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* $m^{1+o(1)}$

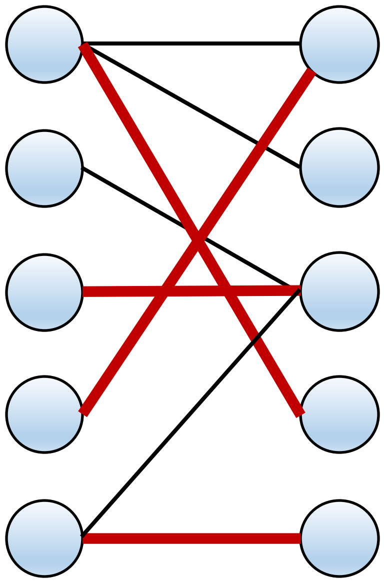
- **Today:** Using maximum-flow/minimum-cut as a building block for solving many more problems

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** = $|M|$

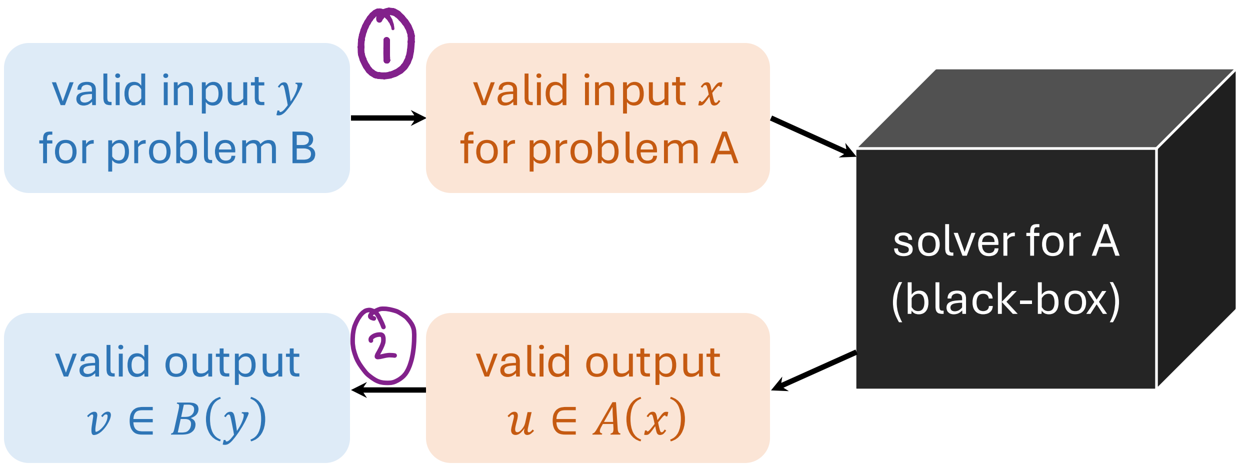
Models any problem where one type of object is assigned to another type:

- 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)

