CS 7800: Advanced Algorithms

Class 13: Network Flow Applications

- Reductions
- Bipartite Matching

Jonathan Ullman

October 17, 2025

Network Flow Summary

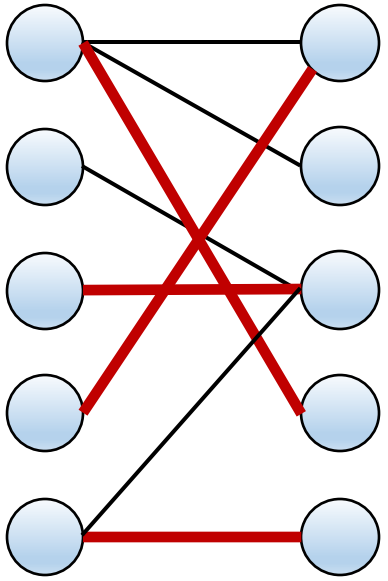
- **First Pass:** Can solve maximum flow in time $O(m \cdot v^*)$
  - Can be very slow when capacities are large
  - Cannot be improved if we allow arbitrary augmenting paths
  - Always finds an integer max flow when capacities are integers
- **Second Pass:** Improved running time via better paths
  - **Widest Augmenting Path:** $O(m \cdot \log v^*)$
  - **Shortest Augmenting Path:** $O(m^2 n)$
- **Still actively studied!**
  - Can solve maximum flow in $O(mn)$ using augmenting path* algos
  - **Recent Breakthrough:** Can solve maximum flow in time* $m^{1+o(1)}$
- **Today:** Using maximum-flow/minimum-cut as a building block for solving many more problems

Maximum Bipartite Matching

- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$
- **Output:** a matching of maximum size
  - A **matching** $M \subseteq E$ is a set of edges such that every node v is an endpoint of at most one edge in M
  - **Size** = $|M|$

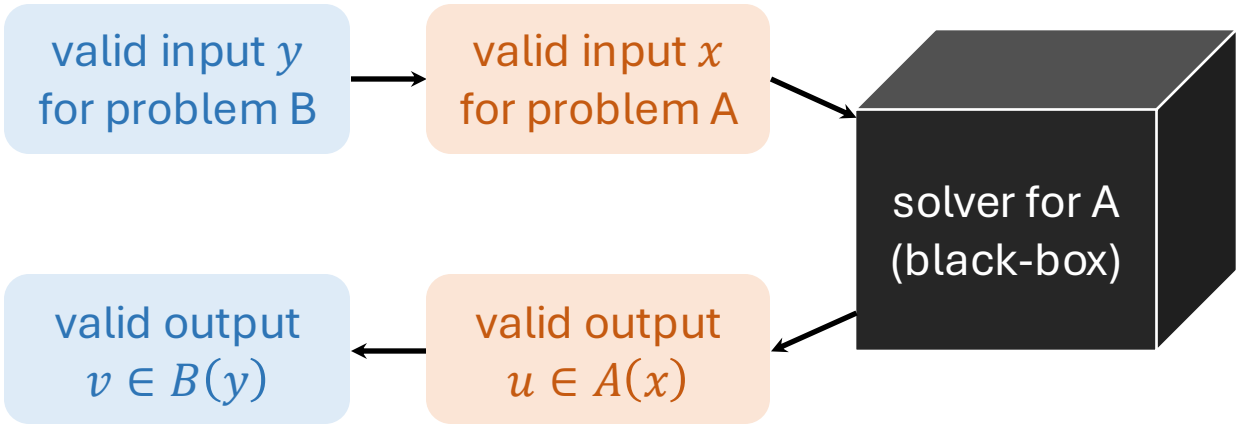
Models any problem where one type of object is assigned to another type:

- doctors to hospitals
- jobs to processors
- advertisements to websites



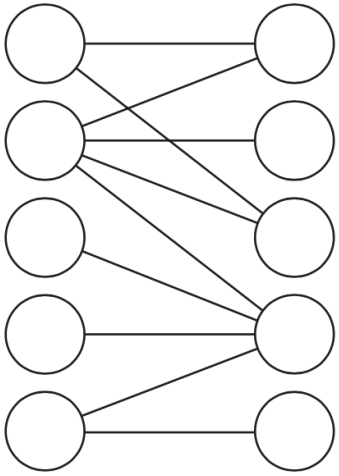
Mechanics of Reductions

- **Theorem:** There is an efficient algorithm that solves maximum bipartite matching (MBM) using an algorithm that solves integer max s-t flow (MF)



