Analysis of Algorithms, I
CSOR W4231.002

Eleni Drinea
Computer Science Department

Columbia University

Thursday, April 9, 2015

Outline

1. Recap
2. Correctness of the Ford-Fulkerson algorithm
3. Application: max bipartite matching

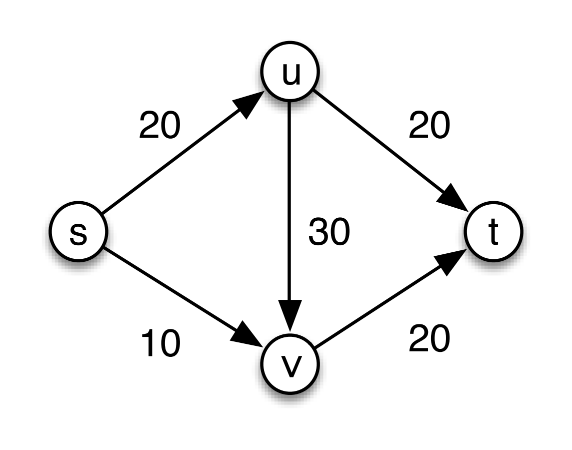
Today

1. Recap
2. Correctness of the Ford-Fulkerson algorithm
3. Application: max bipartite matching

Review of the last lecture

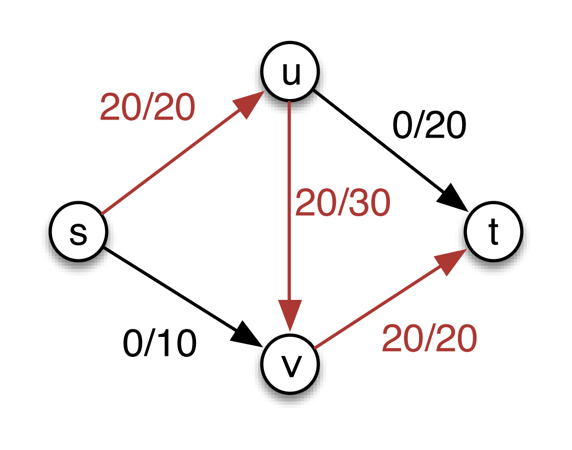
1. Flow networks
   - Applications
2. The residual graph and augmenting paths
3. The Ford-Fulkerson algorithm for max flow
   - Running time analysis

Example flow network $G = (V, E, c, s, t)$



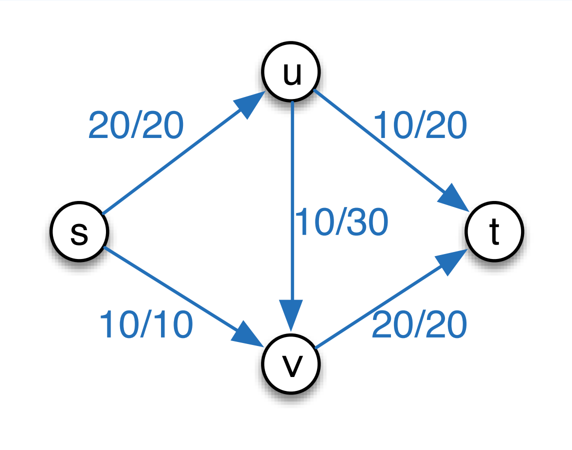
An example flow network.

A flow of value 20



A flow f of value 20.

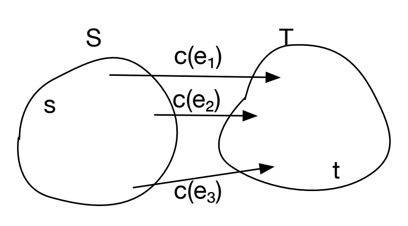
The max flow of value 30



The maximum flow of value 30.

A natural upper bound for the max value of a flow

- An s-t cut (S, T) in G is a partition of the vertices into two sets S and T, such that $s \in S$ and $t \in T$.



- The capacity $c(S, T)$ of s-t cut (S, T) is $\sum_{\text{e out of } S} c(e)$.
- Then

$$\max_f |f| \leq \min_{(S,T) \text{ cut in } G} c(S, T) \tag{1}$$

The residual graph $G_f = (V, E_f, c_f)$

Definition 1.

