CMPUT 675: Topics in Combinatorics and Optimization **Fall 2016**

Lecture 2 (September 9): Network Flow

Lecturer: Zachary Friggstad *Scribe: Bradley Hauer*

2.1 Network Flow

This lecture introduces the concept of a *flow network*. At the most basic level, a flow network can be viewed as an edge-weighted directed graph with all weights being non-negative real numbers. Further, distinct vertices s, t are designated the *source* and *sink*. This definition will lead to some interesting problems, which will in turn lead us to some new theorems and algorithms.

Preliminary notation. For a directed graph $\mathcal{G} = (V, E)$ we let

- For any $U \subseteq V$, $\delta^{out}(U) = \{uv \in E : u \in U, v \notin U\}$ is the set of edges leaving U.
- For any $U \subseteq V$, $\delta^{in}(U) = \{uv \in E : u \notin U, v \in U\}$ is the set of edges entering U.
- For any $v \in V$, $\delta^{out}(v) := \delta^{out}(\{v\})$ and $\delta^{in}(v) := \delta^{in}(\{v\})$.
- For any $g : E \to \mathbb{R}_{\geq 0}$ and any $S \subseteq E$, $g(S) = \sum_{e \in S} g(e)$.

Definition 1 *Let $\mathcal{G} = (V, E)$ be a directed graph, $\mu : E \to \mathbb{R}_{\geq 0}$ be a* **capacity function** *on the edges, and $s, t \in V$, $s \neq t$. A* **flow** *is a mapping $f : E \to \mathbb{R}_{\geq 0}$ which satisfies both of the following:*

1. $f(\delta^{in}(v)) = f(\delta^{out}(v)) \quad \forall v \in V - \{s, t\}$.
2. $f(e) \leq \mu(e) \quad \forall e \in E$

Intuitively, flow networks model problems where we need to transport some divisible quantity from a starting point, through multiple connected junctions or nodes, to an ending point. For example, we might need to route water from a water source, perhaps a river, to a destination, perhaps a water treatment plant. A pipe system is in place connecting the source to the destination, and there are junctions between the pipes where the flow can be controlled and routed. However, each section of pipe has a set capacity, an upper limit on how much water can flow through it. A flow, then, is a way of routing water from the source, to the destination.

Condition 1, flow conservation, can be informally stated as "flow in equals flow out". Whatever we are transporting, if it leaves the source, it must reach the target, and a node cannot send out more of something than it takes in (only the source can "produce" anything).

Condition 2 simply ensures that the capacities on each edge are respected. Going back to the water example, a pipe has a fixed width, and so can only carry a certain quantity of water.

Definition 2 *The* **value** *of a flow f, denoted* $\text{val}(f)$*, is $f(\delta^{out}(s)) - f(\delta^{in}(s))$.*

If no edges enter s, as is often the case, this can be simplified to $f(\delta^{out}(s))$.

In words, the value of a flow the amount of flow that is *created* at the source s. Note that by condition 1 in the definition of a flow, this is also the quantity that is *lost* at the sink t. In the water piping example, this is the amount of water which reaches the water treatment plant from the water source.

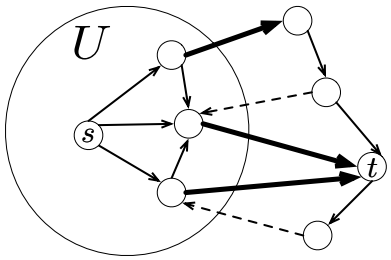


Figure 2.1: An $s-t$ cut U. The thick edges form $\delta^{out}(U)$ and their total capacity is the capacity of the cut. The dashed edges form $\delta^{in}(U)$ and do not contribute to the cut's capacity.

2.2 Flows and Cuts

There is another useful concept which arises in the study of flow networks.

Definition 3 *An $s-t$* **cut** *is a set $U \subseteq V$ with $s \in U$, $t \notin U$. The* **capacity** *of an $s-t$ cut U is $\mu(\delta^{out}(U))$.*

If we think of the vertices in a flow as being points on the plane, a cut is simply the set of vertices in a circle (or any other shape) containing s but not t (e.g. Figure 2.1). The capacity of a cut U is the total capacity of the edges which exit U.

Lemma 1 *Let f be a flow and U an $s-t$ cut. Then* $\text{val}(f) \leq \mu(\delta^{out}(U))$.

Proof.