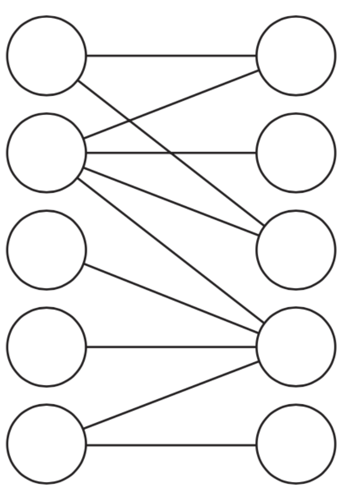


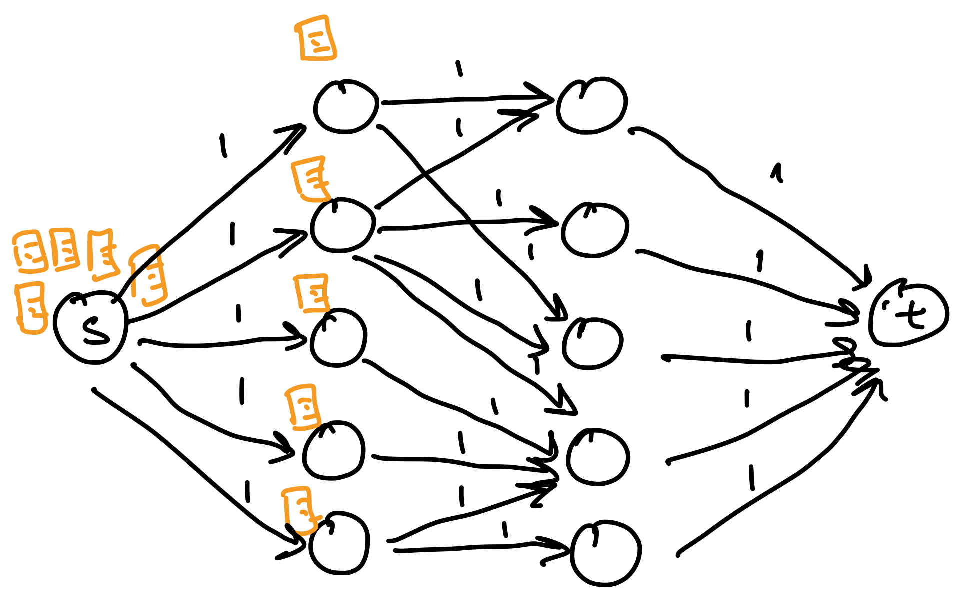
Step 1: Transform the Input

Input $G=(V,E)$
for MBM

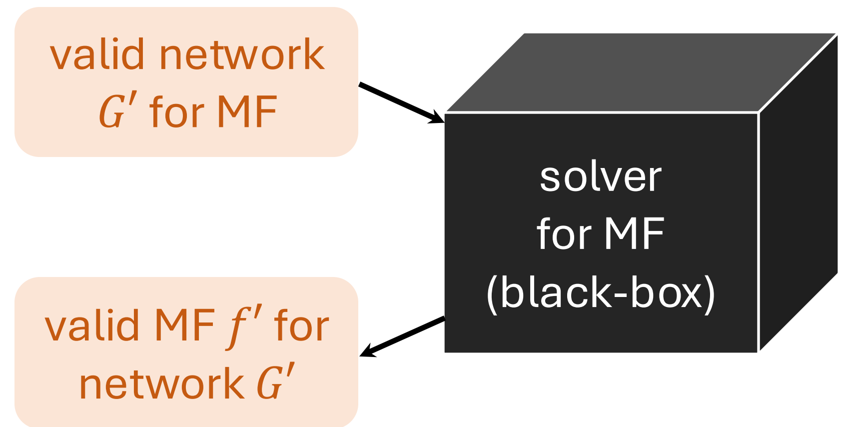
integers

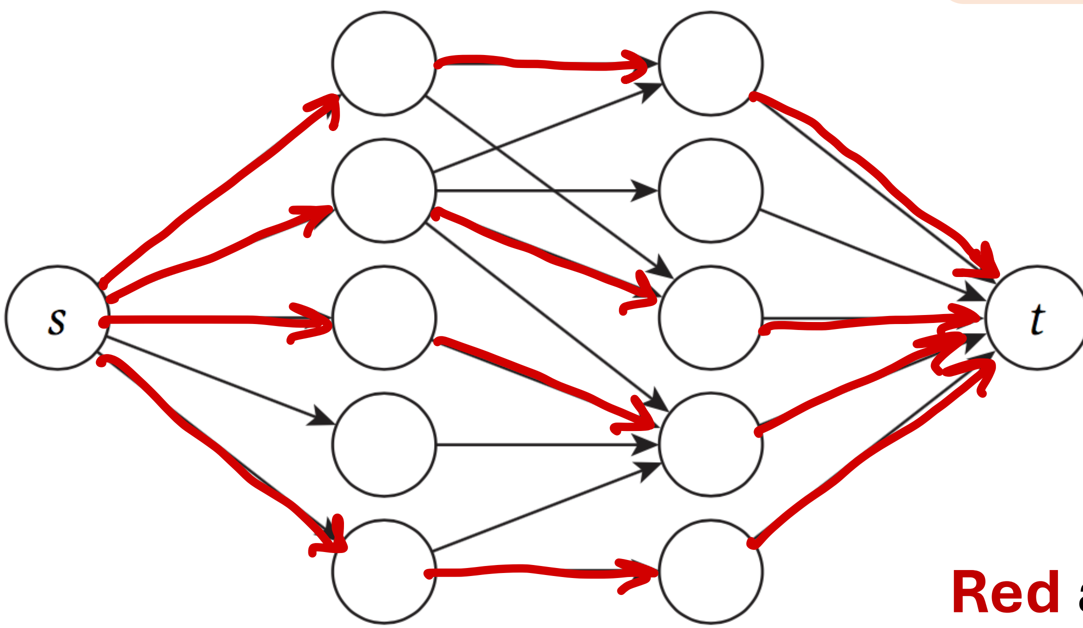
Input $G'=(V,E,s,t,\{c_e\})$