Given flow network G and flow f, the residual graph G_f has

- the same vertices as G;
- for every edge $e = (u, v) \in E$ such that $f(e) < c(e)$, an edge $e = (u, v)$ with capacity $c_f(e) = c(e) - f(e)$ (forward edge);
- for every edge $e = (u, v) \in E$ such that $f(e) > 0$, an edge $e^r = (v, u)$ with capacity $c_f(e^r) = f(e)$ (backward edge).

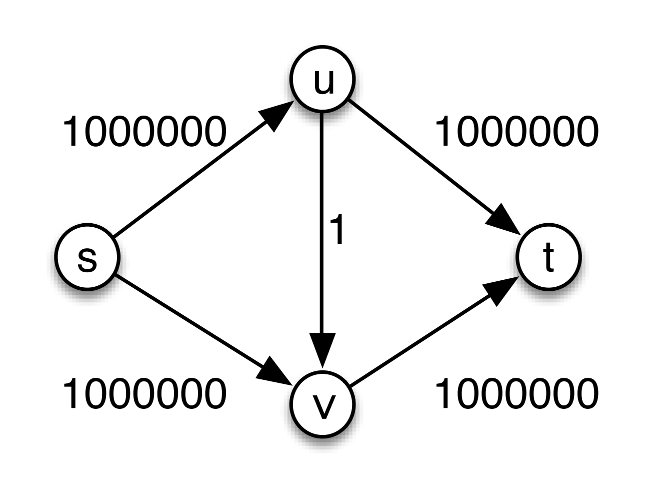
So G_f has $\leq 2m$ edges.

The Ford-Fulkerson algorithm for max flow

```
Ford-Fulkerson(G = (V, E, c, s, t))
   for all e ∈ E do f(e) = 0
   end for
   while there is an s-t path in G_f do
      let P be a simple s-t path in G_f
      f' = Augment(f, P)
      set f = f'
      set G_f = G_{f'}
   end while
   return f'
```

Running time: $O(mnC)$, where $C = \max_{e \in E} c(e)$

Problems with pseudo-polynomial running times



Improved algorithms

- Can be made polynomial: use BFS instead of DFS
  - Edmonds-Karp: $O(nm^2)$, Dinitz: $O(n^2m)$, improvements: $O(nm\log n)$, $O(n^3)$
- **Unit** capacities: $O(\min\{m^{3/2}, mn^{2/3}\})$ [EvenTarjan1975]
  - Improved for sparse graphs: $\tilde{O}(m^{10/7})$ [Madry2013]
- **Integral** capacities: $O(\min\{m^{3/2}, mn^{2/3}) \log(n^2/m) \log C)$ [GoldbergRao1998]
  - Improved: $\tilde{O}(m\sqrt{n}\log^2 C)$ [LeeSidfort2014]; *also yields improvement for unit capacities, dense graphs*
- **Real** capacities: $O(nm\log(n^2/m))$
  - Improved: $O(nm)$ [Orlin2013]

Today

1 Recap

2 Correctness of the Ford-Fulkerson algorithm

3 Application: max bipartite matching

Roadmap for proving optimality of Ford-Fulkerson

Let f be the flow upon termination of the Ford-Fulkerson algorithm.

1. Exhibit a specific s-t cut (S^*, T^*) in G such that the

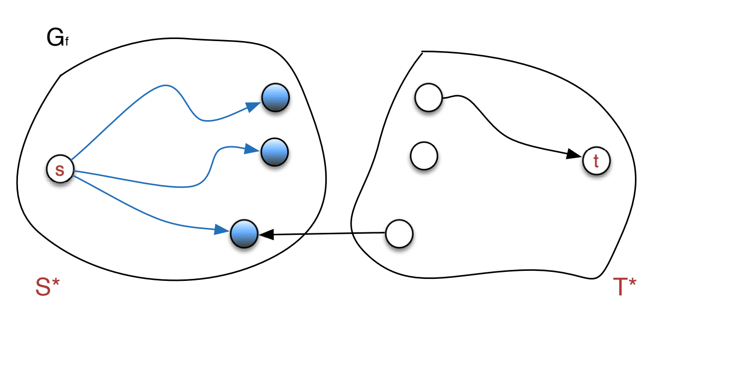
$$\text{value of } f = \text{capacity of the cut } (S^*, T^*)$$

