CS 7800: Advanced Algorithms

Class 14: Reductions and Intractability

- Finish Image Segmentation
- NP-Completeness

Jonathan Ullman

October 17, 2025

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

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

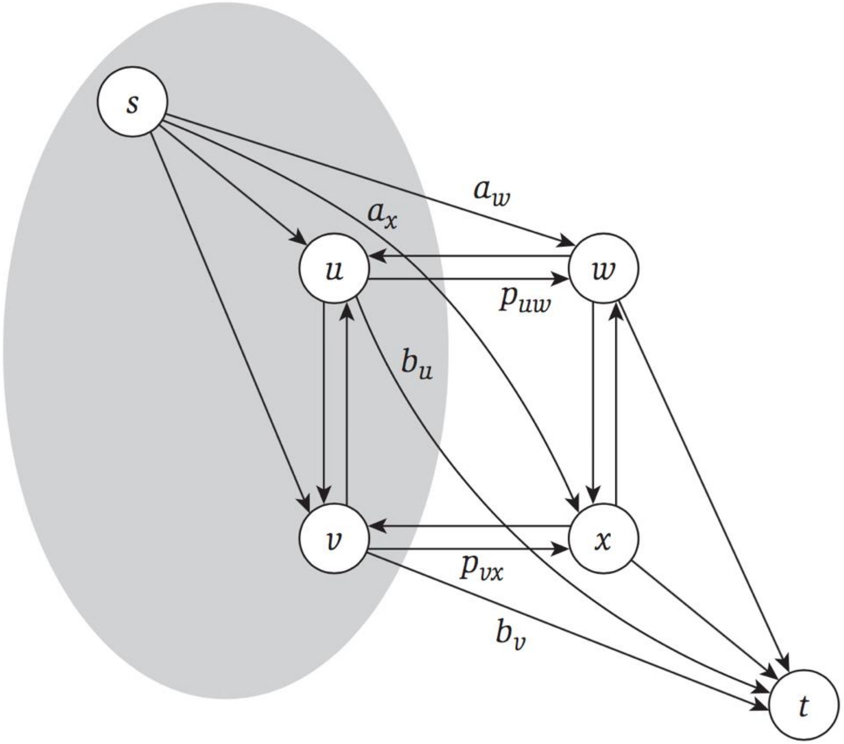
  - SEG allows any partition, MINCUT requires $s \in A, t \in B$
  - SEG counts any cut edge, MINCUT counts $A \to B$ edges

Reduction to MinCut

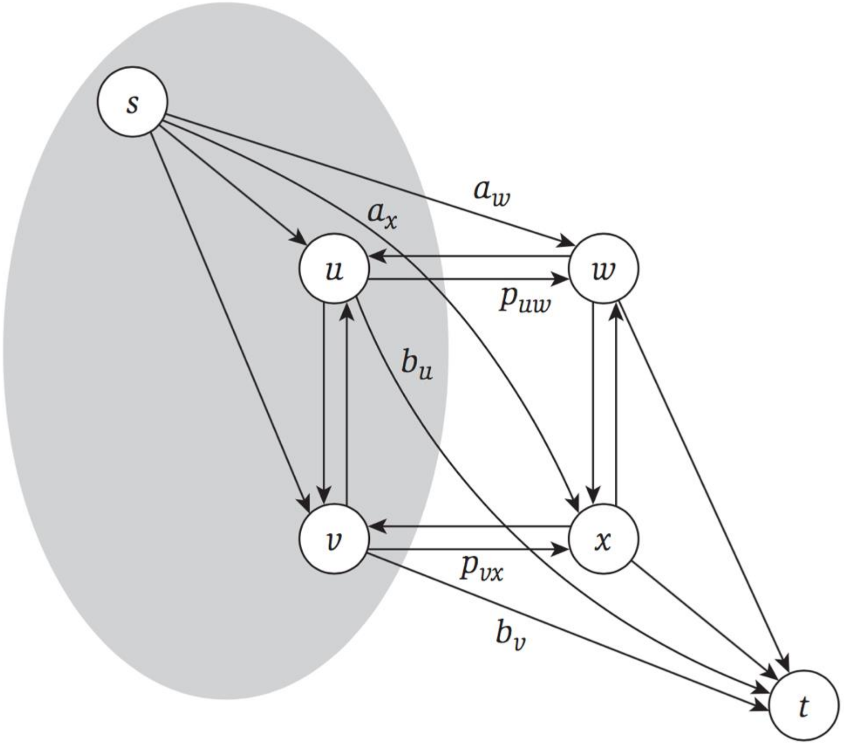
- How can we set up a flow network where the cost of the segmentation is the capacity of a cut