Step 1: Transform the Input

Step 1: Transform the Input

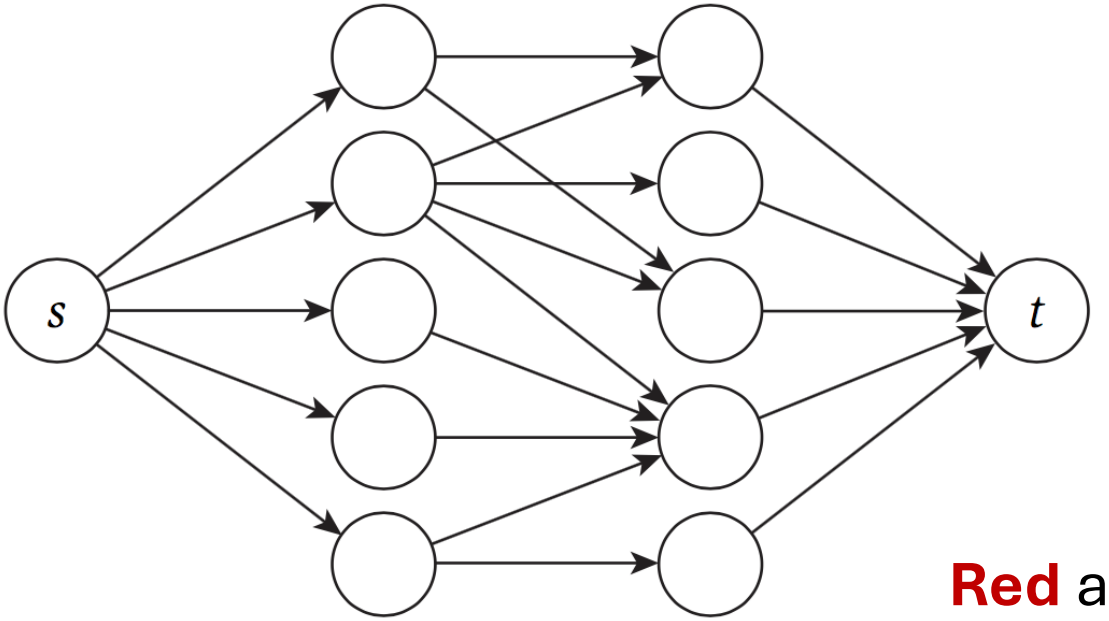


Step 2: Receive the Output

valid network G' for MF

solver for MF (black-box)

valid MF f' for network G'



