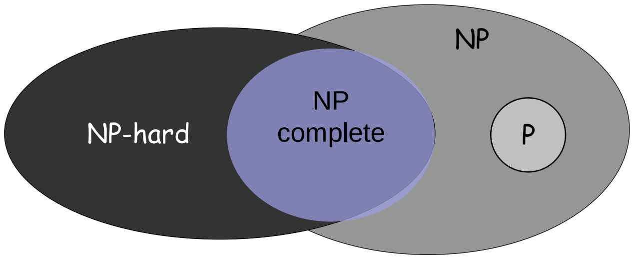
CSE 421

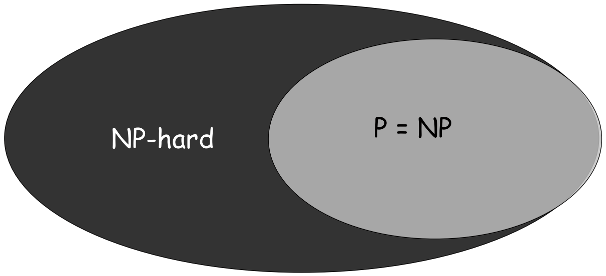
Polynomial Time Reductions, NP, NP-Completeness

Shayan Oveis Gharan

P vs NP



If $P \neq NP$



If $P = NP$

Cook-Levin Theorem

Theorem (Cook 71, Levin 73): 3-SAT is NP-complete, i.e., for all problems $A \in NP$, $A \leq_p$ 3-SAT.

- So, 3-SAT is the hardest problem in NP.

What does this say about other problems of interest? Like Independent set, Vertex Cover, …

Fact: If $A \leq_p B$ and $B \leq_p C$ then, $A \leq_p C$

Pf idea: Just compose the reductions from A to B and B to C

So, if we prove 3-SAT $\leq_p$ Independent set, then Independent Set, Clique, Vertex cover, Set cover are all NP-complete

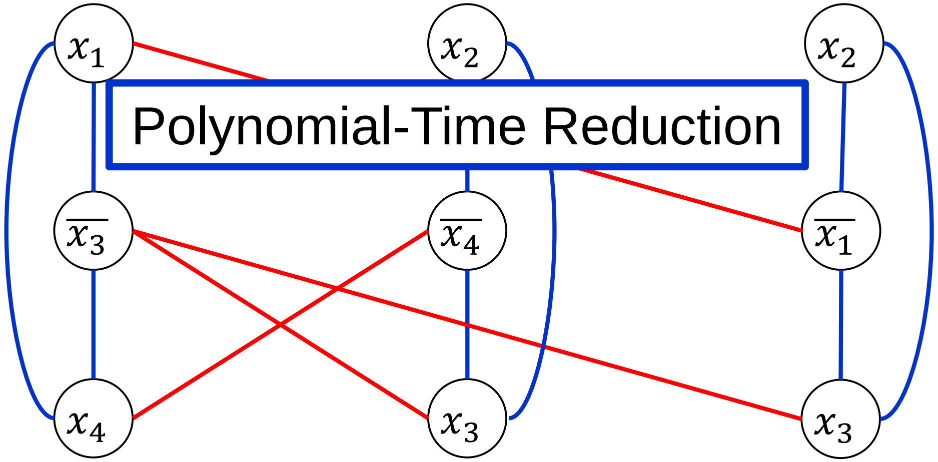
3-SAT $\leq_p$ Independent Set $\leq_p$ Vertex Cover $\leq_p$ Set Cover

3-SAT $\leq_p$ Independent Set

Map a 3-CNF to (G,k). Say m is number of clauses

- Create a vertex for each literal
- Joint two literals if
  - They belong to the same clause (blue edges)
  - The literals are negations, e.g., $x_i, \overline{x_i}$ (red edges)
- Set k=m

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee \overline{x_4} \vee x_3) \wedge (x_2 \vee \overline{x_1} \vee x_3)$$



