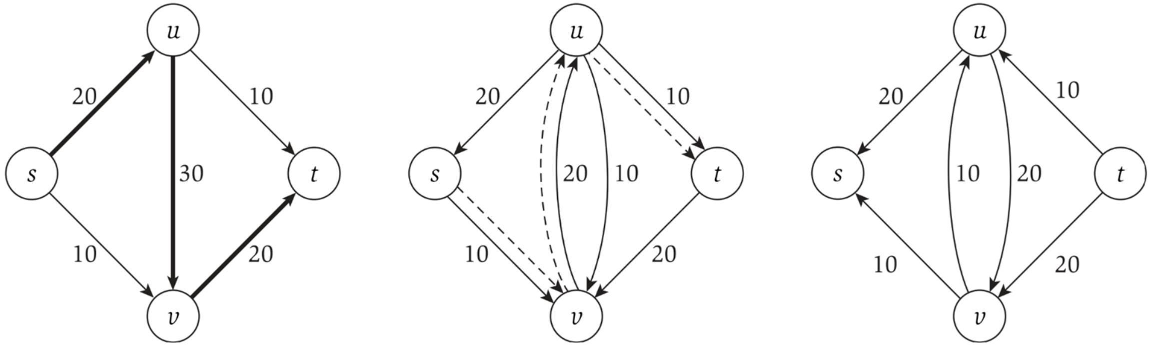
Algorithm Design & Analysis (CSE222)

Lecture-21

Recap

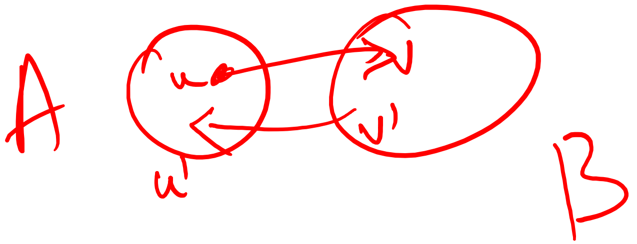
- Network Flow
  - Maximum flow (Ford-Fulkerson).
  - Augmenting path on G_f (increases flow).
  - Augmenting path insures capacity constraints & algorithm ensures conservation constraints.
  - Tractable running time for integer and rational valued capacities.



Outline

- Network Flows

Network Flow



Does Ford-Fulkerson returns the maximum possible flow in G?

- Relation between flow f and total capacity C at source: $|f| \leq C$.
- A <u>cut</u> is a partition of vertices into <u>disjoint</u> set A and B, where $s \in A$ and $t \in B$. The capacity of a cut is $c(A,B) = \sum_{e:A \to B} c(e)$.

Claim: Given a flow network, let f be a s-t flow and (A,B) be a s-t cut. Then the total flow $|f| = f^{out}(A) - f^{in}(A)$.

Network Flow

<u>Claim</u>: Given a flow network, let f be a s-t flow and (A,B) be a s-t cut. Then the total flow $|f| = f^{out}(A) - f^{in}(A)$.

$$\overline{f^{out}(s) - f^{in}(s)} + \sum_{v \in A \setminus s} f^{out}(v) - f^{in}(v)$$

<u>Proof</u>:

- Def: $|f| = f^{out}(s)$. It implies $|f| = f^{out}(s) - f^{in}(s)$.
- Further, $|f| = \sum_{v \in A} f^{out}(v) - f^{in}(v) = \sum_{e \text{ from } A} f(e) - \sum_{e \text{ to } A} f(e) = f^{out}(A) - f^{in}(A)$.

Similar claim: Given a flow network, let f be a s-t flow and (A,B) be a s-t cut. Then the total flow $|f| = f^{in}(B) - f^{out}(B)$.

Network Flow

Claim: Given a flow network, let f be a s-t flow and (A,B) be a s-t cut. Then the total flow $|f| \leq c(A,B)$.

Proof: We know, $|f| = f^{out}(A) - f^{in}(A) \leq f^{out}(A) = \sum_{e \text{ from } A} f(e) \leq \sum_{e \text{ from } A} c(e) = c(A,B)$.

In terms of bounding |f| the above claim is weaker than previous claim. Why?

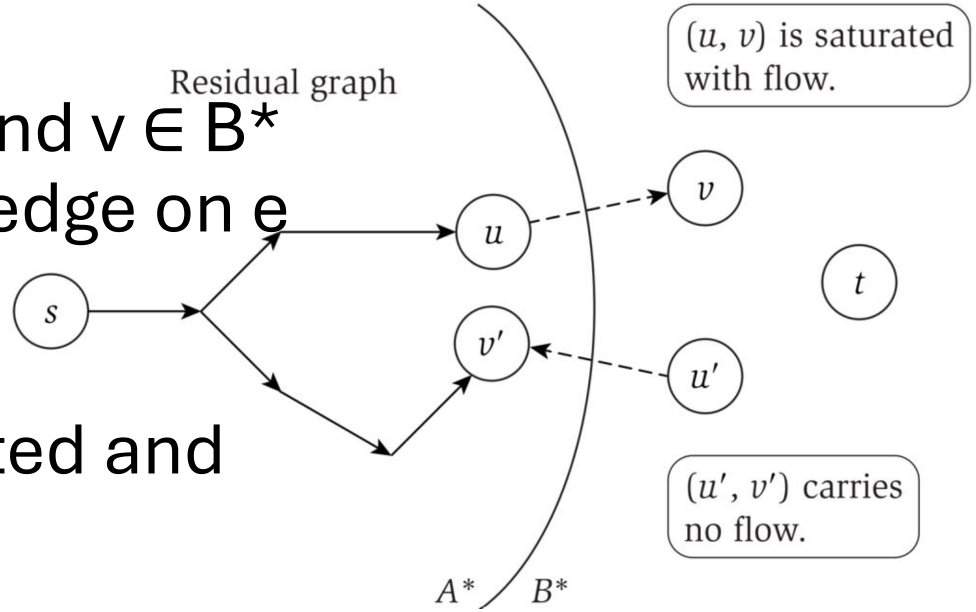
However the above claim is stronger in general. Why?