$\text{val}(f)$	$=$	$f(\delta^{out}(s)) - f(\delta^{in}(s))$	By definition.
	$=$	$\sum_{v \in U} f(\delta^{out}(v)) - f(\delta^{in}(v))$	By flow conservation and $t \notin U$
	$=$	$f(\delta^{out}(U)) - f(\delta^{in}(U))$	The sum counts the flow of each edge across U in each direction exactly once; edges with both endpoints in U "cancel out".
	$\leq$	$\mu(\delta^{out}(U)) - 0$	By condition 2, and because flow is non-negative.

■

So, every cut's capacity provides an upper bound on the value of every flow. The rest of this lecture proves the following and describes an algorithm for finding a maximum-flow and minimum-cut.

Theorem 1 (Max-Flow/Min-Cut Theorem) *Let f be a maximum-value flow and U a minimum-capacity $s-t$ cut. Then* $\text{val}(f) = \mu(\delta^{out}(U))$.

Another important lesson concerns integrality of flows.

Theorem 2 *If all capacities are integers, then there is a maximum flow f with $f(e) \in \mathbb{Z}_{\geq 0}$ for each $e \in E$.*

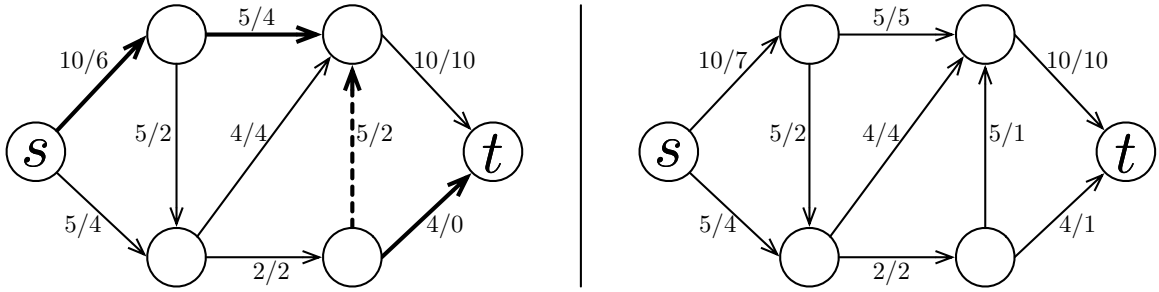


Figure 2.2: Left: a flow f and an augmenting path depicted in bold. The path traversed the dashed edge in the opposite direction, which corresponds to an edge in the residual graph because there is flow across this edge. Right: the resulting flow f' obtained after augmenting along the path as described above. Note, the capacity of the cut U consisting of the three leftmost nodes is equal to the value of f', so f' must be a maximum flow.

2.3 Residual Graphs and Augmenting Paths

Let $E_R = \{vu : uv \in E\}$ be the edges we would get by reversing edges in E. This should be taken to be disjoint from E. For $e \in E$ let $\overleftarrow{e}$ be the reverse of e in E_R and for $e \in E_R$ let $\overleftarrow{e}$ denote the original edge in E whose reverse is e. So if $e \in E_R$ and we let $f = \overleftarrow{e}$ then $e = \overleftarrow{f}$.

Definition 4 *Let f be a flow. Let $\mathcal{G}_f = (V, E_f)$ be the* **residual graph** *(with respect to f) where $E_f = \{e \in E : f(e) < \mu(e)\} \dot\cup \{e \in E_R : f(\overleftarrow{e}) > 0\}$. Let $\mu_f : E_f \to \mathbb{F}_{\geq 0}$ be a* **residual capacity function** *defined as follows for $e \in E_f$,*

$$\mu_f(e) = \begin{cases} \mu(e) - f(e) & \text{if } e \in E \\ f(\overleftarrow{e}) & \text{if } e \in E_R. \end{cases}$$

Note that $\mu_f(e) > 0$ for every $e \in E_f$.

Call an $s - t$ path P in $\mathcal{G}_f$ an **augmenting path**. We will use such a path to "increase the flow" from s to t as follows.

Set

$$\alpha = \min_{e \in P} \mu_f(e) > 0.$$

Let f' be defined as follows for $e \in E$,

$$f'(e) = \begin{cases} f(e) + \alpha & \text{if } e \in E \\ f(e) - \alpha & \text{if } \overleftarrow{e} \in P \\ f(e) & \text{otherwise.} \end{cases}$$

The process is depicted in Figure 2.2.