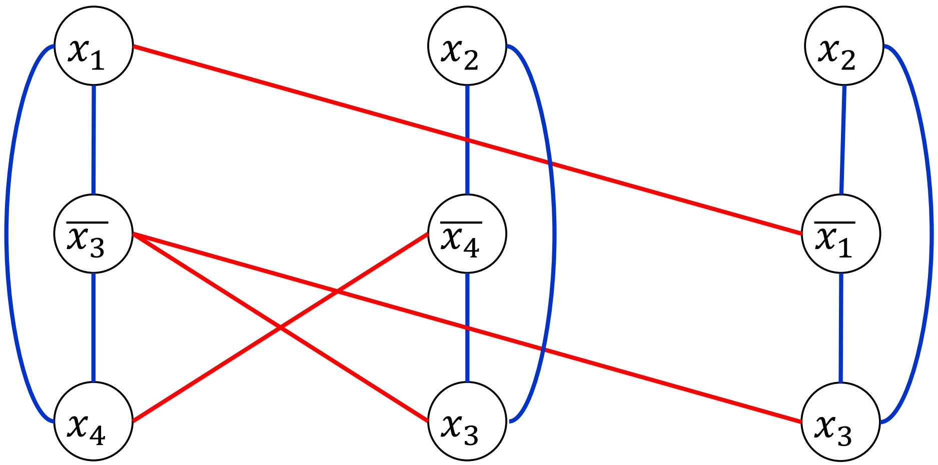
Correctness of 3-SAT $\leq_p$ Indep Set

F satisfiable => An independent of size m

Given a satisfying assignment, Choose one node from each clause where the literal is satisfied

$$(x_1 \lor \overline{x_3} \lor x_4) \land (x_2 \lor \overline{x_4} \lor x_3) \land (x_2 \lor \overline{x_1} \lor x_3)$$

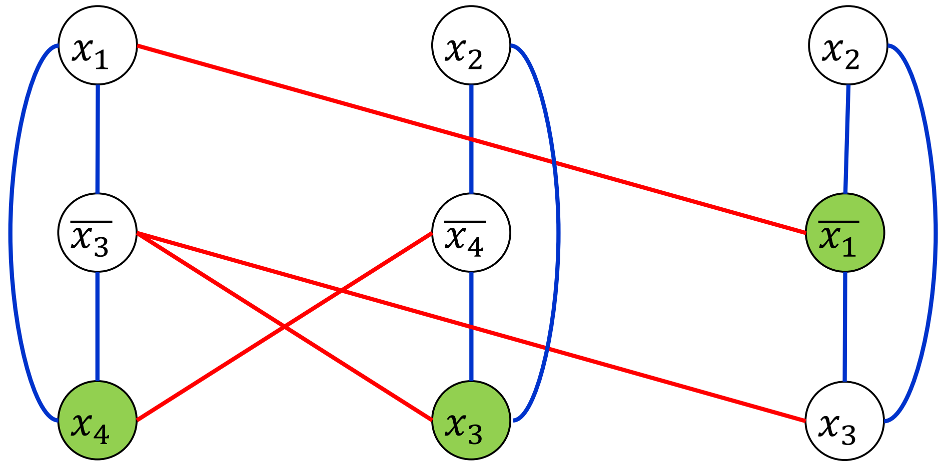
Satisfying assignment: $x_1 = T, x_2 = F, x_3 = T, x_4 = F$



- S has exactly one node per clause => No blue edges between S
- S follows a truth-assignment => No red edges between S
- S has one node per clause => |S|=m

Correctness of 3-SAT $\leq_p$ Indep Set

An independent set of size m => A satisfying assignment
Given an independent set S of size m.
S has exactly one vertex per clause (because of blue edges)
S does not have $x_i, \overline{x_i}$ (because of red edges)
So, S gives a satisfying assignment