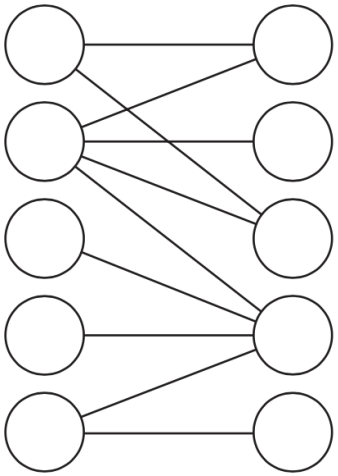
Red arrow means $f'(e) = 1$

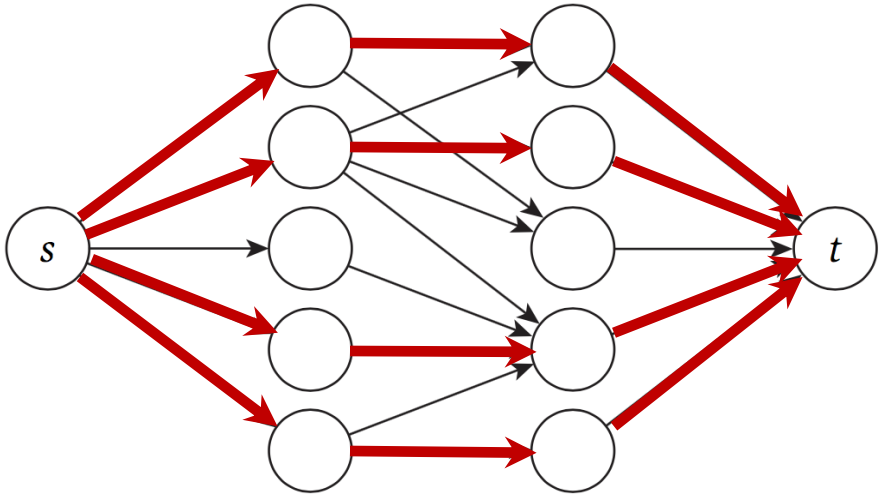
Black arrow means $f'(e) = 0$

Step 3: Transform the Output

valid MBM M for graph G

valid MF f' for network G'





Correctness

- Need to show:
  - Our algorithm returns a matching
  - Our algorithm returns a maximum matching

Correctness

- Our algorithm returns a matching

Correctness

- Out algorithm returns a maximum matching

Running Time

- Need to analyze:
  - Time to transform the input
  - Time to run the max-flow solver
  - Time to transform the otuput

Maximum Bipartite Matching Summary

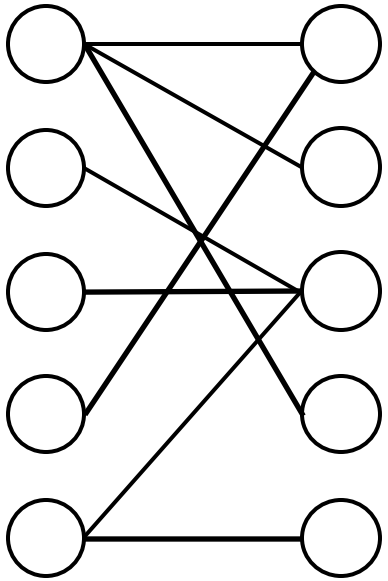
Solve maximum s-t flow in a graph with $n + 2$ nodes and $m + n$ edges and $c(e) = 1$ in time T

Solve maximum bipartite matching in a graph with n nodes and m edges in time $T + O(m)$

- Can solve max bipartite matching in time $O(nm)$ using Ford-Fulkerson
  - Improvement for maximum flow gives improvement for maximum bipartite matching!

Hall's Theorem

- How can we tell that a graph does not have a perfect matching?



Proof of Hall's Theorem via Duality

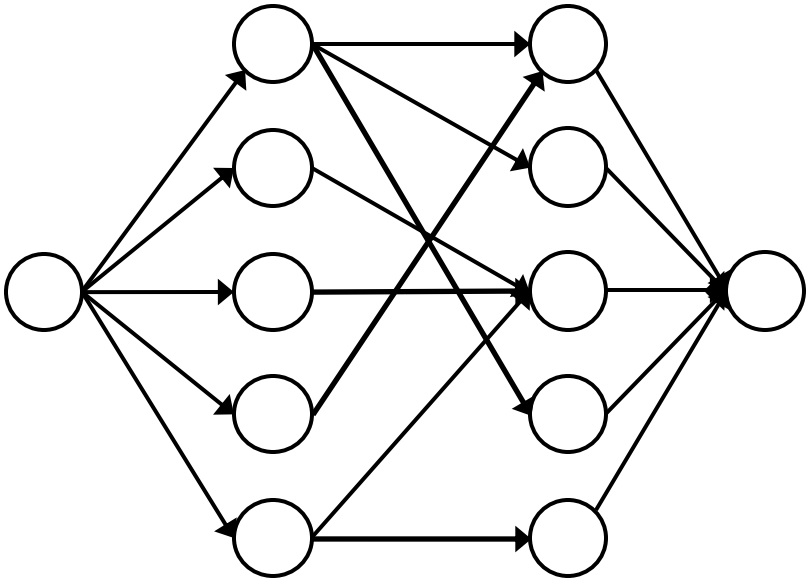


Image Segmentation

- Separate image into foreground and background
- We have some idea of:
  - whether pixel i is in the foreground or background
  - whether pair (i,j) are likely to go together