2. Show that f is maximum
   - This formalizes our intuition that the max flow cannot exceed the capacity of **any** cut

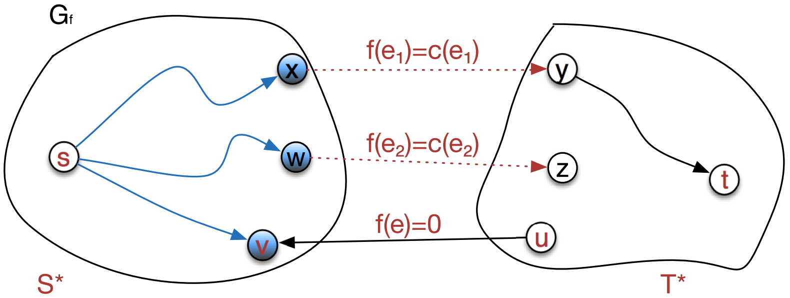
Ford-Fulkerson terminates when no s-t path in G_f

Consider the residual graph G_f upon termination of the algorithm. Let (S^*, T^*) be the cut in G_f where

- S^* is the set of nodes reachable from the source s;
- T^* contains every other node.



The flow on edges crossing between S^* and T^* in G



Every edge e in G crossing from S* to T* satisfies f(e)=c(e) (of course, such e does not appear in G_f).
Every edge e′ in G crossing from T* to S* satisfies f(e′)=0.

On the cut (S^*, T^*)

1. (S^*, T^*) is an s-t cut: that is, $s \in S^*$, $t \in T^*$. *(why?)*
2. In G_f, no edge crosses from S^* to T^*. *(why?)*
3. Hence, if $e = (x, y) \in E$ with $x \in S^*$ and $y \in T^*$, then $f(e) = c(e)$ (thus $e \notin E_f$).
4. Similarly, if $e' = (u, v) \in E$ with $u \in T^*$ and $v \in S^*$, then $f(e') = 0$. *(why?)*

Net flow across a cut

Definition 2.

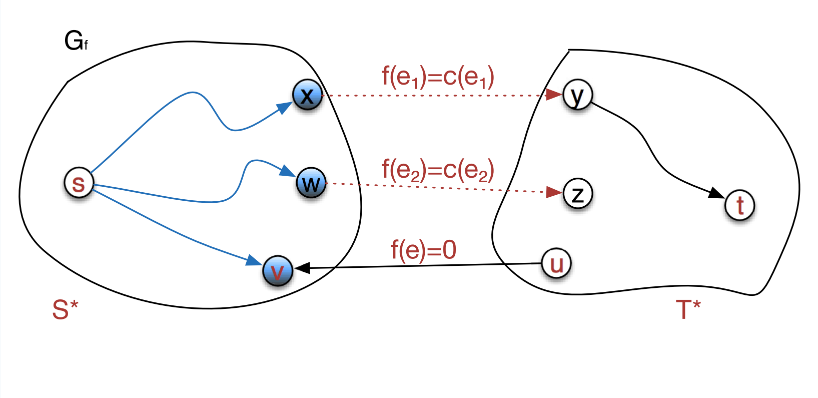
The net flow across an s-t cut (S, T) is the amount of flow leaving the cut minus the amount of flow entering the cut

$$f^{\text{out}}(S) - f^{\text{in}}(S), \tag{2}$$

where

1. $f^{\text{out}}(S) = \sum_{e \text{ out of } S} f(e)$
2. $f^{\text{in}}(S) = \sum_{e \text{ into } S} f(e)$

Net flow across (S^*, T^*) equals capacity of (S^*, T^*)



$$
\begin{aligned}
f^{\text{out}}(S^*) - f^{\text{in}}(S^*) &= \sum_{e \text{ out of } S^*} f(e) - \sum_{e \text{ into } S^*} f(e) \\
&= \sum_{e \text{ out of } S^*} c(e) - 0 \\
&= c(S^*, T^*) \qquad (3)
\end{aligned}
$$

Roadmap revisited

Let f be the flow upon termination of the Ford-Fulkerson algorithm.

