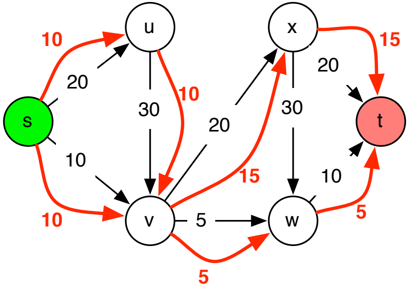
CMSC 451: Maximum Bipartite Matching

Slides By: Carl Kingsford

Department of Computer Science
University of Maryland, College Park

Based on Section 7.5 of *Algorithm Design* by Kleinberg & Tardos.

Network Flows

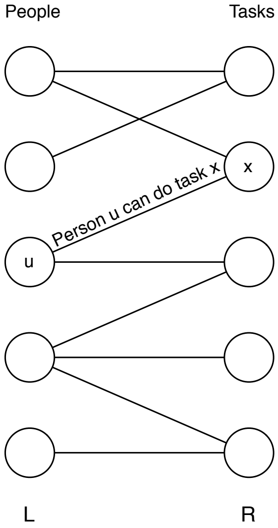


The network flow problem is itself interesting.

But even more interesting is how you can use it to solve many problems that don't involve flows or even networks.

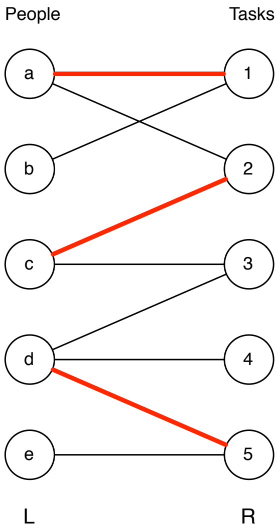
Bipartite Graphs

- Suppose we have a set of people L and set of jobs R.
- Each person can do only some of the jobs.
- Can model this as a bipartite graph $\rightarrow$



Bipartite Matching

- A **matching** gives an assignment of people to tasks.
- Want to get as many tasks done as possible.
- So, want a **maximum matching**: one that contains as many edges as possible.
- (This one is not maximum.)



Maximum Bipartite Matching

Maximum Bipartite Matching

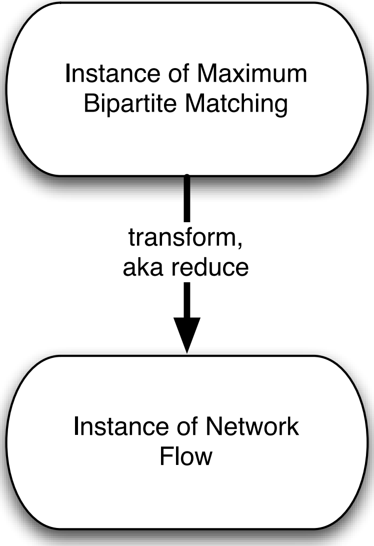
Given a bipartite graph $G = (A \cup B, E)$, find an $S \subseteq A \times B$ that is a matching and is as large as possible.

Notes:

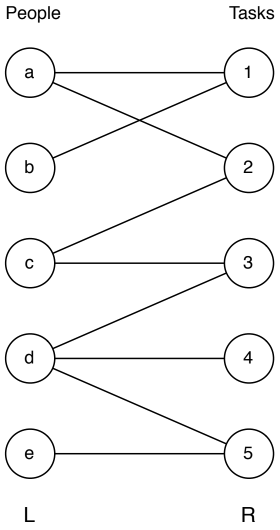
- We're given A and B so we don't have to find them.
- S is a **perfect matching** if every vertex is matched.
- *Maximum* is not the same as *maximal*: greedy will get to maximal.

Reduce

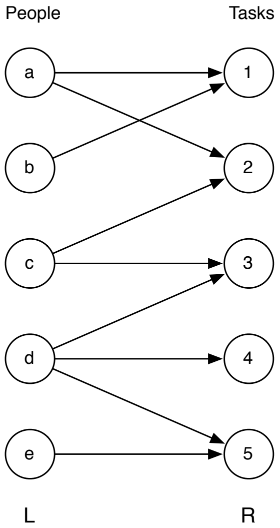
- Given an instance of bipartite matching,
- Create an instance of network flow.
- Where the solution to the network flow problem can easily be used to find the solution to the bipartite matching.