for MF





Step 2: Receive the Output





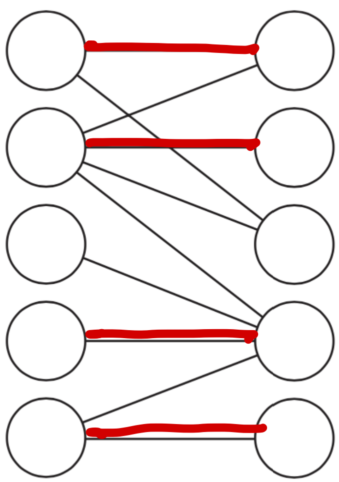
Red arrow means $f'(e) = 1$

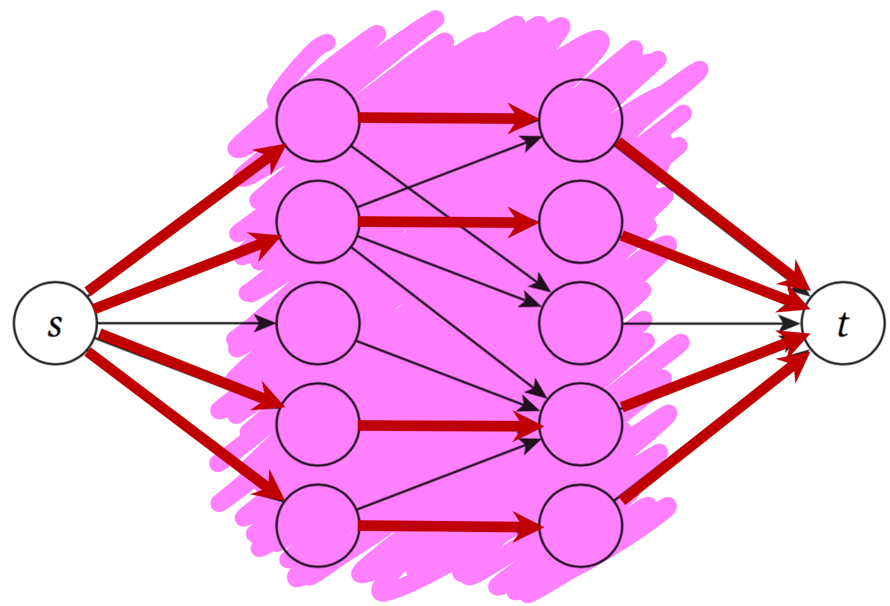
Black arrow means $f'(e) = 0$

Step 3: Transform the Output

valid MBM M for graph G ⬅ valid MF f' for network G'

