CS 580: Algorithm Design and Analysis

Jeremiah Blocki
Purdue University
Spring 2018

8.2 Reductions via "Gadgets"

Basic reduction strategies.

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction via "gadgets."

Recap

- Polynomial Time Reductions ($X \leq_P Y$)
  - **View 1:** A polynomial time algorithm for Y yields a polynomial time algorithm for X.
  - **View 2:** If there is no polynomial time algorithm to solve problem X then there is no polynomial time algorithm to solve problem Y
- Key Problems
  - Independent Set
  - Vertex Cover
  - Set Cover
  - 3-SAT
- Example Reductions
  - Independent Set $\leq_P$ Vertex Cover (Simple Equivalence)
  - Vertex Cover $\leq_P$ Independent Set (Simple Equivalence)
  - Independent Set $\leq_P$ Set Cover (Special Case to General)

2

---

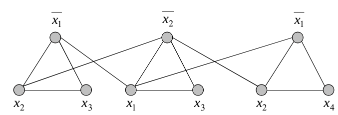
3 Satisfiability Reduces to Independent Set

Claim. 3-SAT $\leq_P$ INDEPENDENT-SET.
Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Construction.

- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

G



k = 3

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

5

Recap: 3-SAT

Literal: A Boolean variable or its negation.

Clause: A disjunction of literals.

Conjunctive normal form: A propositional formula Φ that is the conjunction of clauses.

SAT: Given CNF formula Φ, does it have a satisfying truth assignment?

3-SAT: SAT where each clause contains (*at most*) 3 literals.

Ex: $(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$
Yes: x_1 = true, x_2 = true x_3 = false.

3

---

3 Satisfiability Reduces to Independent Set

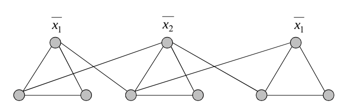
Claim. G contains independent set of size k = $|\Phi|$ iff Φ is satisfiable.

Pf. $\Rightarrow$ Let S be independent set of size k.

- S must contain exactly one vertex in each triangle.
- Set these literals to true. ← and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Pf $\Leftarrow$ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size k. ▪

G



k = 3

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

6

Review

Basic reduction strategies.
- Simple equivalence: INDEPENDENT-SET $\equiv_P$ VERTEX-COVER.
- Special case to general case: VERTEX-COVER $\leq_P$ SET-COVER.
- Encoding with gadgets: 3-SAT $\leq_P$ INDEPENDENT-SET.

Transitivity. If $X \leq_P Y$ and $Y \leq_P Z$, then $X \leq_P Z$.
Pf idea. Compose the two algorithms.

Ex: 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

7

Self-Reducibility

Decision problem. Does there exist a vertex cover of size $\leq k$?
Search problem. Find vertex cover of minimum cardinality.

Self-reducibility. Search problem $\leq_P$ decision version.
- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.
- (Binary) search for cardinality k^* of min vertex cover.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
  - any vertex in any min vertex cover will have this property
- Include v in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.

delete v and all incident edges

8

8.3 Definition of NP

Decision Problems

Decision problem.
- X is a set of strings.
- Instance: string s.
- Algorithm A solves problem X: A(s) = yes iff $s \in X$.

Polynomial time. Algorithm A runs in poly-time if for every string s, A(s) terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial.

length of s

PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }
Algorithm. [Agrawal-Kayal-Saxena, 2002] $p(|s|) = |s|^8$.

10

Definition of P

P. Decision problems for which there is a poly-time algorithm.

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	51, 17	51, 16
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Dynamic programming	niether neither	acgggt ttttta
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

11

NP

Certification algorithm intuition.
- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. Algorithm C(s, t) is a certifier for problem X if for every string s, $s \in X$ iff there exists a string t such that C(s, t) = yes.

"certificate" or "witness"

NP. Decision problems for which there exists a poly-time certifier.

C(s, t) is a poly-time algorithm and $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

Remark. NP stands for nondeterministic polynomial-time.

12

Certifiers and Certificates: Composite

COMPOSITES. Given an integer s, is s composite?