1. Exhibit a specific s-t cut (S^*, T^*) in G such that the

$$|f| = c(S^*, T^*)$$

   Not quite there yet!
   - *We exhibited (S^*, T^*) with net flow equal to its capacity.*
   - *We need to relate the net flow across (S^*, T^*) to the value $|f|$ of the flow (that is, the flow out of s).*
   - *In particular,* **if we showed them equal**, *then we'd have $|f| = c(S^*, T^*)$.*

2. Show that f is maximum
   - This formalizes our intuition that the max flow cannot exceed the capacity of **any** cut

$|f|$ **equals** the net flow across any s-t cut (S, T)

Recall that

- $f^{\text{out}}(S) = \sum_{e \text{ out of } S} f(e)$
- $f^{\text{in}}(S) = \sum_{e \text{ into } S} f(e)$
- the net flow across (S, T) is $f^{\text{out}}(S) - f^{\text{in}}(S)$

Lemma 3.

Let f be any s-t flow, and (S, T) any s-t cut. Then

$$|f| = f^{out}(S) - f^{in}(S).$$

Proof of Lemma 3

Proof

First, observe that

$$|f| \;=\; f^{\text{out}}(s) = \sum_{v \in S} \Big(f^{\text{out}}(v) - f^{\text{in}}(v) \Big) \tag{4}$$

since

- $f^{\text{in}}(s) = 0$
- for every $v \in S - \{s\}$, the terms in the right-hand side of (4) cancel out because of flow conservation constraints

Goal: rewrite the right-hand side of equation 4 in terms of the edges that participate in these sums.

Proof of Lemma 3 (cont'd)

There are three types of edges:

1. Edges with both endpoints in S: such edges appear once in the first sum in equation 4 and once in the second, hence their flows cancel out.
2. Edges with the tail in S and head in T: such edges contribute to the first sum in equation 4 so they appear with a $+$.
3. Edges with the head in S and tail in T: such edges contribute to the second sum in equation 4 so they appear with a $-$.

In effect, the right-hand side of equation 4 becomes

$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ into } S} f(e).$$

The lemma follows.

The value of a flow cannot exceed capacity of any cut

Corollary 4.

Let f be any s-t flow and (S, T) any s-t cut. Then

$$|f| \leq c(S, T).$$

Proof.

$$|f| = f^{\text{out}}(S) - f^{\text{in}}(S) \leq f^{\text{out}}(S) \leq c(S, T).$$

□

Putting everything together

- By Corollary 4, the value of a flow cannot exceed the capacity of any cut; in particular,

$$|f| \leq c(S^*, T^*).$$

- By Lemma 3, $|f|$ is equal to the net flow across any (S, T) cut; in particular,

$$|f| = f^{\text{out}}(S^*) - f^{\text{in}}(S^*).$$

- The net flow across (S^*, T^*) equals $c(S^*, T^*)$ (equation 3). Hence the above becomes

$$|f| = f^{\text{out}}(S^*) - f^{\text{in}}(S^*) = c(S^*, T^*).$$

$\Rightarrow$ Thus the flow computed by Ford-Fulkerson is a maximum flow because it cannot be increased anymore.

The max-flow min-cut theorem

Theorem 5.

If f is an s-t flow such that there is no s-t path in G_f, then there is an s-t cut (S^*, T^*) in G such that $|f| = c(S^*, T^*)$. Therefore, f is a max flow and (S^*, T^*) is a cut of min capacity.

Theorem 6 (Max-flow Min-cut).

In every flow network, the maximum value of an s-t flow equals the minimum capacity of an s-t cut.

Integrality theorem

Recall the following fact from last lecture.

Fact 7.

During execution of the Ford-Fulkerson algorithm, the flow values $\{f(e)\}$ and the residual capacities in G_f are all integers.