Satisfying assignment: $x_1 = F, x_2 =?, x_3 = T, x_4 = T$

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee \overline{x_4} \vee x_3) \wedge (x_2 \vee \overline{x_1} \vee x_3)$$

Summary

- If a problem is NP-hard it does not mean that all instances are hared, e.g., Vertex-cover has a polynomial-time algorithm in trees
- We learned the crucial idea of polynomial-time reduction. This can be even used in algorithm design, e.g., we know how to solve max-flow so we reduce image segmentation to max-flow
- NP-Complete problems are the hardest problem in NP
- NP-hard problems may not necessarily belong to NP.
- Polynomial-time reductions are transitive relations

Linear Programming

Linear System of Equations

In high school we learn Gaussian elimination algorithm to solve a system of linear equations

$$\begin{aligned} x_1 + x_3 &= 7 \\ 2x_2 + x_1 &= 5 \\ 3x_1 + 7x_2 - x_3 &= 1 \end{aligned}$$

We set $x_3 = 7 - x_1$ and we substitute in the following equations.

Then we substitute $x_2 = \frac{5-x_1}{2}$ in to the third equations.

The third equational uniquely defines x_1

Linear Programming

Optimize a linear function subject to linear inequalities

$$
\begin{aligned}
\max \quad & 3x_1 + 4x_3 \\
s.t., \quad & x_1 + x_2 \leq 5 \\
& x_3 - x_1 = 4 \\
& x_3 - x_2 \geq -5 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

- We can have inequalities,
- We can have a linear objective funtions

Applications of Linear Programming

Generalizes: $Ax=b$, 2-person zero-sum games, shortest path, max-flow, matching, multicommodity flow, MST, min weighted arborescence, …

Why significant?

- We can solve linear programming in polynomial time.
- Useful for approximation algorithms
- We can model many practical problems with a linear model and solve it with linear programming

Linear Programming in Practice:

- There are very fast implementations: IBM CPLEX, Gorubi in Python, CVX in Matlab, ….
- CPLEX can solve LPs with millions of variables/constraints in minutes

Example 1: Diet Problem

Suppose you want to schedule a diet for yourself. There are four category of food: veggies, meat, fruits, and dairy. Each category has its own (p)rice, (c)alory and (h)appiness per pound:

	veggies	**meat**	**fruits**	**dairy**
price	p_v	p_m	p_f	p_d
calorie	c_v	c_m	c_f	c_d
happiness	h_v	h_m	h_f	h_d

Suppose we model this as a linear model, i.e., if we eat 0.5lb of meat an 0.2lb of fruits we will be $0.5\ h_m + 0.2\ h_f$ happy

- You should eat 1500 calaroies to be healthy
- You can spend 20 dollars a day on food.

Goal: Maximize happiness?

Diet Problem by LP

- You should eat 1500 calaroies to be healthy
- You can spend 20 dollars a day on food.

Goal: Maximize happiness?

	veggies	meat	fruits	dairy
price	p_v	p_m	p_f	p_d
calorie	c_v	c_m	c_f	c_d
happiness	h_v	h_m	h_f	h_d

$$\begin{array}{ll} \max & x_v h_v + x_m h_m + x_f h_f + x_d h_d \\ s.t. & x_v p_v + x_m p_m + x_f p_f + x_d p_d \leq 20 \\ & x_v c_v + x_m c_m + x_f c_f + x_d c_d \leq 1500 \\ & x_v, x_m, x_f, x_d \geq 0 \end{array}$$

#pounds of veggies, meat, fruits, dairy to eat per day

How to Design an LP?