Certificate. A nontrivial factor t of s. Note that such a certificate exists iff s is composite. Moreover $|t| \le |s|$.

Certifier.

```
boolean C(s, t) {
   if (t ≤ 1 or t ≥ s)
      return false
   else if (s is a multiple of t)
      return true
   else
      return false
}
```

Instance. s = 437,669.
Certificate. t = 541 or 809. ⟵ $437{,}669 = 541 \times 809$

Conclusion. COMPOSITES is in NP.

13

Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ, is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

Ex.

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

instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

certificate t

Conclusion. SAT is in NP.

14

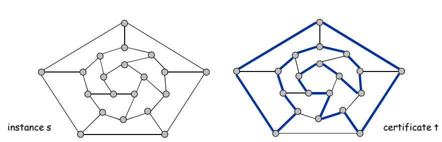
Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Conclusion. HAM-CYCLE is in NP.



15

P, NP, EXP

P. Decision problems for which there is a poly-time algorithm.
EXP. Decision problems for which there is an exponential-time algorithm.
NP. Decision problems for which there is a poly-time certifier.

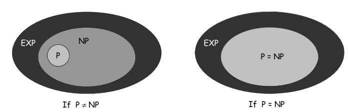
Claim. $P \subseteq NP$.
Pf. Consider any problem X in P.
- By definition, there exists a poly-time algorithm A(s) that solves X.
- Certificate: $t = \varepsilon$, certifier C(s, t) = A(s). ▪

Claim. $NP \subseteq EXP$.
Pf. Consider any problem X in NP.
- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with $|t| \le p(|s|)$.
- Return yes, if C(s, t) returns yes for any of these. ▪

16

The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.



would break RSA cryptography
(and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …
If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

Consensus opinion on P = NP? Probably no.

17

The Simpson's: P = NP?

Futurama: P = NP?

Polynomial Transformation

Def. Problem X polynomial reduces (Cook) to problem Y if arbitrary instances of problem X can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Def. Problem X polynomial transforms (Karp) to problem Y if given any input x to X, we can construct an input y such that x is a yes instance of X iff y is a yes instance of Y.

we require |y| to be of size polynomial in |x|

Note. Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form.

Open question. Are these two concepts the same with respect to NP?

we abuse notation $\le_p$ and blur distinction

22

Looking for a Job?

Some writers for the Simpsons and Futurama.
- J. Steward Burns. M.S. in mathematics, Berkeley, 1993.
- David X. Cohen. M.S. in computer science, Berkeley, 1992.
- Al Jean. B.S. in mathematics, Harvard, 1981.
- Ken Keeler. Ph.D. in applied mathematics, Harvard, 1990.
- Jeff Westbrook. Ph.D. in computer science, Princeton, 1989.

20

NP-Complete

NP-complete. A problem Y in NP with the property that for every problem X in NP, $X \le_p Y$.

Theorem. Suppose Y is an NP-complete problem. Then Y is solvable in poly-time iff P = NP.
Pf. $\Leftarrow$ If P = NP then Y can be solved in poly-time since Y is in NP.
Pf. $\Rightarrow$ Suppose Y can be solved in poly-time.
- Let X be any problem in NP. Since $X \le_p Y$, we can solve X in poly-time. This implies NP $\subseteq$ P.
- We already know P $\subseteq$ NP. Thus P = NP. ▪

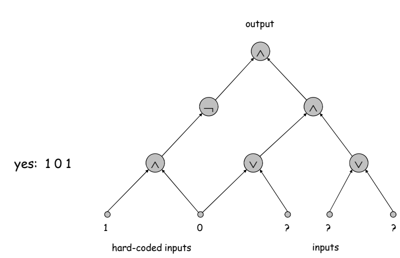
Fundamental question. Do there exist "natural" NP-complete problems?

23

8.4 NP-Completeness

Circuit Satisfiability

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?



