# 6.1220 LECTURE 10

BRENDON JIANG

## Contents



# 1. Max Flow II

## 1.1. Review

**Definition 1.1: Flow Network**

A **flow network** is a directed graph $G = (V, E)$ with two distinguished vertices: a source vertex $s$ and a sink $t$.
Each directed edge $(u, v) \in E$ has a positive capacity $c(u, v)$. If $(u, v) \notin E$, then we define $c(u, v) = 0$.

**Definition 1.2: Value of Net Flow**

The value of a net flow, denoted by $|f|$, is given by $|f| = \sum_{v \in V} f(s, v)$.

Our goal is to maximize the total flow from $s$ to $t$ while respecting the capacity and flow conservation constraints. Once we find the flow via an algorithm, we can prove it is maximal by using cuts on the flow network.

*Date*: September 2024

**Definition 1.3: Cut on flow network**

A cut $(S, T)$ of a flow network $G = (V, E)$ is a partition of $V$ such that $s \in S$ and $t \in T$. If $f$ is a net flow on $G$, then we call $f(S, T)$ the net flow across the cut.

**Theorem 1.4: Max-flow Min-cut**

Suppose $f$ is net flow of maximum possible value. Then:

- there is at least one cut $(S, T)$ that has $|f| = c(S, T)$.

## 1.2. Augmenting paths

We have covered residual networks in the last lecture, which tells us what edges are available for altering in the flow.

**Definition 1.5**

Let $f$ be a net flow on $G = (V, E, c)$. The residual network $G_f(V, E_f, c_f)$ is a weighted directed graph defined as flows:

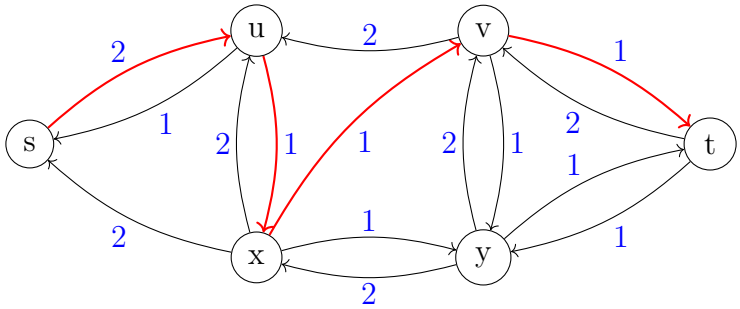
- set $c_f(u, v) = c(u, v) - f(u, v)$
- $E_f$ contains all pairs $(u, v)$ such that $c_f(u, v) > 0$.

An augmenting path is a path in the residual network which we can use to (potentially) better our current flow:, Each edge in the path carries two semantics (pushing or cancelling net flow on edge). We can also consider the augmenting path to be a separate flow, and we add the current flow with the augmenting flow together to get a better flow, while respecting constraints.

**Definition 1.6: Augmenting Path**

Any path from $s$ to $t$ in $G_f$ is an augmenting path in $G$ with respect to $f$. The flow value can be increased along an augmenting path by

$$c_f(p) = \min_{(u,v)\in p} \{c_f(u, v)\}$$




An Augmented Path in a Residue Network

## 1.3. Ford Fulkerson algorithm

Now, we actually prove the Max-flow, min-cut theorem.

**Theorem 1.7: Max-flow Min-cut (complete)**

Suppose $f$ is a net flow. The following are equivalent:

- $|f| = c(S,T)$ for some cut $(S,T)$
- $f$ is a maximum flow
- $f$ admits no augmenting paths

**Proof**

We prove that (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (1).

- (1) $\Rightarrow$ (2). Suppose $|f| = c(S,T)$ for some cut $(S,T)$. Since any alternative flow $f'$ must satisfy $|f'| \leq c(S,T)$, we have $|f'| \leq |f|$. Thus $f$ is a maximum flow.
- (2) $\Rightarrow$ (3). Proof by contrapositive. If there were an augmenting path, the flow value could be increased, contradicting the maximality of $f$.
- (3) $\Rightarrow$ (1). We suppose $f$ admits no augmenting paths. Intuitively, we want to pick a cut that takes advantage of this property. Define $S = \{v \in V : v \text{ is reachable from } s \text{ in } G_f\}$, and set $T = V - S$. Note that there is no path from $s$ to $t$ (no augmenting path), so $s \in S$ and $t \in T$ implying $(S,T)$ is a cut.
  Now, consider any pair of vertices $u \in S$ and $v \in T$. We must have $c_f(u,v) = 0$ since if $c_f(u,v) > 0$, then $v \in S$, not $v \in T$ as assumed. Thus, $f(u,v) = c(u,v)$, and summing this over all vertices in $S$ and $T$ gives $|f| = f(S,T)$.

Now, the most natural algorithm we have is to keep augmenting a flow until it is a max flow. This is exactly Ford-Fulkerson:

**Algorithm 1:** Ford-Fulkerson Algorithm

```
f[u, v] ← 0 for all u, v ∈ V
while an augmented path p exists in G_f do
    augment f by c_f(p).
```

Images of Ford-Fulkerson being ran in action could be found in the lecture slides.

## 1.4. Edmonds Karp algorithm

However, running Ford-Fulkerson naively can induce problems. When the edge weights are large, the augmenting path can be limited by some small weight, and be inefficient when updating our flow. If one of the edge weights is large like 1 billion, we may need steps in the order of 1 billion to solve the max flow problem. Edmonds-Karp avoids this by observing we can take the unweighted version

of $G_f$ and use the shortest path in that. They proved that this algorithm is $O(VE)$ augmentations in the worst case, so $O(VE^2)$ iterations total.

## 1.5. Flow integrality

**Claim 1.8: FLow Integrality**

Suppose the flow network has integer capacities. Then, there is a maximum flow that is integer-valued.

**Proof**

Every iteration of augmenting path retains the integral property for all values in the flow network including capacities and net flow. Thus, the maximum flow will be integer-valued, always. The more rigorous version of this proof can be completed with induction.

## 1.6. Applications

An application of flows is Maximum Bipartite Matching.

**Definition 1.9: Maximum Bipartite Matching**

For a graph $G = (V, E)$, a subset of the edges $M$ forms a matching if and only if no two edges in $M$ are incident to the same vertex.

We can use flow by adding a source and sink node to opposite sides of the bipartite graph and connect it with all vertices in their respective sides. The max flow then gives optimal matching, since the flow is a constant multiple of the matching (specifically 3). Using the flow integrality condition in the last subsection, we can obtain a max flow and maximal matching.