Combine with Theorem 5 to conclude:

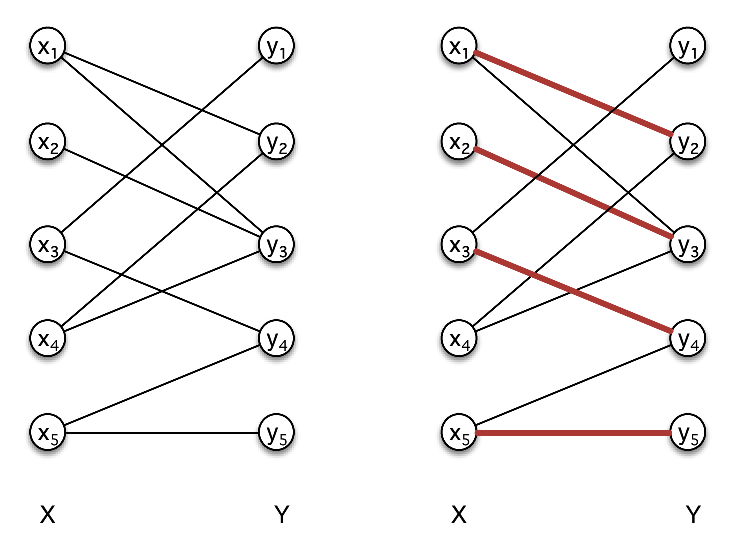
Theorem 8 (Integrality theorem).

If all capacities in a flow network are integers, then there is a maximum flow for which every flow value $f(e)$ is an integer.

Today

1. Recap
2. Correctness of the Ford-Fulkerson algorithm
3. Application: max bipartite matching

Bipartite Matching



Matchings

Definition 9.

A matching M is a subset of edges where every vertex in $X \cup Y$ appears at most once.

- Perfect matching: every vertex in $X \cup Y$ appears exactly once in M
  - not always possible, e.g., $|X| \neq |Y|$
- Maximum matching still desirable in applications
  - If we had an algorithm to find maximum matching then we could also find a perfect matching, if one exists (*why?*)

Finding maximum matchings in bipartite graphs

Idea: Use the Ford-Fulkerson algorithm to find maximum (or perfect) matchings in bipartite graphs.

Strategy: reformulate this problem as a max flow problem which we know how to solve.

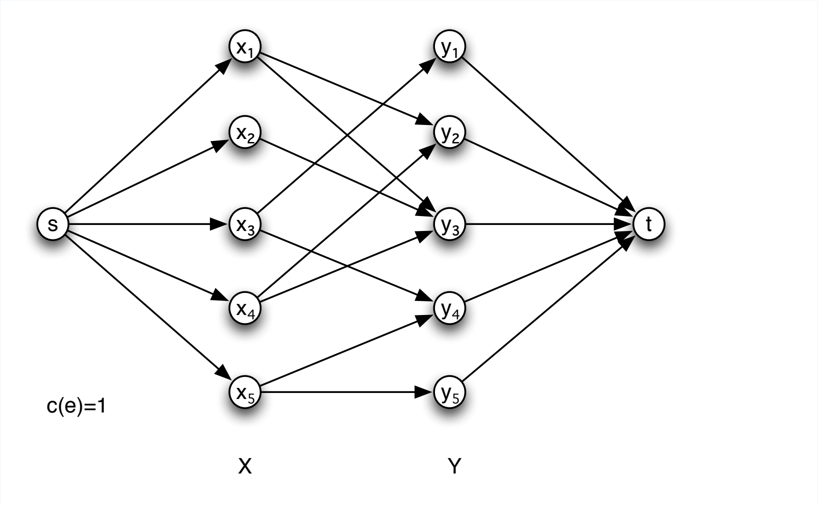
First, we need to transform the bipartite graph into a flow network.

Deriving a flow network given a bipartite graph

Given a bipartite graph $G = (X \cup Y, E)$, we construct a flow network G' as follows.

- Add a source s.
- Add a sink t.
- Add (s, x) edges for all $x \in X$.
- Add (y, t) edges for all $y \in Y$.
- Direct all $e \in E$ from X to Y.
- Assign to every edge capacity of 1.

The flow network for the bipartite graph of slide 29



Computing matchings in G from flows in G'

- $G = (X \cup Y, E)$ is the bipartite graph
- G' is the derived flow network

Claim 1.

The size of the maximum matching in G equals the value of the maximum flow in G'. The edges of the matching are the edges that carry flow from X to Y in G'.