$M = \{$ all $L \to R$ edges with $f'(e) = 1 \}$



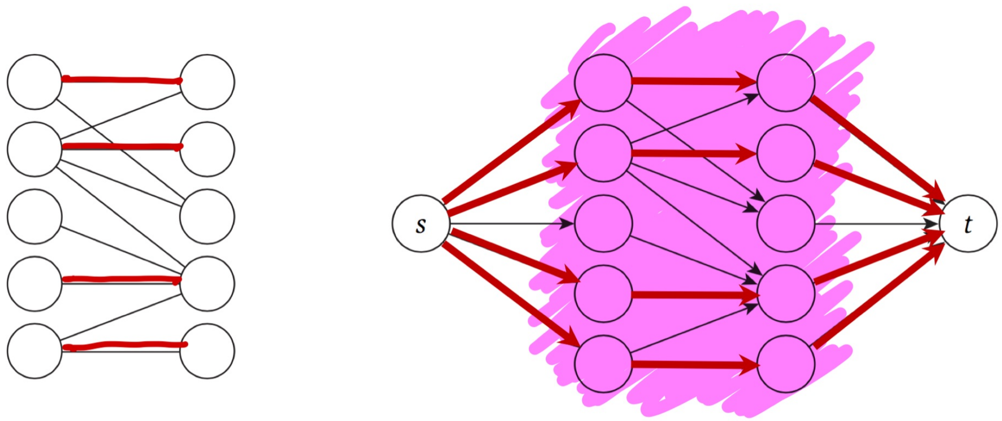


Correctness

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

Correctness

- Our algorithm returns a matching



- every node in L has 1 incoming edge, has at most 1 unit incoming flow

- every node in L has at most one outgoing edge of flow 1

- every node in L is in at most 1 pair in the matching

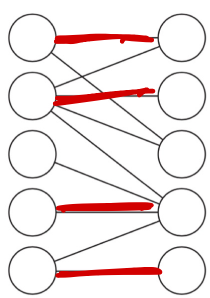
- "same for nodes in R"

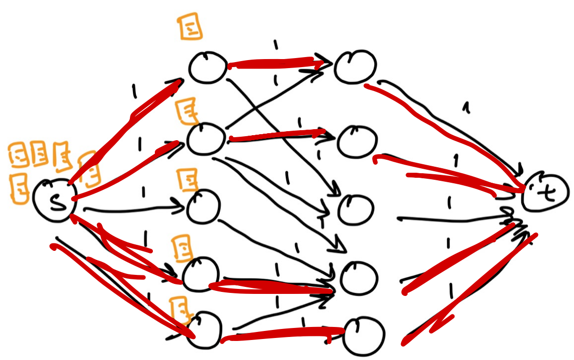
Correctness

- Out algorithm returns a maximum matching

Claim: G has a matching of size k if and only if G' has a flow of value k

① Matching of size k $\Longrightarrow$ Flow of value k



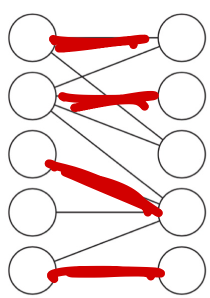


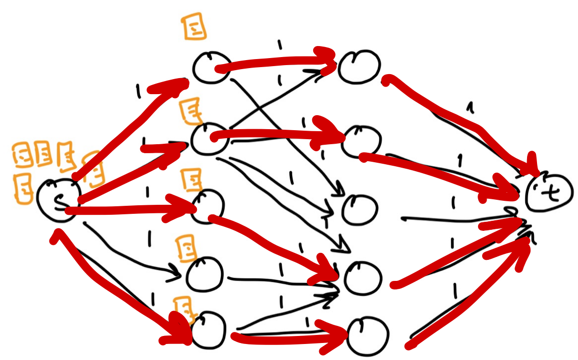
Correctness

- Out algorithm returns a maximum matching

Claim: G has a matching of size k if and only if G' has a flow of value k

(1) Matching of size k $\Longleftarrow$ Flow of value k





Running Time

- Need to analyze:
  - Time to transform the input → $O(m)$
  - Time to run the max-flow solver → $O(mn)$
  - Time to transform the otuput → $O(m)$

: Assuming we can solve max flow on networks with n' nodes and m' edges in time $O(m'n')$

- Our reduction takes a graph with n nodes, m edges and outputs a network with $n' = n+2$ nodes, and $m' = m+n$ edges

$$O((m+n)(n+2)) = O(mn)$$

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

$\Downarrow$

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!