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

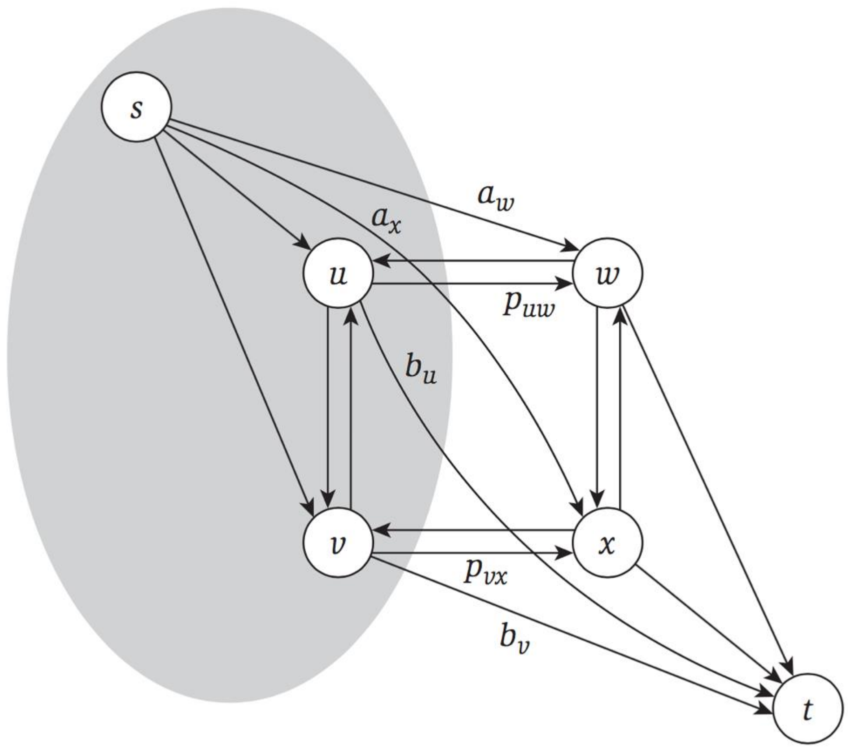
Step 1: Transform the Input



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

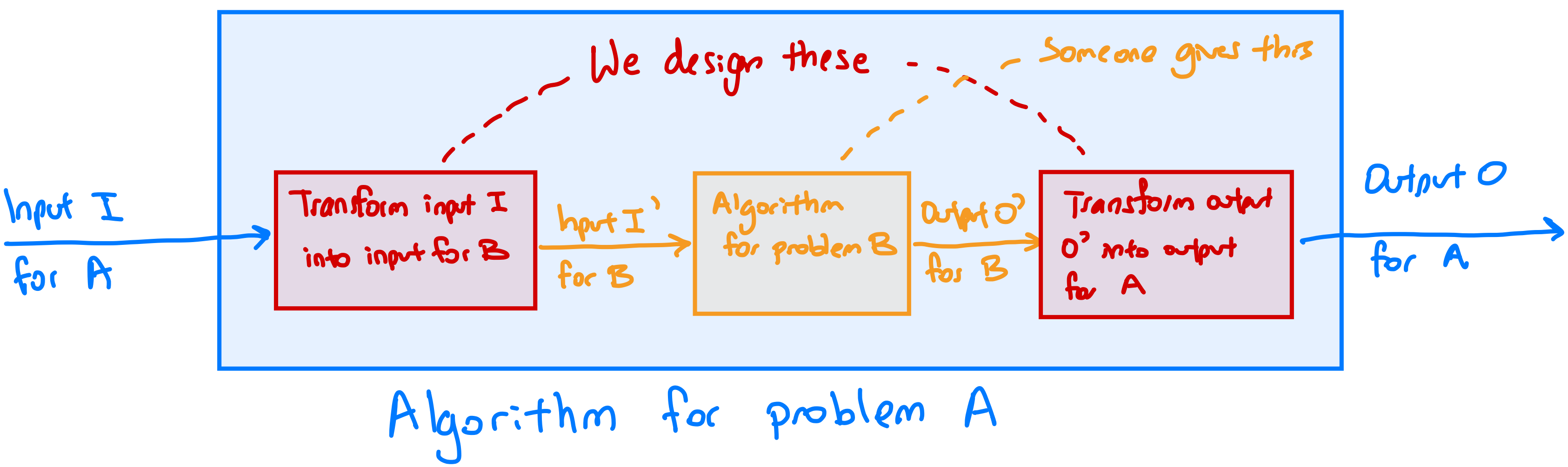
- Many natural applications
  - Bipartite matching
  - Image segmentation
  - Airline scheduling
  - Fair division
  - Auction design
  - ...