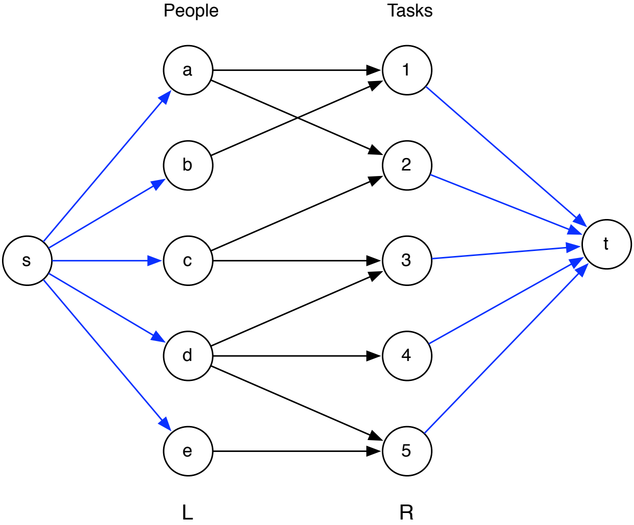
Reducing Bipartite Matching to Net Flow



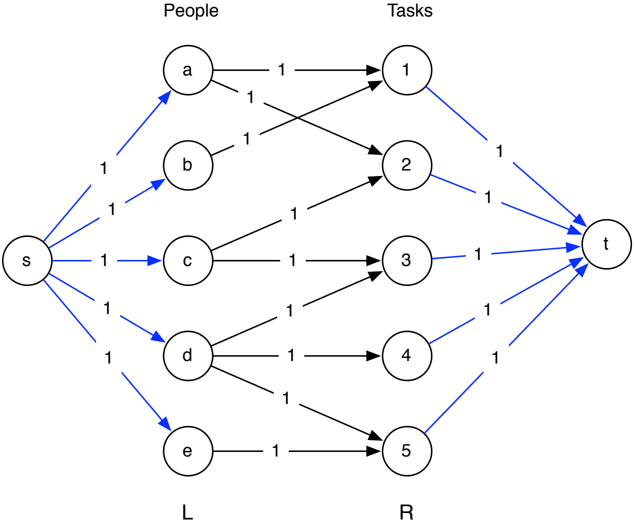
Reducing Bipartite Matching to Net Flow



Reducing Bipartite Matching to Net Flow



Reducing Bipartite Matching to Net Flow



Using Net Flow to Solve Bipartite Matching

To Recap:

1. Given bipartite graph $G = (A \cup B, E)$, direct the edges from A to B.
2. Add new vertices s and t.
3. Add an edge from s to every vertex in A.
4. Add an edge from every vertex in B to t.
5. Make all the capacities 1.
6. Solve maximum network flow problem on this new graph G'.

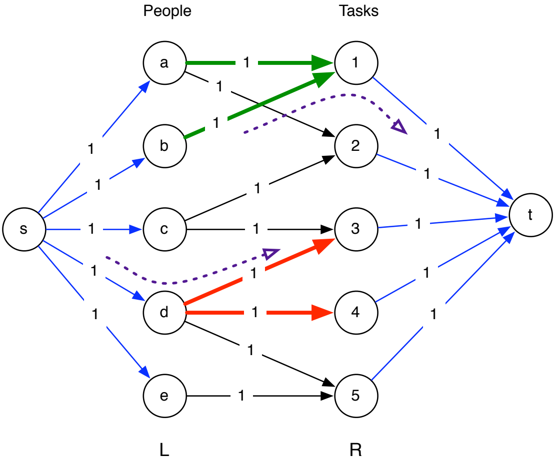
The edges used in the maximum network flow will correspond to the largest possible matching!

Analysis, Notes

- Because the capacities are integers, our flow will be integral.
- Because the capacities are all 1, we will either:
  - use an edge completely (sending 1 unit of flow) or
  - not use an edge at all.
- **Let M be the set of edges going from A to B that we use**.
- We will show that
  1. M is a matching
  2. M is the largest possible matching

M is a matching

We can choose at most one edge leaving any node in A.
We can choose at most one edge entering any node in B.

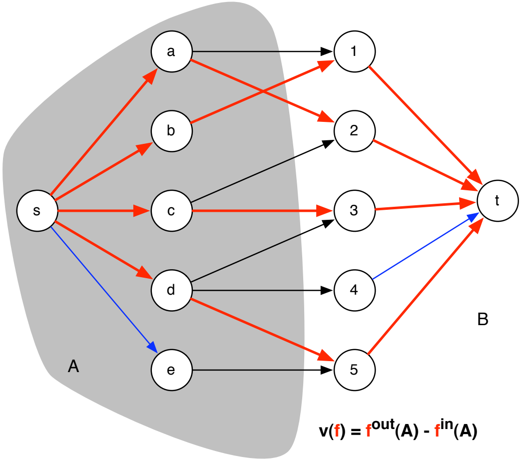


If we chose more than 1, we couldn't have balanced flow.

Correspondence between flows and matchings

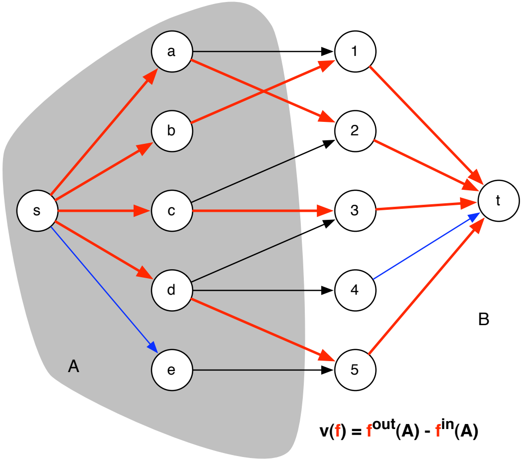
- If there is a matching of k edges, there is a flow f of value k.

- If there is a flow f of value k, there is a matching with k edges.



Correspondence between flows and matchings

- If there is a matching of k edges, there is a flow f of value k.
  - f has 1 unit of flow across each of the k edges.
  - ≤ 1 unit leaves & enters each node (except s, t)
- If there is a flow f of value k, there is a matching with k edges.



M is as large as possible

- We find the **maximum** flow f (say with k edges).
- This corresponds to a matching M of k edges.
- If there were a matching with $> k$ edges, we would have found a flow with value $> k$, contradicting that f was maximum.
- Hence, M is maximum.

Running Time

- How long does it take to solve the network flow problem on G'?
- The running time of Ford-Fulkerson is $O(m'C)$ where m' is the number of edges, and $C = \sum_{e \text{ leaving } s} c_e$.
- $C = |A| = n$.
- The number of edges in G' is equal to number of edges in G (m) plus $2n$.
- So, running time is $O((m + 2n)n) = (mn + n^2) = O(mn)$

Theorem

We can find maximum bipartite matching in $O(mn)$ time.

Summary: Bipartite Matching

- Fold-Fulkerson can find a maximum matching in a bipartite graph in $O(mn)$ time.
- We do this by **reducing** the problem of maximum bipartite matching to network flow.