24

The "First" NP-Complete Problem

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]
Pf. (sketch)

- Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

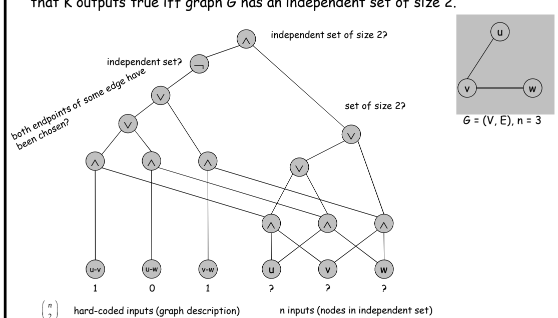
  sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- Consider some problem X in NP. It has a poly-time certifier *C(s, t)*. To determine whether s is in X, need to know if there exists a certificate t of length p(|s|) such that *C(s, t)* = yes.
- View *C(s, t)* as an algorithm on |s| + p(|s|) bits (input s, certificate t) and convert it into a poly-size circuit K.
  - first |s| bits are hard-coded with s
  - remaining p(|s|) bits represent bits of t
- Circuit K is satisfiable iff *C(s, t)* = yes.

25

Example

Ex. Construction below creates a circuit K whose inputs can be set so that K outputs true iff graph G has an independent set of size 2.



26

Establishing NP-Completeness

Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y.

- Step 1. Show that Y is in NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_P Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \underbrace{\leq_P}_{\text{by definition of NP-complete}} X \underbrace{\leq_P}_{\text{by assumption}} Y$.

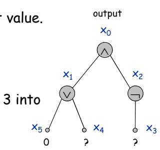
- By transitivity, $W \leq_P Y$.
- Hence Y is NP-complete. ▪

27

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.
Pf. Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

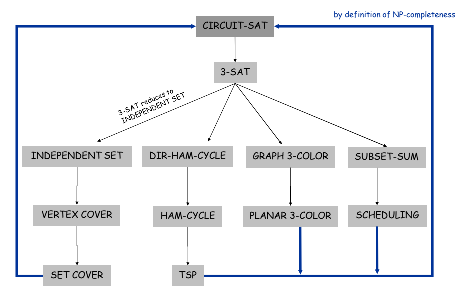
- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3$ ⇒ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5$ ⇒ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2$ ⇒ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$
- Hard-coded input values and output value.
  - $x_5 = 0$ ⇒ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ ⇒ add 1 clause: x_0
- Final step: turn clauses of length < 3 into clauses of length exactly 3. ▪



28

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!



29

Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.

30

Extent and Impact of NP-Completeness

Extent of NP-completeness. [Papadimitriou 1995]
- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (title, abstract, keywords).
  - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.
- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
- 2000: Istrail proves 3D problem NP-complete.

More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.
Biology: protein folding.
Chemical engineering: heat exchanger network synthesis.
Civil engineering: equilibrium of urban traffic flow.
Economics: computation of arbitrage in financial markets with friction.
Electrical engineering: VLSI layout.
Environmental engineering: optimal placement of contaminant sensors.
Financial engineering: find minimum risk portfolio of given return.
Game theory: find Nash equilibrium that maximizes social welfare.
Genomics: phylogeny reconstruction.
Mechanical engineering: structure of turbulence in sheared flows.
Medicine: reconstructing 3-D shape from biplane angiocardiogram.
Operations research: optimal resource allocation.
Physics: partition function of 3-D Ising model in statistical mechanics.
Politics: Shapley-Shubik voting power.
Pop culture: Minesweeper consistency.
Statistics: optimal experimental design.

8.9 co-NP and the Asymmetry of NP

Asymmetry of NP

Asymmetry of NP. We only need to have short proofs of yes instances.

Ex 1. SAT vs. TAUTOLOGY.
- Can prove a CNF formula is satisfiable by giving such an assignment.
- How could we prove that a formula is not satisfiable?

Ex 2. HAM-CYCLE vs. NO-HAM-CYCLE.
- Can prove a graph is Hamiltonian by giving such a Hamiltonian cycle.
- How could we prove that a graph is not Hamiltonian?

Remark. SAT is NP-complete and SAT $\equiv_P$ TAUTOLOGY, but how do we classify TAUTOLOGY?

↑
not even known to be in NP

NP and co-NP

NP. Decision problems for which there is a poly-time certifier.
Ex. SAT, HAM-CYCLE, COMPOSITES.

