COMPSCI 311: Introduction to Algorithms

Lecture 21: Intractability: SAT, NP

Marius Minea

University of Massachusetts Amherst

slides credit: Dan Sheldon

10 April 2019

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 to General Case: Set Cover

Problem. Given a set U of n elements, subsets $S_1, \ldots, S_m \subset U$, and a number k, does there exist a collection of at most k subsets S_i whose union is U?

- Example: $U = \{A, B, C, D, E\}$ is the set of all skills, there are five people with skill sets:

$$S_1 = \{A, C\}, \quad S_2 = \{B, E\}, \quad S_3 = \{A, C, E\}$$

$$S_4 = \{D\}, \quad S_5 = \{B, C, E\}$$

- Find a small team that has all skills. S_1, S_4, S_5

Theorem. VertexCover $\leq_P$ SetCover

Clicker Question

Vertex Cover is a special case of Set Cover with:

A. $U = V$ and $S_e =$ the two endpoints of e for each $e \in E$.

B. $U = E$ and $S_v =$ the set of edges incident to v for each $v \in V$.

C. $U = V \cup E$ and $S_v =$ the set of neighbors of v together with edges incident to v for each $v \in V$.

Reduction of Vertex Cover to Set Cover

Reduction.

- Given Vertex Cover instance $\langle G, k \rangle$
- Construct Set Cover instance $\langle U, S_1, \ldots, S_m, k \rangle$ with $U = E$, and $S_v =$ the set of edges incident to v
- Return Yes iff `solveSC`$(\langle U, S_1, \ldots, S_m, k \rangle) =$ Yes

Proof

- Straightforward to see that $S_{v_1}, \ldots, S_{v_\ell}$ is a set cover of size ℓ if and only if $v_1, \ldots, v_\ell$ is a vertex cover of size ℓ
- This implies the algorithm correctly outputs: Yes if G has a vertex cover of size $\leq k$ and No otherwise
- Polynomial # of steps outside of `solveSC`
- Only one call to `solveSC`

Reduction Strategies

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

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	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	$C_1 \wedge C_2 \wedge \ldots \wedge C_m$	"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"

Solving 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?

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

Some simplification rules:

If x_i is unit clause: $x_i \wedge (\ldots) \wedge (\ldots)$, x_i must be true
If $\bar{x_j}$ is unit clause, x_j must be false

Propagation: if x_i is true, all clauses with x_i are true and $\bar{x_i}$ can be removed from all clauses

But, if no more simplifications, must still try both cases for x_i
worst-case asymptotics still exponential (brute force is 2^n)

SETH: Strong Exponential Time Hypothesis (more than P $\neq$ NP): SAT cannot be solved in subexponential time in the worst case.

Clicker Question: Variations of SAT

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

2-SAT — each C_i has exactly two terms

What is the strongest statement below that follows trivially from the definitions above?

A. 2-SAT $\leq_P$ 3-SAT $\leq_P$ SAT

B. 2-SAT $\leq_P$ SAT and 3-SAT $\leq_P$ SAT

C. SAT $\leq_P$ 3-SAT $\leq_P$ 2-SAT

Reduction by Gadgets: Satisfiability

Claim: 3-SAT $\leq_P$ IndependentSet.

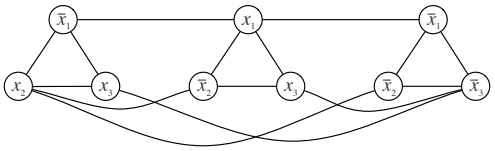
Reduction:

- 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 Yes if `solveIS`$(\langle G, m \rangle)$ = Yes

Reduction

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

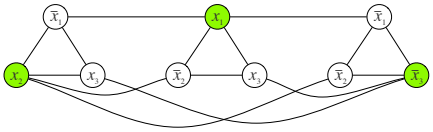
$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$



- 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

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

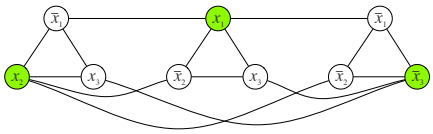


Claim: if G has an independent set of size m, then $\langle C_1, \ldots, C_m \rangle$ is satisfiable

- 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

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

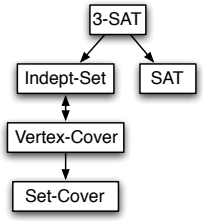


Claim: if $\langle C_1, \ldots, C_m \rangle$ is satisfiable, then G has an independent set of size m

- 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

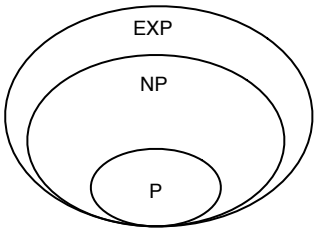
Partial map of problems we can use to solve others in polynomial time, through transitivity of reductions:



- $Y \longrightarrow X$ means $Y \leq_P X$.

Toward a Definition of NP

Remember our problem hierarchy:



Let's formally define NP.

Remember: exponential time means $O(2^{n^d})$ for some constant d.

