CS 312: Algorithms

Reductions and NP-Completeness

Dan Sheldon

Mount Holyoke College

Last Compiled: December 3, 2018

Polynomial-Time Reduction

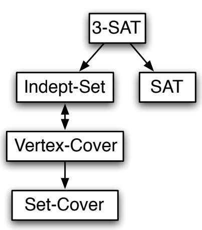
- $Y \leq_P X$

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

- Statement about relative hardness. Suppose $Y \leq_P X$, then:
  1. If X is solvable in poly-time, so is Y
  2. If Y is *not* solvable in poly-time, neither is X
- 1: design algorithms, 2: prove hardness

Preview

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



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

Reduction Strategies

- Reduction by equivalence (Vertex Cover and Indpendent Set)
- Reduction to a more general case
- Reduction by "gadgets"

Reduction by Gadgets: Satisfiability

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

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

- Terminology

Variables	$x_1, \ldots, x_n$	
Term	x_i or $\bar{x}_i$	variable or its negation
Clause	$C = x_1 \vee \bar{x}_2 \vee x_4$	"or" of terms
Formula	$C_1 \wedge C_2 \wedge C_3 \wedge C_4$	"and" of clauses
Assignment	$(x_1, x_2, x_3) = (1, 1, 1)$	assign 0/1 to each variable
Satisfying assigment	$(x_1, x_2, x_3) = (0, 0, 0)$	all clauses are "true"

Reduction by Gadgets: Satisfiability

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

3-SAT – Same, but each C_i has exactly three terms

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

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

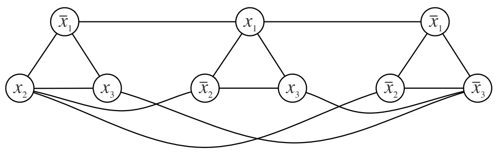
Reduction:

- Given 3-SAT instance Φ, 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

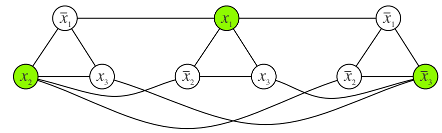
$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$



- 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 1

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

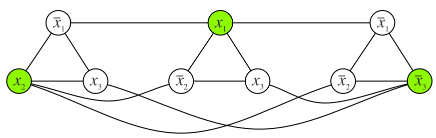


Claim: if G has an independent set of size m, then Φ is satisfiable

- 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 2

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

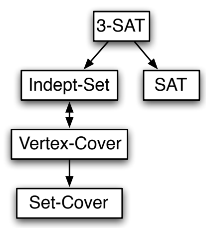


Claim: if Φ is satisfiable, then G has an independent set of size m

- Consider a satsifying assignment of Φ
- Let S consist of *one node* per triangle corresponding to true literal in that clause. Then $|S| = m$.
- For (u, v) within triangle, 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

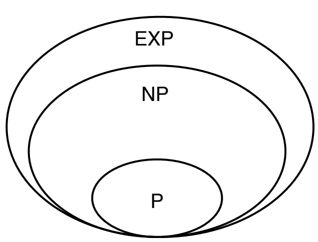
Partial map of problems we can use to solve others in polynomial time, through transitivity of reductions:



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

Toward a Definition of NP

Remember our mystery problems:



What is special about these?

P and NP

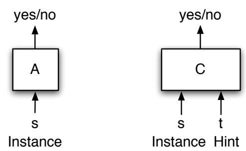
- P: Decision problems for which there is a polynomial time algorithm.
- NP: Decision problems for which there is a polynomial time certifier.

Intuition: A correct solution can be certified in 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 $A(s) = \text{YES}$ 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 $C(s,t) = \text{YES}$ t iff correct answer is YES, and running time is polynomial in $|s|$.

- t is the "certificate" or hint. Size of t 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?
Hint t: a candidate independent set U of size k

`CertifyIS(` $\langle G, k \rangle, U$`)`
- ▷ Check if size at most k
- **if** $|U| < k$ return NO

- ▷ Check if independent set
- **for** each edge $e = (u, v) \in E$ **do**
  - **if** $u \in U$ and $v \in U$ return NO
- **end for**
- Return YES

Polynomial time?

Example: Independent Set

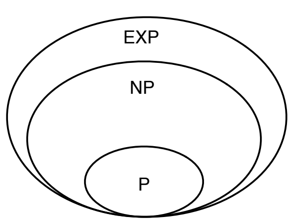
- INDEPENDENT SET $\in$ P?
  - Unknown. No known polynomial time algorithm.
- INDEPENDENT SET $\in$ NP?
  - Yes. Easy to certify solution in polynomial time.

Example: 3-SAT

Input: formula Φ on n variables.
Problem: Is Φ satisfiable?
Hint t: the satisfying assignment

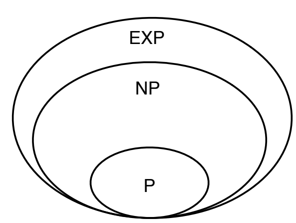
`Certify3SAT(` $\langle \Phi \rangle, t$`)`
- ▷ Check if t makes Φ true

Takeaway



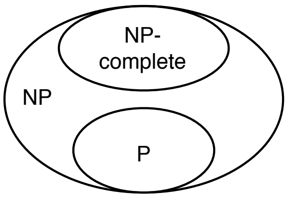
- 3SAT and INDEPENDENT SET are in NP, as are many other problems that are hard to solve, but easy to certify!

P, NP, EXP



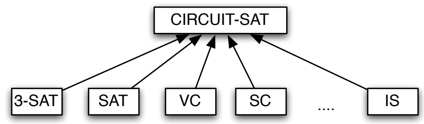
- **Claim**: P $\subseteq$ NP
- **Claim**: NP $\subseteq$ EXP
- Both straightforward to prove, but not critical right now.

NP-Complete



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

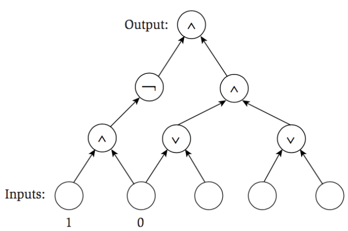
NP-Complete



- **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?

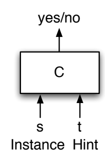


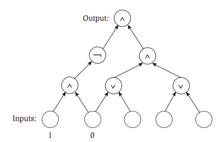
Satisfiable? Yes. Set inputs: 1, 1, 0.

CIRCUIT-SAT

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

Proof Idea: encode certifier as a circuit. If $X \in$ NP, then X has a poly-time certifier $C(s, t)$





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

Example

See Independent Set example in other slides.

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 $Y \leq_P X$.

Theorem: 3-SAT is NP-Complete.

- In NP? Yes, check satisfying assignment in poly-time.
- Can show that CIRCUIT-SAT $\leq_P$ 3-SAT

NP-Complete Problems