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

Reductions

Reduction: Problem A reduces to Problem B if there is a polynomial-time algorithm that solves A using any efficient algorithm that solves B.

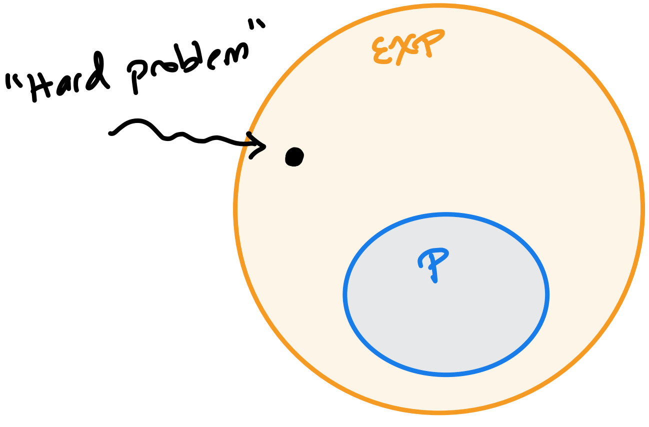
- Denoted $A \leq_P B$ (i.e. "A is at most as hard as B")
- **View 1:** If B can be solved efficiently then so can A
- **View 2:** If A can't be solved efficiently then neither can B



Tractable and Intractable Problems

- **Definition:** $\mathcal{P}$ is the set of decision problems that can be solved in polynomial time

- **Definition**: $\mathcal{EXP}$ is the set of decision problems that can be solved in exponential time
- **Theorem:** $\mathcal{P} \neq \mathcal{EXP}$



Allegedly Intractable Problems

INDSET $\underbrace{\equiv_p}$ CLIQUE $\equiv_p$ VERTEX COVER

Reduction in both directions ("Equally hard")

Allegedly Intractable Problems

VERTEX COVER $\leq_p$ SET COVER

Allegedly Intractable Problems

VERTEX COVER $\leq_p$ ILP (0/1 Integer Linear Programming)

Allegedly Intractable Problems

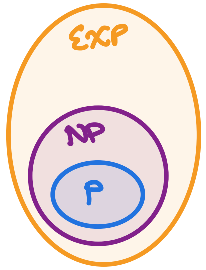
3-SAT $\leq_p$ INDSET

Allegedly Intractable Problems

$$
\begin{tikzcd}
 & & \text{SETCOVER} \\
 & \text{VERTEX COVER} \arrow[ur, "\leq_p"] \arrow[r, "\leq_p"] \arrow[d, phantom, "\equiv_p"] & \text{ILP} \\
 & \text{INDSET} \arrow[d, phantom, "\equiv_p"] & \\
\text{3-SAT} \arrow[ur, "\leq_p"] & \text{CLIQUE} &
\end{tikzcd}
$$

Note: Reductions are transitive

The Class NP



- **Definition:** $\mathcal{NP}$ is the class of problems for which there is an efficient verifier for solutions
  - An algorithm V is an efficient verifier for problem A if
    - (1) V takes as input I and a solution S
    - (2) V is a polynomial-time algorithm
    - (3) $I \in A$ if and only if there exists a polynomial-size solution S such that $V(I, S) = \text{YES}$