Image Segmentation

- Input:
  - a directed graph $G = (V, E)$
    - V = "pixels", E = "pairs"
  - likelihoods $a_i, b_i \geq 0$ for every $i \in V$
  - separation penalty $p_{ij} \geq 0$ for every $(i, j) \in E$
- Output:
  - a partition of V into (A, B) that maximizes

$$q(A, B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ \text{cut by } A,B}} p_{ij}$$

Reduction to MinCut

- Differences between SEG and MINCUT:
  - SEG asks us to maximize, MINCUT asks us to minimize

$$\max_{A,B} \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ \text{btw } A \text{ and } B}} p_{ij} \quad \Leftrightarrow \quad \min_{A,B} \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i,j) \in E \\ \text{btw } A \text{ and } B}} p_{ij}$$

  - SEG allows any partition, MINCUT requires $s \in A$, $t \in B$

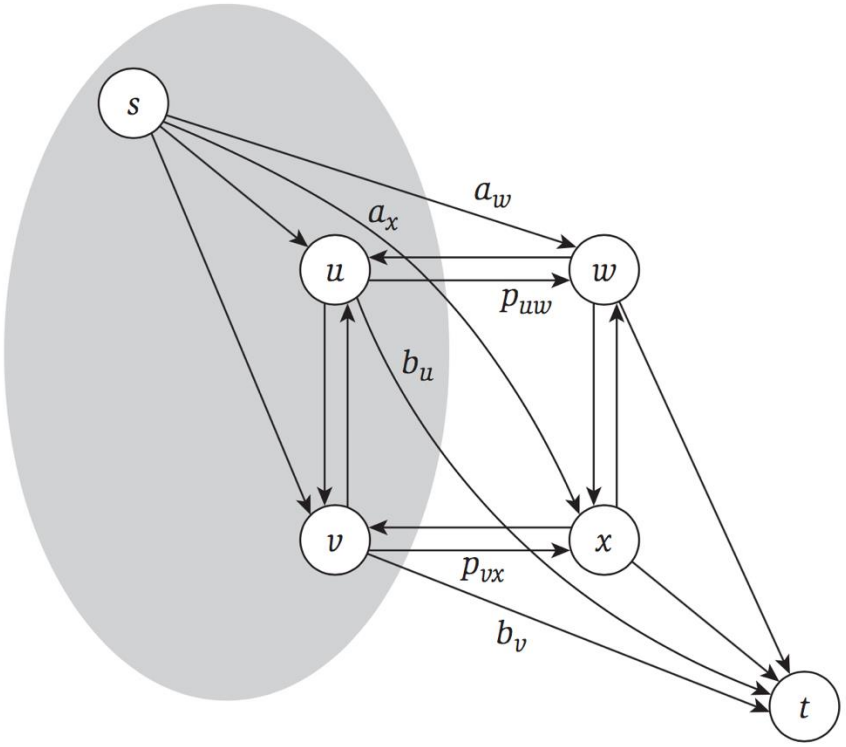
  - SEG counts any cut edge, MINCUT counts $A \to B$ edges