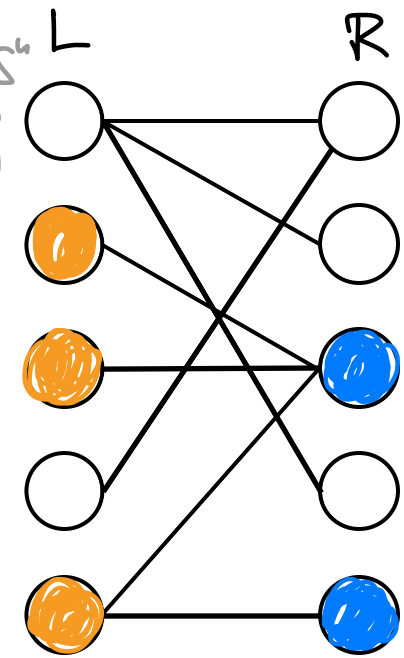
Hall's Theorem

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

Obstruction to a Perfect Matching:

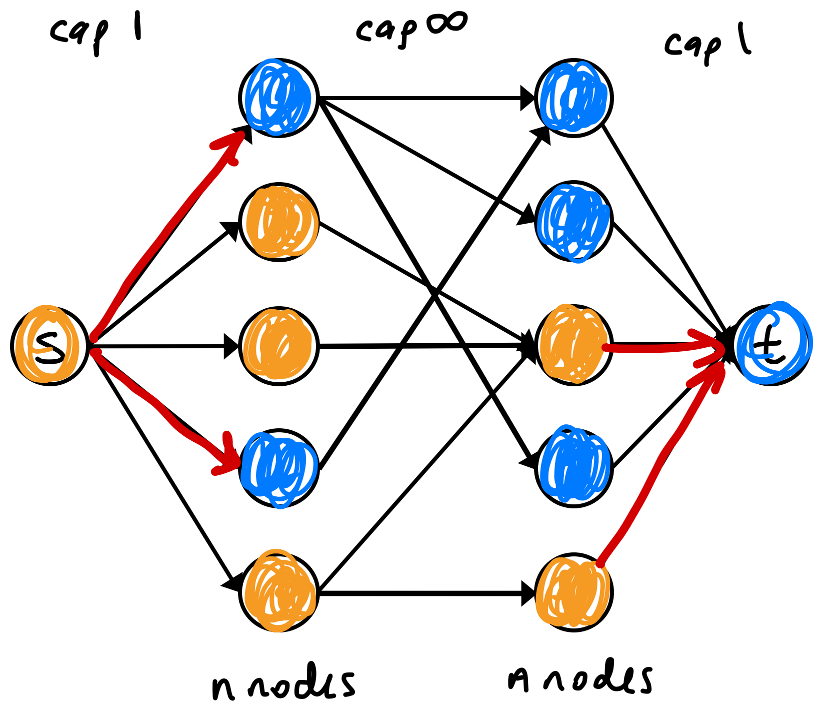
A set $S \subseteq L$ such that $|N(S)| < |S|$

$N(S)$ = "neighborhood of S"
$\{v : (u,v) \in E \text{ and } u \in S\}$



Hall's Theorem: A bipartite graph $G = (L \cup R, E)$ with $|L| = |R|$ has a perfect matching if and only if for every $S \subseteq L$, $|N(S)| \geq |S|$

Proof of Hall's Theorem via Duality



- Observation: Min cut A, B does not cut any $L \to R$ edges
- For every node in $L \cap A$, all of its neighbors are also in A
- $cap(A,B) = |L \setminus A| + |N(L \cap A)|$

$$cap(A,B) = n - |L \cap A| + |N(L \cap A)|$$

$$|N(L \cap A)| = |L \cap A| + (cap(A,B) - n)$$

$$|N(L \cap A)| = |L \cap A| + (val(f^*) - n)$$

If the value of the max flow is $< n$ then $|N(L \cap A)| < |L \cap A|$ $\implies$ If there is no perfect matching then $\exists\, S \subseteq L$ s.t. $|N(S)| < |S|$

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

$$\min_{A,B} \sum_{i\in A} -a_i + \sum_{j\in B} -b_j + \sum_{\substack{i,j \\ \text{btw } A,B}} p_{ij}$$