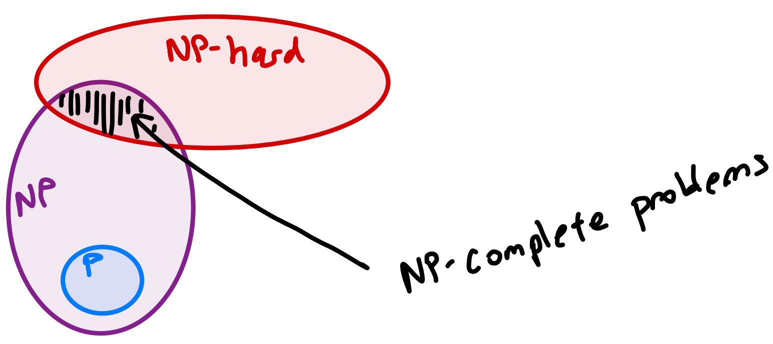
- $\mathcal{P}$ = easy to solve, $\mathcal{NP}$ = easy to check solution

- Natural hard optimization problems are in $\mathcal{N}P$
  - 3-SAT, Vertex-Cover, Independent-Set…

Does $\mathcal{P} = \mathcal{NP}$?

- We do not know, but we believe it very strongly!
  - One of the Millenium Problems

- If we believe $\mathcal{P} \neq \mathcal{NP}$ what does that tell us about problems we care about?
  - **Def:** B is $\mathcal{NP}$-hard if for $A \in \mathcal{NP}$, $A \leq_P B$
  - **Def:** B is $\mathcal{NP}$-complete if $B \in \mathcal{NP}$ and B is $\mathcal{NP}$-hard
  - If B is $\mathcal{NP}$-hard and $B \in \mathcal{P}$ then $\mathcal{P} = \mathcal{NP}$



What problems are $\mathcal{NP}$-complete?

- The Circuit Satisfiability Problem (CKT-SAT)
  - **Input:** Circuit C with n wires and AND/OR/NOT gates
  - **Output:** Decide if there exists x such that $C(x) = 1$

- **Thm:** CIRCUIT-SAT is $\mathcal{NP}$-complete

What problems are $\mathcal{NP}$-complete?

($\Rightarrow$ 3SAT is NPC)

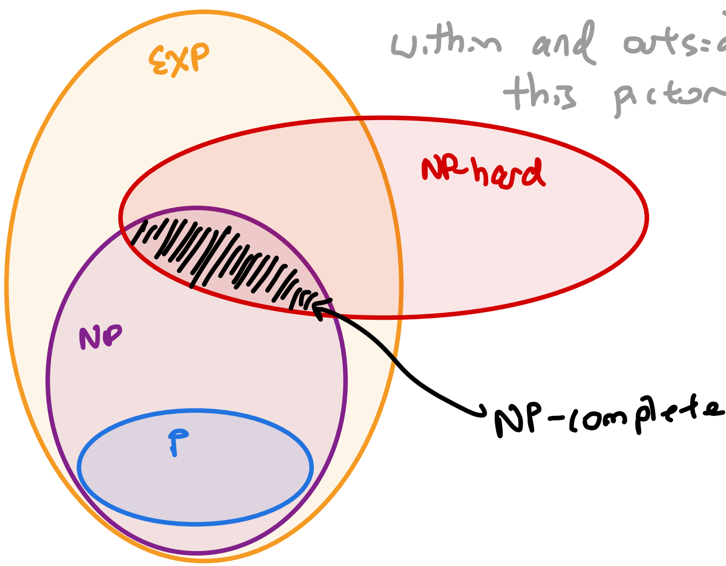
- **Thm (Cook '71, Levin '73):** CKT-SAT $\leq_P$ 3-SAT

What problems are $\mathcal{NP}$-complete?

($\Rightarrow$ 3-SAT is NPC)

- **Thm (Cook '71, Levin '73):** CKT-SAT $\leq_P$ 3-SAT
  - Now we know IND-SET, CLIQUE, VERTEX-COVER, SET-COVER, IP, and 3-SAT are all $\mathcal{NP}$-complete
  - There are thousands more known $\mathcal{NP}$-complete problems in essentially every area within CS

Lots of interesting classes within and outside of this picture.



What problems are $\mathcal{NP}$-complete?

- **Thm (Cook '71, Levin '73):** CKT-SAT $\leq_P$ 3-SAT
  - Now we know IND-SET, CLIQUE, VERTEX-COVER, SET-COVER, IP, and 3-SAT are all $\mathcal{NP}$-complete
  - There are thousands more known $\mathcal{NP}$-complete problems in essentially every area within CS