CS161 Max Flow, Min Cut
Scribe: Chuanqi Shen (2015), Virginia **Date:** May 27, 2016

1 History of Flows and Cuts

Today we will continue the theme of studying cuts in graphs. In particular we will be studying a very interesting problem called the max flow problem. Before that, a short history lesson. During the Cold War, the US Air Force at that time was very interested in the Soviet train networks. In reports that were declassified in 1999, it was revealed that the Air Force collected enough information about the train network that they were able to determine how resources were shipped from the Soviet Union to Europe. The Air Force was very interested in determining how much resources can be transported from the Soviet Union to Europe, and what needed to be done to destroy this movement of resources. What this translates to is the min cut problem i.e. cut the minimum number of train tracks so that nothing goes to Europe. Here, cutting an edge means dropping a bomb. Nowadays however, there are much milder applications of this problem, for instance, understanding the flow of information through the Internet.

2 Formulation of max flow

You are given an input graph $G = (V, E)$, where the edges are directed. There is a function $c : E \to \mathbb{R}^+$ that defines the capacity of each edge. We also label two nodes, s and t in G, as the source and destination respectively. The task is to output a *flow* of *maximum value.* We will shortly define what a *flow* is and what a flow of *maximum value* means.

A flow f is a function $f : E \to \mathbb{R}_0^+$ such that

1. (Capacity Constraint)
$$\forall (u, v) \in E, 0 \le f(u, v) \le c(u, v)$$
.
2. (Flow Conservation Constraint)
$$\forall v \in V \backslash \{s, t\}, \sum_{x \in N_{in}(v)} f(x, v) = \sum_{y \in N_{out}(v)} f(v, y)$$
Here $N_{in}(v)$ denotes the set of nodes with an edge that points to v and $N_{out}(v)$ denotes the set of nodes that v points to.

Suppose that there are no edges going into s and no edges coming out of t, From the above, you can verify yourself that $\sum_{x \in N_{out}(s)} f(s, x) = \sum_{y \in N_{in}(t)} f(y, t)$. We define the value $\sum_{x \in N_{out}(s)} f(s, x)$ to be the value of the flow f. We usually denote the value of a flow f as $|f|$. If there are edges into s and out of t, then the value of f is
$$|f| = \sum_{x \in N_{out}(s)} f(s, x) - \sum_{y \in N_{in}(s)} f(y, s).$$
The max flow problem is to find some flow f such that $|f|$ is maximized.

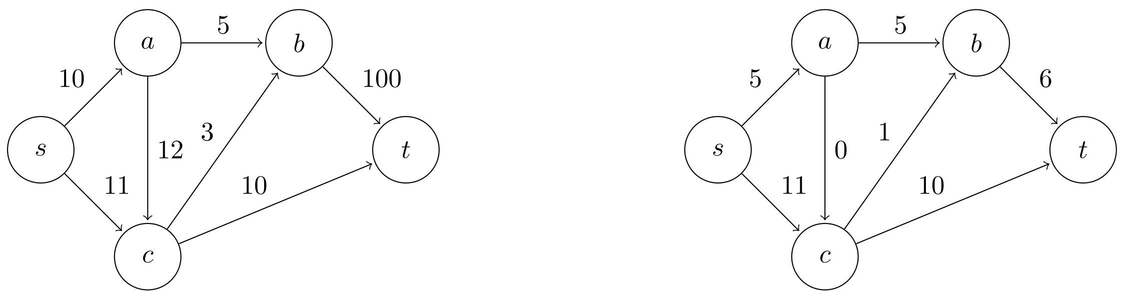

Figure 1: (Left) Graph G with edge capacities (Right) Graph G with a sample flow

3 Example

In Figure 1, we have a graph G and a sample flow f. Observe that the two constraints for a flow are satisfied. There can be multiple other flows possible that can satisfy the constraints. For our given flow, $|f| = 16$. The max flow for this graph is actually 18, as we will see shortly.

Here in this case, there is no edge into s. However, when there can be edges into s, we can define $|f|$ more generally as

$$|f| = \sum_{x \in N_{out}} f(s,x) - \sum_{y \in N_{in}(s)} f(y,s)$$

4 Formulation of min cut

Now, we give a formulation of min cut. We have previously introduced min cuts, but we will now introduce another definition of min cut defined for directed graphs in terms of source and destination nodes s and t.

An $s-t$ cut is a partition $V = S \bigcup T$ where S and T are disjoint and $s \in S, t \in T$, and the size/cost of an $s-t$ cut is

$$\|S,T\| = \sum_{x \in S, y \in T} c(x,y)$$

For our graph G shown above, if we set $S = \{s,a,c\}$ and $T = \{b,t\}$, then the cost of the cut is $c(a,b) + c(c,b) + c(c,t) = 5 + 3 + 10 = 18$. If we take another cut $S' = \{s,c\}, T' = \{a,b,t\}$, then $\|S',T'\| = c(s,a) + c(c,b) + c(c,t) = 10 + 3 + 10 = 23$. Note that we **do not** consider the edge $\{a,c\}$ as it is in the wrong direction (we only consider edges from S' to T').