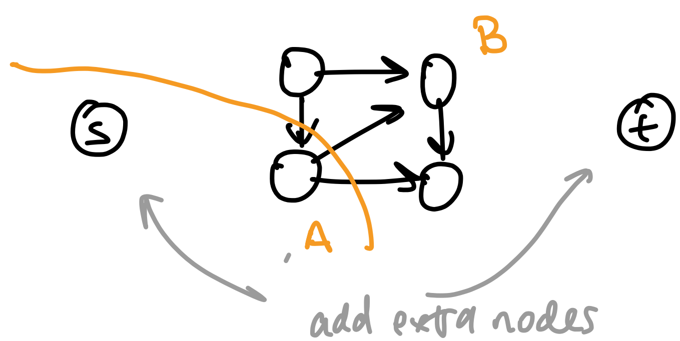
$$\min_{A,B} \sum_{i\in A} a_i + b_i + \text{''}$$

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

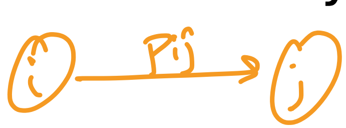
$$\max_{A,B} \sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{\substack{(i,j)\in E \\ \text{btw } A \text{ and } B}} p_{ij} \quad \Longleftrightarrow \quad \min_{A,B} \sum_{i\in A} b_i + \sum_{j\in B} a_j + \sum_{\substack{(i,j)\in E \\ \text{btw } A \text{ and } B}} p_{ij}$$