The following lemma explains why we do this:

Lemma 2 *With f, f', and α as described above, f' is a flow and* $\mathrm{val}(f') = \mathrm{val}(f) + \alpha$. *Furthermore, if f is integral and all capacities are integral, then f' is also integral.*

Proof. By our choice of α, we have $0 \leq f'(e) \leq \mu(e)$ for every edge e: If $e \in P$, then $f(e) < \mu(e)$ by the definition of μ_f, and by how α was chosen,

$$f(e) + \alpha \leq f(e) + \mu_f(e) = f(e) + \mu(e) - f(e) = \mu(e).$$

Similarly, if the residual edge of e is in P, then the flow along $\overleftarrow{e}$ is at least α so

$$f(e) - \alpha \geq f(e) - \mu_f(\overleftarrow{e}) = f(e) - f(e) = 0.$$

Further, f' is flow-conserving: for each vertex $v \neq s, t$ which P visits let $e = uv, e' = vw \in E_f$ denote the edges of P incident to v.

- If $e, e' \in E$ or $e, e' \in E_R$ then $f(\delta^{in}(v))$ and $f(\delta^{out}(v))$ either both increase or both decrease by α.
- If $e \in E$ and $e' \in E_R$ then $f'(e) = f(e) + \alpha$ and $f'(e') = f(e') - \alpha$. As both e, e' enter v, the net flow through v does not change.
- Similarly, if $e \in E_R$ and $e' \in E$ then $f'(e) = f(e) - \alpha$ and $f'(e') = f(e') + \alpha$. As both e, e' exit v, the net flow through v does not change.

Applying the same reasoning to s, let $e = sv$ be the edge that exits s on P. If $e \in E$ then $f'(e) = f(e) + \alpha$ where $e \in \delta^{out}(s)$. If $e \in E_R$ then $f'(\overleftarrow{e}) = f(\overleftarrow{e}) - \alpha$ where $e \in \delta^{in}(s)$. In either case, $f'(\delta^{out}(s)) - f'(\delta^{in}(s)) = f(\delta^{out}(s)) - f(\delta^{in}(s)) + \alpha$. so, by definition, $\text{val}(f') = \text{val}(f) + \alpha$.

Summarizing, f' is a nonnegative flow that obeys the capacities and sends α more flow than f. The observation about integrality of f' follows because the residual capacities μ_f are integers (given the assumptions), so α is then integral. ■

Since α is positive by definition, then finding an $s - t$ path in a flow's residual graph allows us to efficiently construct a flow with strictly higher value. Algorithm 1 uses this process to find a maximum value flow.

Algorithm 1 The Ford-Fulkerson MAXIMUM FLOW Algorithm

Input: A flow network $\mathcal{G} = (V, E)$ with capacity function μ
Output: A maximum-value flow f

```
f(e) ← 0 ∀e ∈ E
while there exists an s − t path P in G_f do
    augment f along P (as described above) to get a flow f' with larger value
    f ← f'
end while
return f
```

The following, in conjunction with Lemma 1, proves Theorem 1.

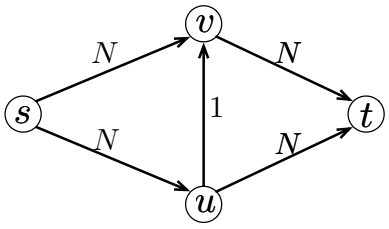
Theorem 3 *Let f be the flow returned by Algorithm 1 and let $U \subseteq V$ be the nodes reachable from s in $\mathcal{G}_f$. Then $\text{val}(f) = \mu(\delta^{out}(U))$. Furthermore, if all capacities are integers then $f(e) \in \mathbb{Z}_{\geq 0}$.*

Proof.

$\text{val}(f)$	$=$	$f(\delta^{out}(U)) - f(\delta^{in}(U))$	Because U is an $s - t$ cut.
	$=$	$\mu(\delta^{out}(U)) - \mu(\delta^{in}(U))$	No $e \in \delta^{out}(U)$ has $f(e) < \mu(e)$; if some e did, we would have $e \in E_f$.
	$=$	$\mu(\delta^{out}(U))$	$f(e) = 0$ for $e \in \delta^{in}(U)$, otherwise we would have $\overleftarrow{e} \in E_f$.

The statement about f being integral is simply by induction on the number of iterations, given the last statement of Lemma 2. ■

Important note: This is not a polynomial-time algorithm! Depending on the paths found, the number of iterations may be exponential in the size of the input. Consider Figure 2.3 where N is a positive integer.



For $1 \leq i \leq 2N$ let the i'th iteration of Algorithm 1 choose augmenting path $\langle s, u, v, t \rangle$ if i is odd or $\langle s, v, u, t \rangle$ if i is even.

The middle edge traversed by the augmenting path always has residual capacity 1 meaning each iteration increases the flow by 1. Overall algorithm runs for $2N$ iterations. But the number of bits used to write N is $\sim \log_2 N$ so the algorithm takes exponential time.

In fact, the exercises in the Korte-Vygen textbook give an example where if the capacities are irrational then the Ford-Fulkerson algorithm may never terminate.