) \neq c(e)$, $f(e) < c(e)$, so the edge $e$ is present in $G_f$ with capacity $c(e) - f(e) > 0$. Therefore, if $e = (u, v)$ with $u \in A^*$, there is a path from $s$ to $u$ in $G_f$ by hypothesis, which can be extended to a path to $v$ by traversing $e$. So, $v \in A^*$ also. Hence, if $e = (u, v)$ with $u \in A^*$ and $v \in B^*$, $c(e) = f(e)$. \[\Box\]

**Claim**: if $e' = (u', v')$ such that $u' \in B^*$ and $v' \in A^*$, $f(e') = 0$.\[\Box\]
Proof of Claim: If \( f(e') > 0 \), then \( G_f \) would contain a backwards edge \((v', u')\) with capacity \( f(e') \). Hence, the \( s-v' \) path \((s, v' \in A^*)\) could again be extended to obtain a \( s-u' \) path, which would place \( v' \) in \( A^* \), not \( B^* \).

In conclusion, we find by Lemma 5:
\[
v(f) = f^{\text{out}}(A^*) - f^{\text{in}}(A^*) = \sum_{e \in \text{out of } A^*} f(e) - \sum_{e \in \text{into } A^*} f(e) = \sum_{e \in \text{out of } A^*} c(e) - 0 = c(A^*, B^*)
\]

In conclusion, since the lemma applies when the loop terminates.

Theorem Ford-Fulkerson returns a flow \( f \) of maximum value. Furthermore, observe that by running breadth-first search in the residual graph \( G_f \) obtained on the final iteration of Ford-Fulkerson, we can find the cut \((A^*, B^*)\) in \( O(\text{E}) \) time.

Corollary There is an algorithm that finds a cut of minimum capacity in time \( O(\text{E}^2) \).

Finally, here is one more significant corollary:

Corollary Ford-Fulkerson returns an integer-valued flow \( f \) that attains the maximum value in time \( O(\text{E}^2) \). (By Lemma 2, \( f \) is integer valued, and Ford-Fulkerson solves Max-Flow in time \( O(\text{E}^2) \).)

So, not only is there an integer-valued maximum flow, but Ford-Fulkerson finds one efficiently. (Note that not all maximum flows must be integer-valued.) We'll see that this property is very useful in solving other seemingly hard optimization problems.