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

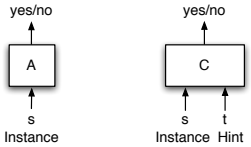
Problem classes

- 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. Must also be polynomial-size 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

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

```
CertifyIS( ⟨G, k⟩, t)
  if |t| < k return NO
  for each edge e = (u, v) ∈ E do
    if u ∈ t and v ∈ t return NO
  end for
  Return YES
```

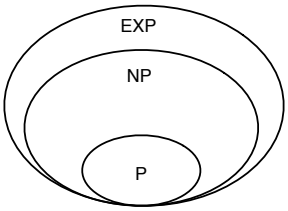
Polynomial time? Yes, linear in $|E|$.

Example: 3-SAT

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

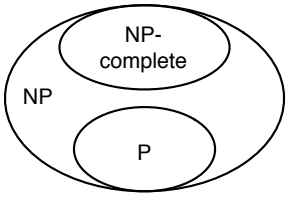
`Certify3SAT`($\langle\Phi\rangle, t$)
▷ Check if t makes Φ true

P, NP, EXP



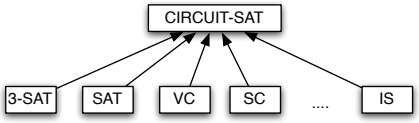
- 3SAT and Independent Set are in NP, as are many other problems that are hard to solve, but easy to certify!
- **Claim**: P $\subseteq$ NP
- **Claim**: NP $\subseteq$ EXP
- Both straightforward to prove, but not critical right now.

NP-Complete



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

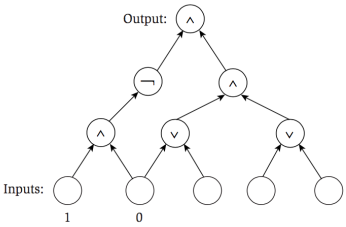
NP-Complete



- **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?



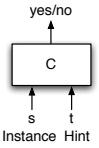
Satisfiable? Yes. Set inputs: 1, 1, 0.

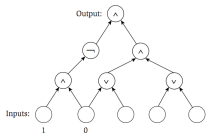
Circuit-SAT

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 $C(s,t)$

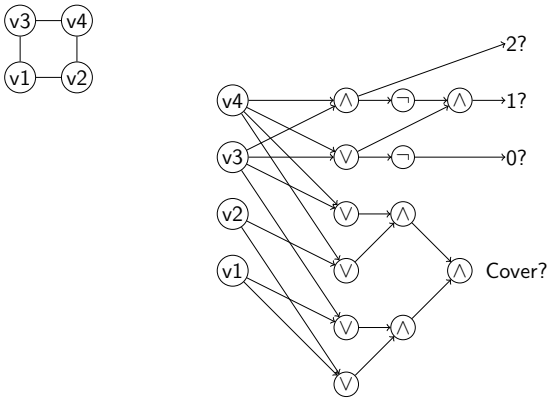




- Construct a circuit where s is hard-coded, and circuit is satisfiable iff $\exists t$ that causes $C(s,t)$ to output Yes
- s is Yes instance $\Leftrightarrow \exists\ t$ such that $C(s,t)$ outputs Yes
- s is Yes instance $\Leftrightarrow$ circuit is satisfiable
- Algorithm for Circuit-Sat implies an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

A Circuit-SAT reduction

- Vertex Cover – Does G have VC of size at most k?



Clicker Question

It's easy to give a reduction from 3-SAT to Circuit-SAT, i.e., to show that 3-SAT $\leq_P$ Circuit-SAT.
What can we conclude from this?

A. 3-SAT is NP-complete.

B. 3-SAT is in NP.

C. If Circuit-SAT is NP-complete, then 3-SAT is also NP-complete.

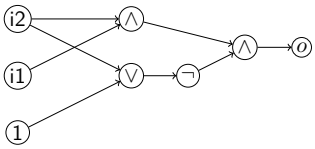
Back to 3-SAT

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

Theorem: 3-SAT is NP-Complete.

- In NP? Yes, check satisfying assignment in poly-time.
- Prove by reduction from Circuit-SAT.

Example.



Reduction: Circuit-SAT $\leq_P$ 3-SAT

- One variable x_v per circuit node v plus clauses to enforce circuit computations
- Equality = equivalence (conjunction of two implications)
- Write implication $A \Rightarrow B$ as clause $\neg A \vee B$
- Negation node: $x_v = \neg x_u$
  - $x_u \Rightarrow \neg x_v$
  - $\neg x_u \Rightarrow x_v$
- OR node: $x_v = x_u \vee x_w$
  - $x_u \Rightarrow x_v$
  - $x_w \Rightarrow x_v$
  - $x_v \Rightarrow x_u \vee x_w$
- AND node: $x_v = x_u \wedge x_w$
  - $x_v \Rightarrow x_u$
  - $x_v \Rightarrow x_w$
  - $\neg x_v \Rightarrow \neg x_u \vee \neg x_w$

Reduction: Circuit-SAT $\leq_P$ 3-SAT

- Clause $C = x_v$ for input bits v fixed to one
- Clause $C = \neg x_v$ for input bits v fixed to zero
- Clause $C = x_o$ for output bit
- This formula is satisfiable iff circuit is satisfiable.
- Deal with clauses of size 1 and 2 by introducing two new variables and clauses that force them to be equal to zero.

Proving New Problems NP-Complete

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