5 The max flow min cut theorem

Lemma 5.1. *For any flow f and any $s-t$ cut (S,T) of G, we have $|f| \leq \|S,T\|$. In particular, the value of the max flow is at most the value of the min cut.*

Proof.

$$
\begin{aligned}
|f| &= \sum_{x \in N_{out}(s)} f(s,x) - \sum_{y \in N_{in}(s)} f(y,s) \\
&= \sum_{v \in S} \left(\sum_{x \in N_{out}(v)} f(v,x) - \sum_{y \in N_{in}(v)} f(y,v) \right) \text{ by the Flow Conservation Constraint all added terms sum to 0} \\
&= \sum_{v \in S} \left(\sum_{x \in N_{out}(v) \cap S} f(v,x) - \sum_{y \in N_{in}(v) \cap S} f(y,v) \right) + \sum_{v \in S} \left(\sum_{x \in N_{out}(v) \cap T} f(v,x) - \sum_{y \in N_{in}(v) \cap T} f(y,v) \right) \\
&= \sum_{v \in S} \left(\sum_{x \in N_{out}(v) \cap T} f(v,x) - \sum_{y \in N_{in}(v) \cap T} f(y,v) \right) \text{ first term sum to 0 because everything cancels out} \\
&\leq \sum_{v \in S, x \in T, x \in N_{out}(v)} f(v,x) \leq \sum_{v \in S, x \in T, x \in N_{out}(v)} c(v,x) = \|S,T\|
\end{aligned}
$$

□

We get the following consequence.

Corollary 5.1. *If we can find f and (S,T) such that $|f| = \|S,T\|$, then f is a max flow and (S,T) is a min cut.*

It turns out that we can always find such f and (S,T) for any graph.

Theorem 5.1 (Max flow-Min cut theorem)**.** *For any graph G, source s and destination t, the value of the max flow is equal to the cost of the min cut.*

We will show this by coming up with an algorithm. The algorithm will take the graph G and some flow f that has already been constructed, and create a new graph that is called the residual graph. In this new graph, the algorithm will try to find a path from s to t. If no such path exists, we will show that the value of the flow we started with is maximum. If not, we show how to increase the value of our flow by pushing some flow on that path.

6 Algorithm

We will make an assumption on our graph. The assumption can be removed, but it will make our lives easier. We will assume that for all $u, v \in V$, G does not have both (u,v) and $(v,u) \in E(G)$. We can make this condition hold by modifying the original graph. If $(u,v), (v,u) \in E(G)$, we split the edge (u,v) to two edges (u,x) and (x,v), where x is a new node we introduce into the graph. This makes the number of nodes at most $m + n$.

Now, let f be a flow given to us. We will try to see if we can improve this flow. We will define the *residual capacity* $c_f : V \times V \to \mathbb{R}_0^+$ as follows.

$$
c_f(u,v) = \left\{ \begin{array}{cl} c(u,v) - f(u,v) & \text{if } (u,v) \in E(G) \\ f(v,u) & \text{if } (v,u) \in E(G) \\ 0 & \text{otherwise} \end{array} \right\}
$$

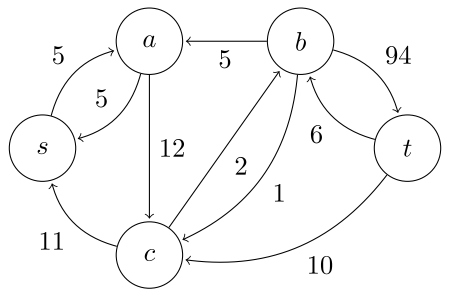

Figure 2: residual network

Basically, what this does is that, if there is any flow through the edge, you remove the flow from the capacity and add an edge in the opposite direction with the value of the flow. The reason we do this is because the flow we picked thus far might not be the correct flow, and this formulation allows us to undo changes that we have done.

We define G_f to be a residual network defined with respect to f, where $V(G_f) = V(G)$ and $(u, v) \in E(G_f)$ if $c_f(u, v) > 0$. Figure 2 shows G with the residual edges.

We will show that, if there is a path from s to t in G_f, then f is not a max flow. If no such path exists, that f is max flow.

Lemma 6.1. *If t is not reachable from s is G_f, then f is a maximum flow.*

Proof. Let S be the set of nodes reachable from s in G_f and $T = V \backslash S$. There are no edges in G_f from S to T since t is not reachable from s. Note that (S, T) defines an $s - t$ cut. Now consider any $v \in S, w \in T$. We have $c_f(v, w) = 0$ since (v, w) is not an edge in G_f. There are three cases

1. If $(v, w) \in E$, then by definition $c_f(v, w) = c(v, w) - f(v, w) = 0 \implies c(v, w) = f(v, w)$.
2. If $(w, v) \in E$, then $c_f(v, w) = f(w, v) = 0$.
3. If $(v, w) \notin E, (w, v) \notin E$, then we can disregard (v, w) and (w, v) since they do not appear in any flow or cut.