Reduction to MinCut

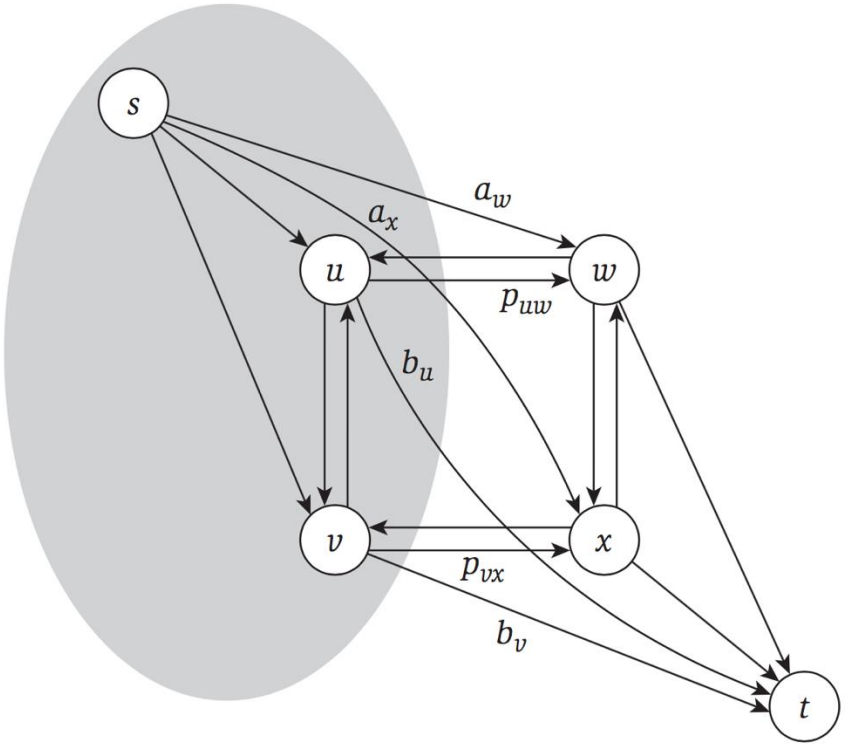
- How can we set up a flow network where the cost of the segmentation is the capacity of a cut

$$\min_{A,B} \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{\substack{(i,j) \in E \\ \text{btw } A \text{ and } B}} p_{ij}$$

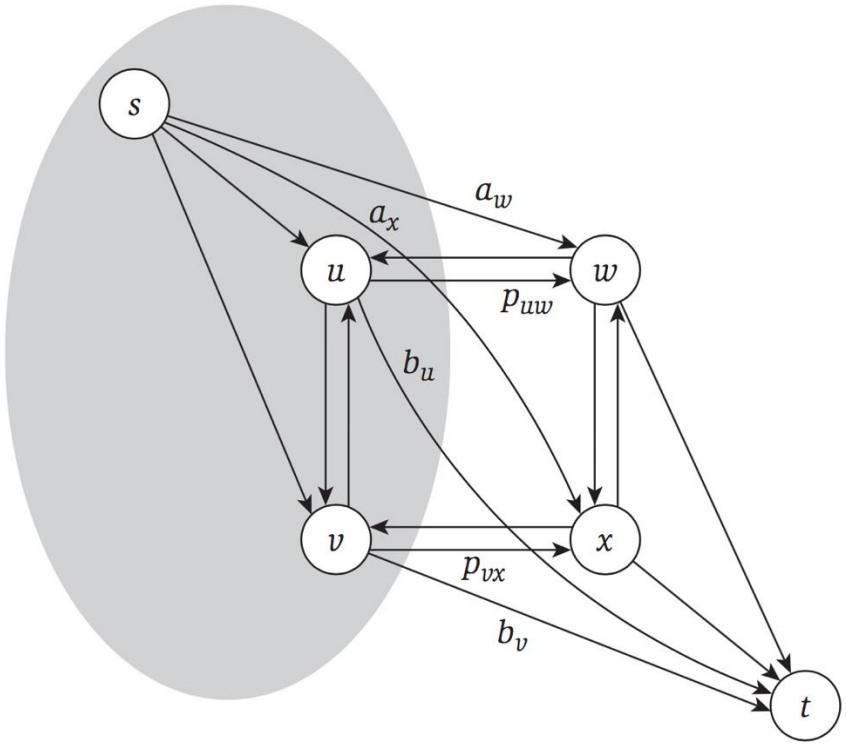
Step 1: Transform the Input



Step 2: Receive the Output



Step 3: Transform the Output



Summary

Solving minimum s-t cut in a graph with.
$n + 2$ nodes and $2m + 2n$ edges in time T

Solving image segmentation in a graph with n nodes and m edges in time $T + O(m)$

- Can solve image segmentation in $O(mn)$ time

Flow Applications Summary

- Network flow algorithms are powerful
  - Can use them to solve many optimization problems
  - Improvements for maxflow implies lots of new algorithms

- Many natural applications
  - Bipartite matching
  - Image segmentation
  - Airline scheduling
  - Fair division
  - Auction design
  - …

- Maxflow-Mincut duality (often) implies interesting duality theorems for these problems