Def. Given a decision problem X, its complement $\overline{X}$ is the same problem with the yes and no answers reverse.

Ex. $\overline{X}$ = { 0, 1, 4, 6, 8, 9, 10, 12, 14, 15, … }
X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, … }

co-NP. Complements of decision problems in NP.
Ex. TAUTOLOGY, NO-HAM-CYCLE, PRIMES.

NP = co-NP ?

Fundamental question. Does NP = co-NP?
- Do yes instances have succinct certificates iff no instances do?
- Consensus opinion: no.

Theorem. If NP ≠ co-NP, then P ≠ NP.
Pf idea.
- P is closed under complementation.
- If P = NP, then NP is closed under complementation.
- In other words, NP = co-NP.
- This is the contrapositive of the theorem.

Good Characterizations

Good characterization. [Edmonds 1965] NP $\cap$ co-NP.
- If problem X is in both NP and co-NP, then:
  - for yes instance, there is a succinct certificate
  - for no instance, there is a succinct disqualifier
- Provides conceptual leverage for reasoning about a problem.

Ex. Given a bipartite graph, is there a perfect matching.
- If yes, can exhibit a perfect matching.
- If no, can exhibit a set of nodes S such that $|N(S)| < |S|$.

37

FACTOR is in NP $\cap$ co-NP

FACTORIZE. Given an integer x, find its prime factorization.
FACTOR. Given two integers x and y, does x have a nontrivial factor less than y?

Theorem. FACTOR $\equiv_P$ FACTORIZE.

Theorem. FACTOR is in NP $\cap$ co-NP.
Pf.
- Certificate: a factor p of x that is less than y.
- Disqualifier: the prime factorization of x (where each prime factor is less than y), along with a certificate that each factor is prime.

40

Good Characterizations

Observation. P $\subseteq$ NP $\cap$ co-NP.
- Proof of max-flow min-cut theorem led to stronger result that max-flow and min-cut are in P.
- Sometimes finding a good characterization seems easier than finding an efficient algorithm.

Fundamental open question. Does P = NP $\cap$ co-NP?
- Mixed opinions.
- Many examples where problem found to have a non-trivial good characterization, but only years later discovered to be in P.
  - linear programming [Khachiyan, 1979]
  - primality testing [Agrawal-Kayal-Saxena, 2002]

Fact. Factoring is in NP $\cap$ co-NP, but not known to be in P.

if poly-time algorithm for factoring, can break RSA cryptosystem

38

Primality Testing and Factoring

We established: PRIMES $\le_P$ COMPOSITES $\le_P$ FACTOR.

Natural question: Does FACTOR $\le_P$ PRIMES ?
Consensus opinion. No.

State-of-the-art.
- PRIMES is in P. ← proved in 2001
- FACTOR not believed to be in P.

RSA cryptosystem.
- Based on dichotomy between complexity of two problems.
- To use RSA, must generate large primes efficiently.
- To break RSA, suffixes to find efficient factoring algorithm.

41

PRIMES is in NP $\cap$ co-NP

Theorem. PRIMES is in NP $\cap$ co-NP.
Pf. We already know that PRIMES is in co-NP, so it suffices to prove that PRIMES is in NP.

Pratt's Theorem. An odd integer s is prime iff there exists an integer $1 < t < s$ s.t.

$$\begin{aligned} t^{s-1} &\equiv 1 \pmod{s} \\ t^{(s-1)/p} &\not\equiv 1 \pmod{s} \end{aligned}$$

for all prime divisors p of s-1

Input. s = 437,677
Certificate. t = 17, $2^2 \times 3 \times 36{,}473$

prime factorization of s-1
also need a recursive certificate to assert that 3 and 36,473 are prime

Certifier.
- Check s-1 = $2 \times 2 \times 3 \times 36{,}473$.
- Check $17^{s-1} = 1$ (mod s).
- Check $17^{(s-1)/2} \equiv 437{,}676$ (mod s).
- Check $17^{(s-1)/3} \equiv 329{,}415$ (mod s).
- Check $17^{(s-1)/36{,}473} \equiv 305{,}452$ (mod s).

use repeated squaring

39