Using this, and the proof in Lemma 5.1, we have

$$\begin{aligned}
|f| &= \sum_{v \in S} \left(\sum_{x \in N_{out}(v) \cap T} f(v, x) - \sum_{y \in N_{in}(v) \cap T} f(y, v) \right) \\
&= \sum_{v \in S} \sum_{x \in N_{out}(v) \cap T} f(v, x) \text{ (because from case 2 the second term above is 0)} \\
&= \sum_{v \in S} \sum_{x \in N_{out}(v) \cap T} c(v, x) \text{ (from case 1)} \\
&= \|S, T\|
\end{aligned}$$

Thus, we show that the flow is equal to the cut. From Corollary 5.1 we know that f is a maximum flow, and $\|S, T\|$ is a min cut. ☐

Lemma 6.2. *If G_f has a path from s to t, we can modify f to f' such that $|f| < |f'|$.*

Proof. Pick a path P from s to t in G_f, and consider the edge of minimum capacity on the path. Let that capacity be v. Then we can increase our flow by v. For each edge in P, if $c_f(v,w)$ is the right direction (i.e there is an edge $(v,w) \in E(G)$), then we can increase our flow on this edge by v. If $c_f(v,w)$ is in the opposite direction (i.e. $(w,v) \in E(G)$), then we can decrease the flow on this edge by v. In effect, we are "undoing" the flow on this edge. By doing so, we have increased our flow by v.

As an example, Figure 2 again. The path $s \to a \to c \to b \to t$ is a path with minimum capacity 2. Therefore, we can update our flow and push an additional 2 units of flow, resulting in a flow of 18.

Formally, Let $s = x_0 \to x_1 \to ... \to x_k = t$ be a path P in G_f, and let $F = \min_i c_f(x_i, x_{i+1})$. Define a new flow f' where

$$f'(u,v) = \left\{ \begin{array}{c} f(u,v) + F \text{ if } (u,v) \in P \\ f(u,v) - F \text{ if } (v,u) \in P \\ f(u,v) \quad \text{otherwise} \end{array} \right\}$$

We now need to show that f' is a flow. The capacity constraints are satisfied because

1. If $(u,v) \in P$, then $0 \le f(u,v) + F \le f(u,v) + c_f(u,v) = f(u,v) + c(u,v) - f(u,v) = c(u,v)$
2. If $(v,u) \in P$, then $f(u,v) - F \le f(u,v) \le c(u,v)$ and $f(u,v) - F \ge f(u,v) - c_f(u,v) = 0$.
3. Otherwise, $f(u,v)$ is from the original flow f.

The conservation constraints are also satisfied: Suppose that P is simple. Thus, for every $v \in V$, P uses 0 or two edges incident on v. If P uses 0 edges on v, then the edge flow values incident on v have not changed when going from f to f'. Thus, suppose that P uses two edges (x,v) and (v,y) incident on v. Because in G_f some edges appear in the opposite direction compared to G, we need to consider a few cases.

1. (x,v) and (v,y) are both in the same direction (an edge into v and an edge out of v); the flow into v increases by F and the flow out of it also increases by F
2. (x,v) and (v,y) are both in the opposite direction (an edge opposite to one into v and an edge opposite to one out of v); the flow into v decreases by F and the flow out of it also decreases by F
3. (x,v) is in the correct direction and (v,y) is in the opposite direction. Then the flow into V changes by $F - F = 0$.
3. (x,v) is in the opposite direction and (v,y) is in the correct direction. Then the flow out of V changes by $F - F = 0$.

Finally, $|f'| > |f|$ because we just increase our flow by F, and by our definition of G_f, it must be that $F > 0$. □

From this, we can construct an algorithm to find the maximum flow. Starting with some arbitrary flow of the graph, construct the residual network, and check if there is a path from s to t. If yes, update the flow, construct the new residual graph and repeat. Otherwise, we have found the max flow.

A path from s to t in the residual graph is called an *augmenting* path, and pushing flow through it to modify the current flow is referred to as *augmenting* along the path.

Here is the max flow algorithm attributed to Ford-Fulkerson.

The runtime of this algorithm is bounded by the number of times we update our flow. If edge capacities are all integers, then we can increase the flow by at least 1 each time we update our flow. Therefore, the runtime if $O(|f|m)$ where $|f|$ is the value of the max flow. If we have rational edge capacities, then we can multiply all edge capacities by a factor to make them all integers. However, the runtime blows up by a factor as well.

Algorithm 1: maxflow(G, s, t)

```
$f \leftarrow$ all zeroes flow;
$G_f \leftarrow G$;
while $t$ is reachable from $s$ in $G_f$ (check using DFS) do
    $P \leftarrow$ path in $G_f$ from $s$ to $t$;
    $F \leftarrow$min capacity on $P$;
    $f \leftarrow f'$ as defined in Lemma 6.2;
return $f$;
```

If we have irrational edge capacities, then the algorithm is no longer guaranteed to terminate. So we have a problem.

We will save the day in the next lecture.