Proof of Claim 1

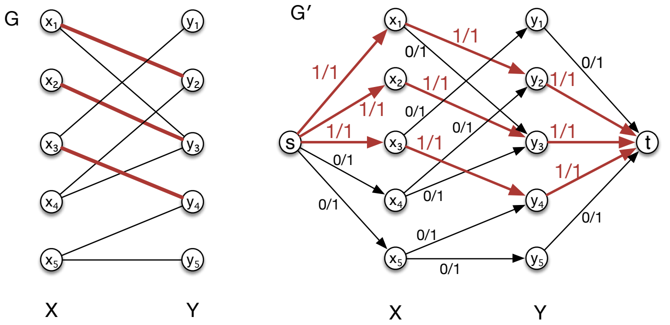
The proof follows if we show the following two statements.

1. ($\Rightarrow$ Forward direction) Given any matching M in G, we can construct a flow f in G' with value equal to the size of the matching, that is, $|M| = |f|$.

2. ($\Leftarrow$ Reverse direction) Given a max flow f in G', we can construct a matching M in G, with size equal to the value of the max flow.

(1. $\Rightarrow$) from a matching M to a flow f with $|f| = |M|$

Let $|M| = k$. Send one unit of flow along each of the k edge-disjoint s-t paths that use the edges in M; then $|f| = k$.



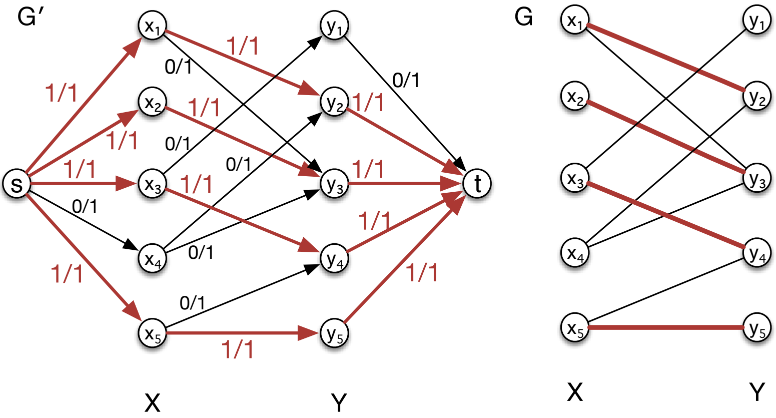
Given matching M (the red edges in G), construct the integral flow f in G′. Then the value of f equals the number of edges in M.

(2. $\Leftarrow$) from a flow f' to a matching M' with $|M'| = |f'|$

Given a max flow f' in G' with $|f'| = k$, we want to select a set of edges M' in G so that M' is a matching of size k.

- By the integrality theorem, there is an integer-valued flow f of value k.
- Then for every edge e, $f(e) = 0$ or $f(e) = 1$ (*why?*).
- Define M' fo consist of all edges of the form $e = (x, y)$ such that $f(e) = 1$.

Obtaining a matching M' from an integral flow f



Given integral flow f in G′, construct matching M′ (the red edges in G), so that the number of edges in M′ equals the value of f.

M' is a matching

We need to show that

1. **Fact 1**: M' is a matching.
2. **Fact 2**: M' has size k.

Proof of Fact 1.

Must show that every node in G' appears at most once in M'.

- Each node in X is the tail of at most one edge in M' (*flow conservation constraints*).
- Each node in Y is the head of at most one edge in M' (*flow conservation constraints*).

□

M' has size k

Proof of Fact 2.

- Consider the cut (S, T) where $S = \{s\} \cup X$, $T = Y \cup \{t\}$.
- We will compute its net flow.
  1. By Lemma 3 the net flow across (S, T) equals $|f|$.
     So the net flow across (S, T) equals k.
  2. By definition, the net flow of (S, T) is

$$f^{\text{out}}(S) - f^{\text{in}}(S) = |M'|$$

     since
     - the only edges that carry flow out of S are the edges in M'
     - the flow into S is 0 (no edges enter S)

$\Rightarrow$ Thus $|M'| = k$.

□

Time for finding max matching in bipartite graphs

1. Ford-Fulkerson: $O(mnC) = O(mn)$
2. Improved: $O(m\sqrt{n})$ [HopcroftKarp, Karzanov 1973]
3. Improved further for sparse ($m = O(n)$) graphs: $\tilde{O}(m^{10/7})$ [Madry2013]