Max-flow & Min-cut

<u>Claim</u>: For a flow f in a flow graph there is a cut (A*,B*) such that |f| = c(A*,B*).

<u>Proof</u>: Let A* be the set of all nodes such that there is path s-v in G_f & B* = V \ A*.
(A*,B*) is a cut.

- Let, e = (u,v) be an edge such that u ∈ A* and v ∈ B*
- Then f(e) = c(e). Since there is no forward edge on e
- Let e' = (u',v') edge in G, u'∈B* & v'∈A*.
- Then f(e') = 0. As no backward edge in G_f.
- So all edge from A* are completely saturated and all edge into A* are completely unused.



So, $|f| = f^{out}(A^*) - f^{in}(A^*) = \sum_{e \text{ from } A^*} f(e) - \sum_{e \text{ into } A^*} f(e) = \sum_{e \text{ from } A^*} c(e) - 0 = c(A^*,B^*)$.

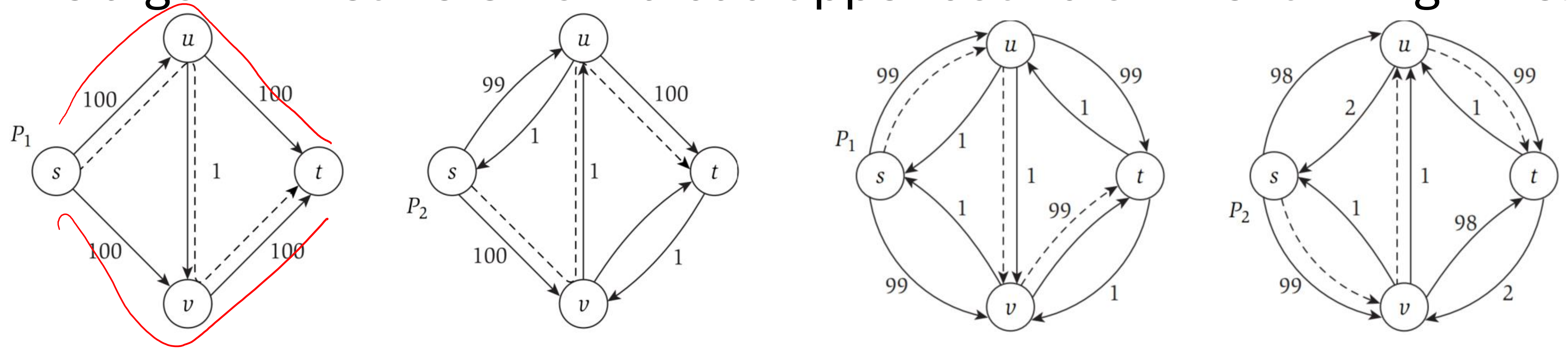
Network Flow

Claim: Given a max flow f, one can compute s-t min cut in O(m) time.

Proof: Based on the previous claim.

Ford-Fulkerson

The algorithm suffers from a bad upper bound on the running time.



How to improve the running time: Choose a random path s-t in G_f.

- Guarantees, that over expectation the augmenting path with worst bottleneck not selected. Bounding running time not easy.
- Augmenting path with highest bottleneck capacity in G_f. How to find?
- Large enough bottleneck capacity. How to decide?

Scaling Max Flow

CAPACITY-SCALING(G)

FOREACH edge $e \in E$: $f(e) \leftarrow 0$.

$\Delta \leftarrow$ largest power of 2 $\leq C$.

WHILE ($\Delta \geq 1$)

$G_f(\Delta) \leftarrow \Delta$-residual network of G with respect to flow f.

WHILE (there exists an $s \leadsto t$ path P in $G_f(\Delta)$)

$f \leftarrow$ AUGMENT(f, c, P).

Update $G_f(\Delta)$.

Δ-scaling phase

$\Delta \leftarrow \Delta / 2$.

RETURN f.

The residual graph $G_f(\Delta)$ contains only those edges whose capacity $\geq \Delta$.

Every augmenting path in this residual graph has a bottleneck at least Δ.

Reference

Slides

Jeff Erickson Chp-10

Algorithms Design by Kleinberg & Tardos - Chp 7.2 & 7.3