COMPSCI 311: Introduction to Algorithms

Lecture 22: Intractability: SAT, NP

Dan Sheldon

University of Massachusetts Amherst

Review: Polynomial-Time Reduction

- $Y \leq_P X$: Problem Y is **polynomial-time reducible** to Problem X,

```
solveY(yInput)
  Construct xInput              // poly-time
  foo = solveX(xInput)          // poly # of calls
  return yes/no based on foo // poly-time
```

- ... if any instance of Problem Y can be solved using
  1. A polynomial number of standard computational steps
  2. A polynomial number of calls to a black box that solves problem X
- Statement about relative hardness
  1. If $Y \leq_P X$ and $X \in P$, then $Y \in P$
  2. If $Y \leq_P X$ and $Y \notin P$ then $X \notin P$

Reduction Strategies

- Reduction by equivalence
  ($\textsc{Vertex-Cover} \leq_P \textsc{Indept-Set}$ and vice versa)
- Reduction to a more general case
  ($\textsc{Vertex-Cover} \leq_P \textsc{Set-Cover}$)
- Reduction by "gadgets"

Reduction by Gadgets: Satisfiability

- Can we determine if a Boolean formula has a satisfying assignment?

$$\underbrace{(x_1 \vee \bar{x}_2)}_{\text{"clause"}} \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$$

- Terminology

Variables	$x_1, \ldots, x_n$	
Term / literal	x_i or $\bar{x}_i$	variable or its negation
Clause	$C = \bar{x}_1 \vee x_2 \vee \bar{x}_3$	"or" of terms
Formula	$C_1 \wedge C_2 \wedge \ldots \wedge C_k$	"and" of clauses
Assignment	$(x_1, x_2, x_3) = (1, 0, 1)$	assign 0/1 to each variable
Satisfying assigment	$(x_1, x_2, x_3) = (1, 1, 0)$	all clauses are "true"

Reduction by Gadgets: Satisfiability

SAT – Given boolean formula $C_1 \wedge C_2 \ldots \wedge C_m$ over variables $x_1, \ldots, x_n$, does there exist a satisfying assignment?

$3\text{-}\mathrm{SAT}$ – Same, but each C_i has exactly three terms

$2\text{-}\mathrm{SAT}$ — each C_i has exactly two terms

Clicker. What is the strongest statement below that follows easily from the definitions above?

A. $2\text{-}\mathrm{SAT} \leq_P 3\text{-}\mathrm{SAT} \leq_P \mathrm{SAT}$

B. $2\text{-}\mathrm{SAT} \leq_P \mathrm{SAT}$ and $3\text{-}\mathrm{SAT} \leq_P \mathrm{SAT}$

C. $\mathrm{SAT} \leq_P 3\text{-}\mathrm{SAT} \leq_P 2\text{-}\mathrm{SAT}$

Reduction by Gadgets: Satisfiability

Claim: $3\text{-SAT} \leq_P \textsc{IndependentSet}$.

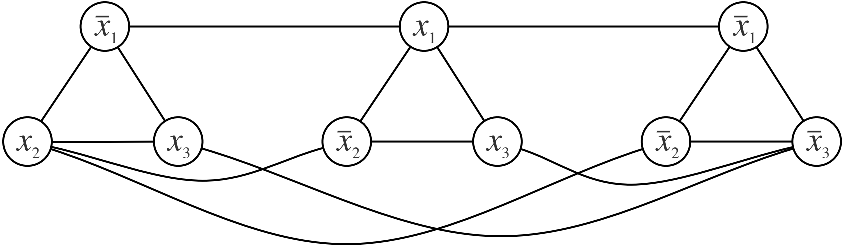
Reduction:

- Given 3-SAT instance $\Phi = \langle C_1, \ldots, C_m \rangle$, we will construct an independent set instance $\langle G, m \rangle$ such that G has an independent set of size m iff Φ is satisfiable
- Return $\textsc{Yes}$ if `solveIS`$(\langle G, m \rangle) = \textsc{Yes}$

Reduction

- **Idea**: construct graph G where independent set will select one term per clause to be true

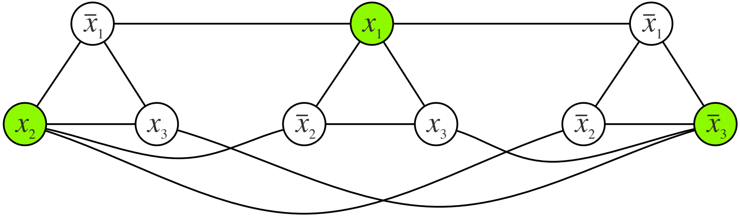
$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$



- One node per term
- Edges between all terms in same clause (select at most one)
- Edges between a literal and all of its negations (consistent truth assignment)