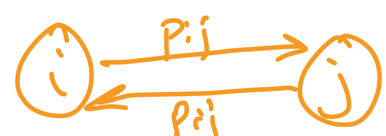
minimized by same cut A, B

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



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



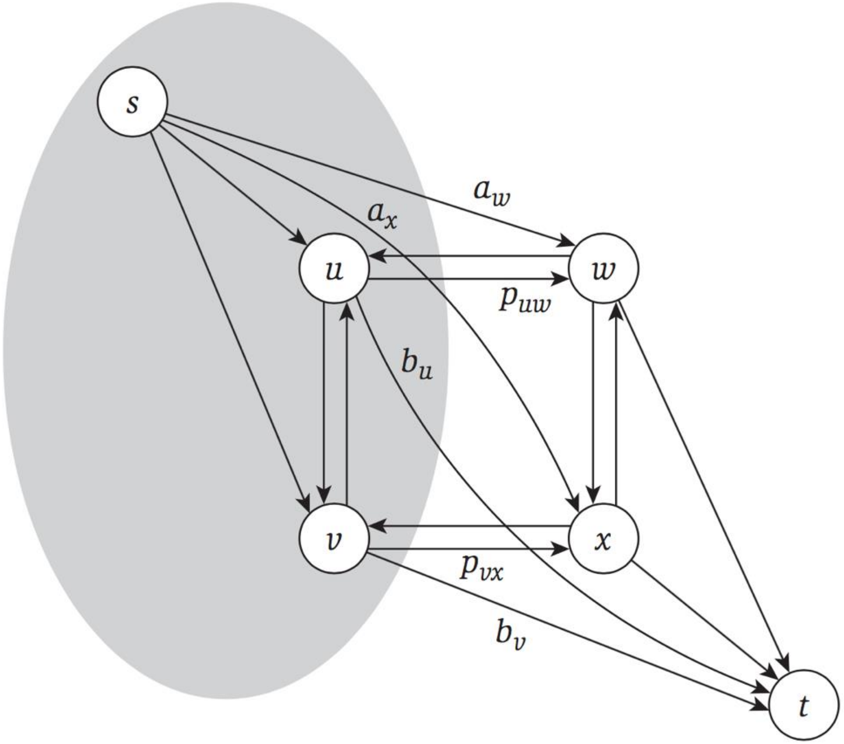


Reduction to MinCut

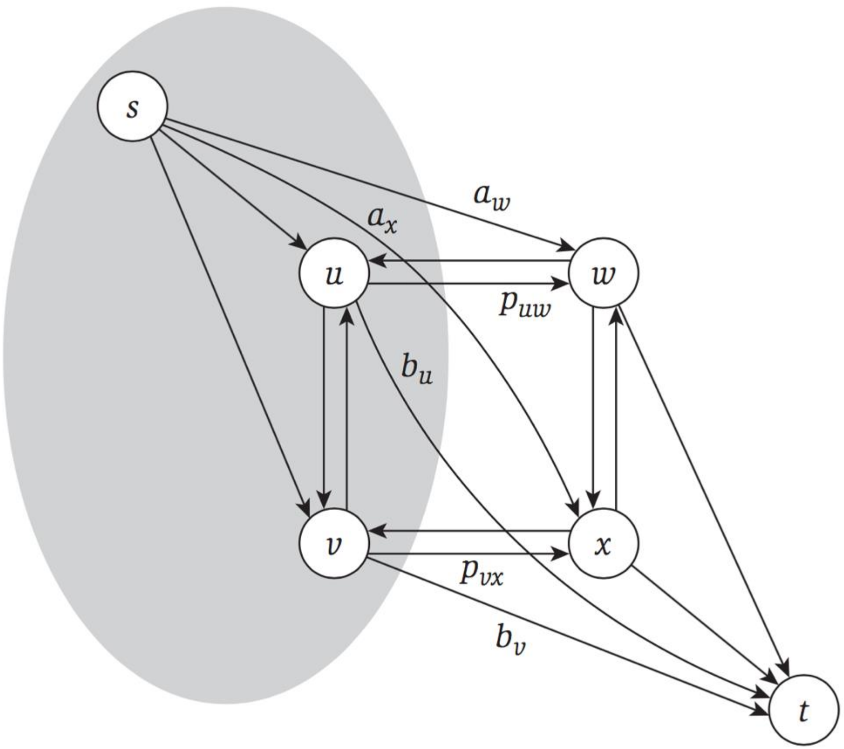
- How can we set up a flow network where the cost of the segmentation is the capacity of a cut

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

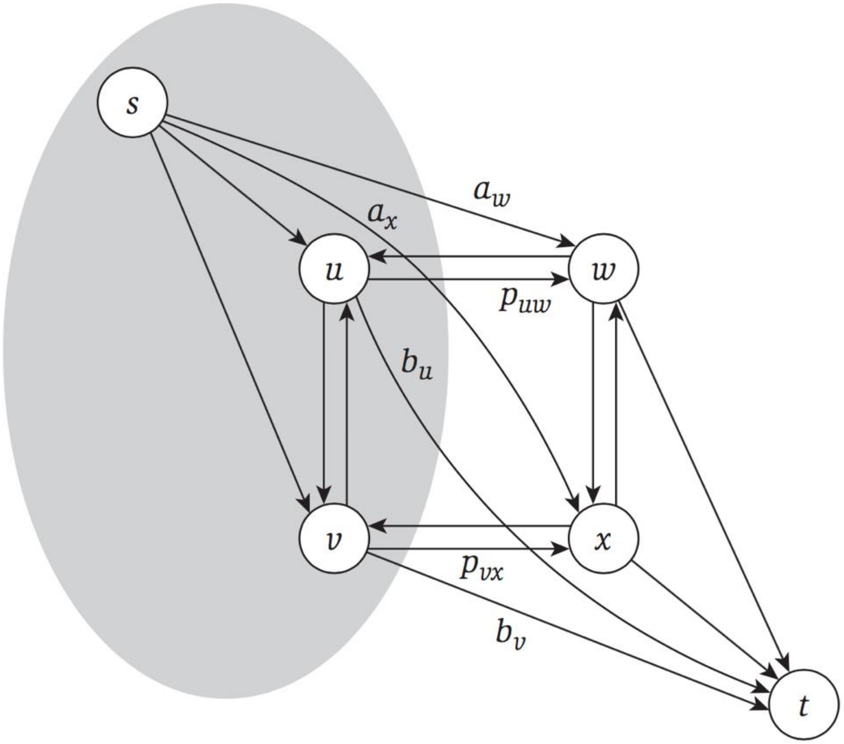
Step 1: Transform the Input



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

$\Downarrow$

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