- Define the set of variables
- Put constraints on your variables,
  - should they be nonnegative?
- Write down the constraints
  - If a constraint is not linear try to approximate it with a linear constraint
- Write down the objective function
  - If it is not linear approximation with a linear function
- Decide if it is a minimize/maximization problem

Example 2: Max Flow

Define the set of variables

- For every edge e let x_e be the flow on the edge e

Put constraints on your variables

- $x_e \geq 0$ for all edge e (The flow is nonnegative)

Write down the constraints

- $x_e \leq c(e)$ for every edge e, (Capacity constraints)
- $\sum_{e \text{ out of } v} x_e = \sum_{e \text{ in to } v} x_e \quad \forall v \neq s, t$ (Conservation constraints)

Write down the objective function

- $\sum_{e \text{ out of } s} x_e$

Decide if it is a minimize/maximization problem

- max

Example 2: Max Flow

$$\begin{aligned}
\max \quad & \sum\nolimits_{e \text{ out of } s} x_e \\
s.t. \quad & \sum\nolimits_{e\ out\ of\ v} x_e = \sum\nolimits_{e\ in\ to\ v} x_e && \forall v \neq s, t \\
& x_e \leq c(e) && \forall e \\
& x_e \geq 0 && \forall e
\end{aligned}$$

Q:Do we get exactly the same properties as Ford Fulkerson?
A: Not necessarily, the max-flow may not be integral

Example 3: Min Cost Max Flow

Suppose we can route 100 gallons of water from s to t.
But for every pipe edge e we have to pay $p(e)$ for each gallon of water that we send through e.

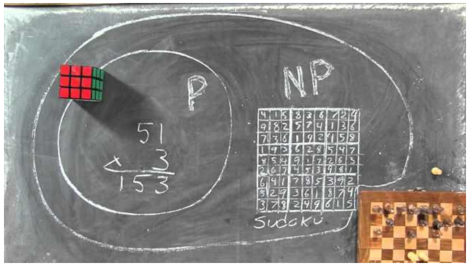
Goal: Send 100 gallons of water from s to t with minimum possible cost

$$
\begin{aligned}
\min \quad & \sum_{e \in \mathrm{E}} p(e) \cdot x_e \\
s.t. \quad & \sum_{e \text{ out of } v} x_e = \sum_{e\, in\, to\, v} x_e \qquad \forall v \neq s, t \\
& \sum_{e \text{ out of } s} x_e = 100 \\
& x_e \leq c(e) \qquad \forall e \\
& x_e \geq 0 \qquad \forall e
\end{aligned}
$$

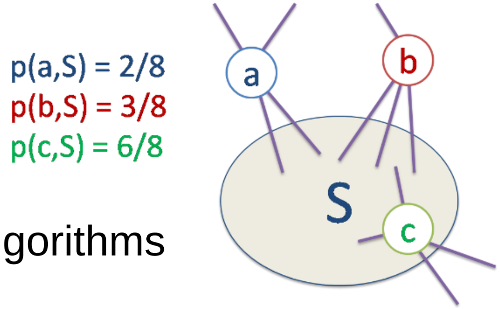
Summary (Linear Programming)

- Linear programming is one of the biggest advances in 20^{th} century
- It is being used in many areas of science: Mechanics, Physics, Operations Research, and in CS: AI, Machine Learning, Theory, …
- Almost all problems that we talked can be solved with LPs, Why not use LPs?
  - Combinatorial algorithms are typically faster
  - They exhibit a better understanding of worst case instances of a problem
  - They give certain structural properties, e.g., Integrality of Max-flow when capacities are integral
- There is rich theory of LP-duality which generalizes max-flow min-cut theorem

What is next?



- CSE 431 (Complexity Course)
  - Learn about how to prove lower bounds on algorithms



- CSE 521 (Graduate Algorithms Course)
  - How to design streaming algorithms?
  - How to design algorithms for high dimensional data?
  - How to use matrices/eigenvalues/eigenvectors to design algorithms
  - How to use LPs to design algorithms?

- CSE 525 (Graduate Randomized Algorithms Course)
  - How to use randomization to design algorithms?
  - How to use Markov Chains to design algorithms?