Correctness

$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

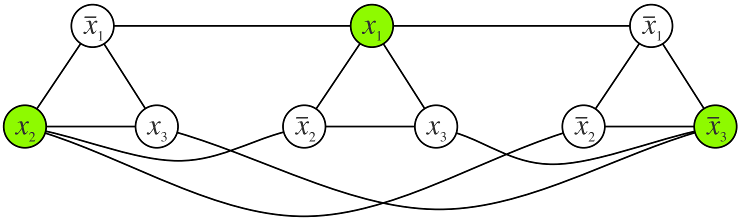


Claim: if G has an independent set of size m, then $\langle C_1, \ldots, C_m \rangle$ is satisfiable

- Suppose S is an independent set of size m
- Assign variables so selected literals are true. Edges from terms to negations ensure non-conflicting assignment.
- Set any remaining variables arbitrarily
- At most one term per clause is selected. Since m are selected, every clause is satisfied.

Correctness

$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

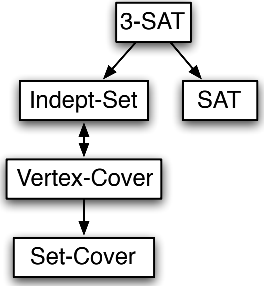


Claim: if $\langle C_1, \ldots, C_m \rangle$ is satisfiable, then G has an independent set of size m

- Consider any satsifying assignment of $\langle C_1, \ldots, C_m \rangle$
- Let S consist of one node per triangle corresponding to true literal in that clause. Then $|S| = m$.
- For (u, v) within clause, at most one endpoint is selected
- For edge $(x_i, \bar{x}_i)$ between clauses, at most one endpoint is selected, because $x_i = 1$ or $\bar{x}_i = 1$, but not both
- Therefore S is an independent set

Reductions So Far

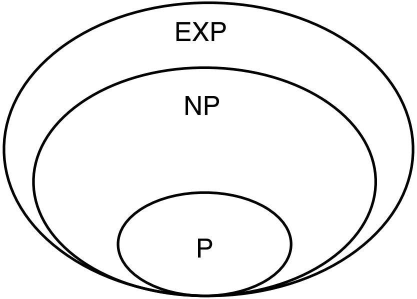
Partial map of problems we can use to solve others in polynomial time, through transitivity of reductions:



- $\boxed{Y} \longrightarrow \boxed{X}$
  means $Y \leq_P X$.

Toward a Definition of NP

Remember our problem hierarchy:



What is special about the mystery problems (NP)?

P and NP

Intuition. For many "hard" decision problems, at least one thing is "easy": if the correct answer is YES, there is an easy proof

- Independent set: show an independent set of size at least k
- SAT: show a satisfying assignment

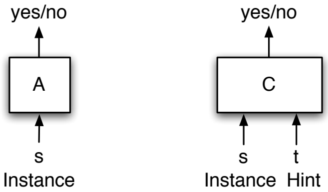
Problem classes

- P: Decision problems for which there is a polynomial time algorithm.
- NP: Decision problems for which there is a polynomial time certifier.
  - A solution can be "certified" in polynomial time.
  - NP $=$ "non-deterministic polynomial time"

Solver vs. Certifier

Let X be a decision problem and s be problem instance (e.g., $s = \langle G, k \rangle$ for Independent Set)

Poly-time solver. Algorithm $A(s)$ such that $A(s) = $ Yes iff correct answer is Yes, and running time polynomial time in $|s|$



Poly-time certifier. Algorithm $C(s, t)$ such that for every instance s, there is some t such that $C(s, t) = $ Yes iff correct answer is Yes, and running time is polynomial in $|s|$.

- t is the "certificate" or hint; size must also be polynomial in $|s|$

Certifier Example: Independent Set

Input $s = \langle G, k \rangle$.
Problem: Does G have an independent set of size at least k?
Idea: Certificate $t =$ an independent set of size k

`CertifyIS(` $\langle G, k \rangle, t$)
  if $|t| < k$ return $\textsc{No}$
  for each edge $e = (u, v) \in E$ **do**
    if $u \in t$ and $v \in t$ return $\textsc{No}$
  Return $\textsc{Yes}$

Polynomial time? Yes, linear in $|E|$.

Example: Independent Set

- INDEPENDENT SET $\in$ P?
  - Unknown. No known polynomial time algorithm.
- INDEPENDENT SET $\in$ NP?
  - Yes. Easy to certify solution in polynomial time.

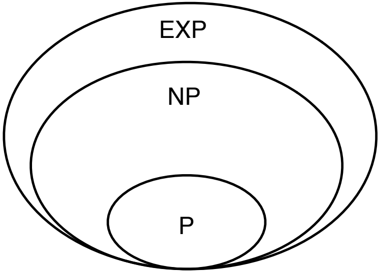
Example: 3-SAT

Input: formula Φ on n variables.
Problem: Is Φ satisfiable?
Idea: Certificate $t =$ the satisfying assignment

`Certify3SAT(` $\langle \Phi \rangle, t$)

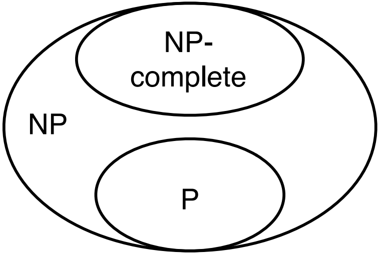
$\triangleright$ Check if t makes Φ true

P, NP, EXP



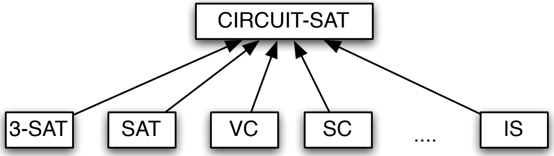
- 3SAT and Independent Set are in NP, as are many other problems that are hard to solve, but easy to certify!
- **Claim**: P $\subseteq$ NP
- **Claim**: NP $\subseteq$ EXP
- Both straightforward to prove, but not critical right now.

NP-Complete



- NP-complete = a problem $Y \in$ NP with the property that $X \leq_P Y$ for every problem $X \in$ NP!

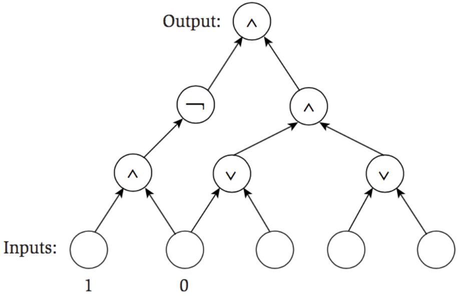
NP-Complete



- **Cook-Levin Theorem**: In 1971, Cook and Levin independently showed that particular problems were NP-Complete.
- We'll look at CIRCUIT-SAT as canonical NP-Complete problem.

CIRCUIT-SAT

Problem: Given a circuit built of AND, OR, and NOT gates with some hard-coded inputs, is there a way to set remaining inputs so the output is 1?



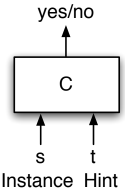
Satisfiable? Yes. Set inputs: 1, 1, 0.

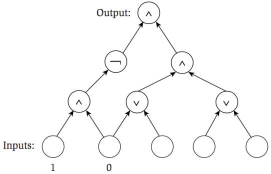
Circuit-SAT

Cook-Levin Theorem Circuit-SAT is NP-Complete.

Proof Idea: encode arbitrary certifier $C(s,t)$ as a circuit

- If $X \in$ NP, then X has a poly-time certifier $C(s,t)$:





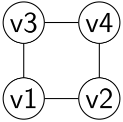
- s is Yes instance $\Leftrightarrow \exists\, t$ such that $C(s,t)$ outputs Yes
- Construct a circuit where s is hard-coded, and circuit is satsifiable iff $\exists\, t$ that causes $C(s,t)$ to output Yes
- s is Yes instance $\Leftrightarrow$ circuit is satisfiable
- Algorithm for Circuit-Sat implies an algorithm for X

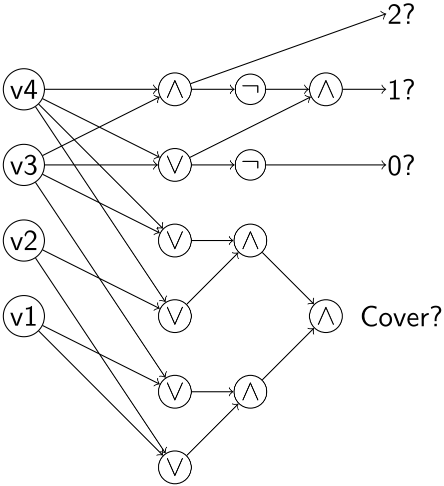
A Circuit-SAT reduction

See Independent Set example in other slides

A Circuit-SAT reduction

- Vertex Cover – Does G have VC of size at most k? (Counting gadget is an example for v_3, v_4 only)





Proving New Problems NP-Complete

Fact: If Y is NP-complete and $Y \leq_P X$, then X is NP-complete.

Want to prove problem X is NP-complete

- Check $X \in$ NP.
- Choose known NP-complete problem Y.
- Prove $Y \leq_P X$.

Clicker

It's easy to show that $3\text{-SAT} \leq_P \textsc{Circuit-SAT}$. What can we conclude from this?

A. 3-SAT is NP-complete.

B. 3-SAT is in NP.

C. If 3-SAT is NP-complete, then $\textsc{Circuit-SAT}$ is also NP-complete.

Proving New Problems NP-Complete

Theorem: 3-SAT is NP-Complete.

- In NP? Yes, check satisfying assignment in poly-time.
- Can show that $\text{Circuit-SAT} \leq_P 3\text{-SAT}$ (next time)

NP-Complete Problems: Preview