CMPSCI 311: Introduction to Algorithms

Lecture 19: Reductions and Intractability

Akshay Krishnamurthy

University of Massachusetts

Last Compiled: April 18, 2018

Recap

- Reductions. $Y \leq_P X$ if can solve Y in poly-time with algorithm for X.
- New problems. IndependentSet, VertexCover, SetCover, SAT, 3-SAT.
- Results.

$$\text{3-SAT} \leq_P \text{IS} \leq_P \text{VC} \leq_P \text{SC}$$
$$\text{VC} \leq_P \text{IS}$$

Reduction #3: Satisfiability

- Let $X = \{x_1, \ldots, x_n\}$ be boolean variables
  - A term or literal is x_i or $\neg x_i$.
  - A clause is *or* of several terms $(t_1 \vee t_2 \vee \ldots \vee t_\ell)$.
  - A formula is *and* of several clauses
  - An assignment $\phi : X \to \{0, 1\}$ gives T/F to each variable.
- ϕ satisfies formula if all clauses evaluate to True.

Example.

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_4 \vee \neg x_3) \wedge (\neg x_1 \vee x_4) \wedge (x_3 \vee x_2)$$

Reduction #3: Satisfiability

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

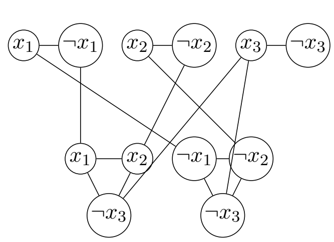
3-SAT – Given boolean formula $C_1 \wedge C_2 \ldots \wedge C_m$ over variables $X = \{x_1, \ldots, x_n\}$ where each C_i has three literals, does there exist a satisfying assignment?

Theorem. 3-SAT $\leq_P$ IndependentSet.

Reduction

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

- Associate nodes in graph with literals (≥ 2 per variable).
- Associate 3 nodes per clause in a *gadget*.
- If $\phi(x_i) = 1$ in assignment, then cannot select some nodes.

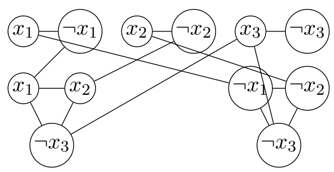


Formally

- Given $\{x_1, \ldots, x_n\}$ and clauses $C_1, \ldots, C_m$.
- Make graph with:
  - Vertices v_{i1}, v_{i0} and t_{j1}, t_{j2}, t_{j3} for $i \in [n], j \in [m]$.
  - Edges (v_{i1}, v_{i0}) for all i and $(t_{jk}, t_{jk'})$ for $k, k' \in [3]$.
  - If jth clause is $x_a \vee \neg x_b \vee x_c$, edges $(t_{j1}, v_{a0}), (t_{j2}, v_{b1}), (t_{j3}, v_{c0})$.
- If G has IS of size $n + m$, output True, else False.

Satisfiability Proof

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$



Claim. Reduction takes polynomial time.

Claim. Graph has IS of size $n + m$ if and only if formula satisfiable.

Satisfiability Proof

- If satisfiable, exists $\phi : X \to \{0, 1\}$ such that $C_j(\phi) = 1$ for all j.
- If $\phi(x_i) = 1$ select v_{i1} in IS, else select v_{i0}.
- For C_j there must be a term corresponding to true literal.
  - If term is x_i, it connects to v_{i0} but we know $\phi(x_i) = 1$, so v_{i0} is not selected and we can select this term without conflict.
- If graph has IS of size $n + m$,
  - At most one of v_{i0}, v_{i1} and at most one of t_{j1}, t_{j2}, t_{j3}.
  - If select v_{i0}, will never select term corresponding to x_i.
  - Hence cannot use x_i in one clause and $\neg x_i$ in another.

3-SAT Reduction

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

- For every 3-SAT formula, exists a graph G s.t. formula satisfiable if and only if G has IS of size $n + m$.
- Does not imply $\textsc{IndependentSet} \leq_P \text{3-SAT}$.
  - For this, need to prove: For every (G, k), exists formula that is satisfiable iff G has IS of size k.

A class of problems

- Decision vs certification.
  - Seems hard to find a large independent set.
    - Or check if one exists.
  - But *easy* to certify a proposed solution, by checking for adjacent vertices.
- Formal languages and decision problems.
  - Encode problem inputs as binary strings s.
  - A decision problem X is the set of binary strings that have TRUE answer.
  - Algorithm A solves problem X if $A(s) = \text{TRUE}$ iff $s \in X$.

Certification and NP.

- Algorithm A solves problem X if $A(s) = \text{TRUE}$ iff $s \in X$.
- Running time now measured in $|s|$, still want polytime.
- $\mathcal{P}$: problems that can be solved by a polytime algorithm.
- B is a polytime **certifier** for problem X if
  - B is a polytime algorithm of two inputs s, t.
  - $s \in X$ iff exists t with $|t| \leq \text{poly}(|s|)$ and $B(s, t) = \text{TRUE}$.
  - **Example.** Certifier for independent set.
- $\mathcal{NP}$: problems with polytime certifier.

P and NP

Claim. $\mathcal{P} \subset \mathcal{NP}$.

Proof.

- If $X \in \mathcal{P}$, exists algorithm A that solves X.
- Need to design certifier B.
  - Set $B(s, t) = A(s)$.
  - B runs in polynomial time
  - If $s \in X$, $B(s, t) = A(s) = \textsc{True}$ for all t.
  - If $s \notin X$, $B(s, t) = A(s) = \textsc{False}$ for all t.

Some NP problems.

- INDEPENDENTSET
- VERTEXCOVER
- SETCOVER
- Basically all problems we have seen so far!
- UNSATISFIABILITY – not in $\mathcal{NP}$.

Million dollar question

Question. Does $\mathcal{P} = \mathcal{NP}$?

Can make some progress by considering "hardest" $\mathcal{NP}$ problems.

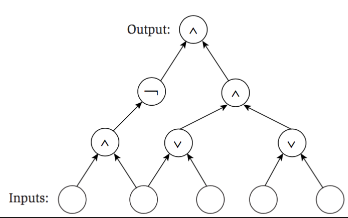
Definition. X is NP-Complete if $X \in \mathcal{NP}$ and for all $Y \in \mathcal{NP}$ $Y \leq_P X$.

- If X is NP-Complete then X has poly-time algorithm iff $\mathcal{P} = \mathcal{NP}$.

CIRCUIT-SAT

Problem. Given a boolean circuit with some inputs and single boolean output, are there inputs that produce 1 at the output?

- A circuit is a labeled DAG.
- Sources (no incoming edges) labeled with constant or with input variable name.
- Other nodes labeled with $\wedge$ (and), $\vee$ (or), $\neg$ (not).
- Single node with no outgoing edges computes the output bit.



CIRCUIT-SAT

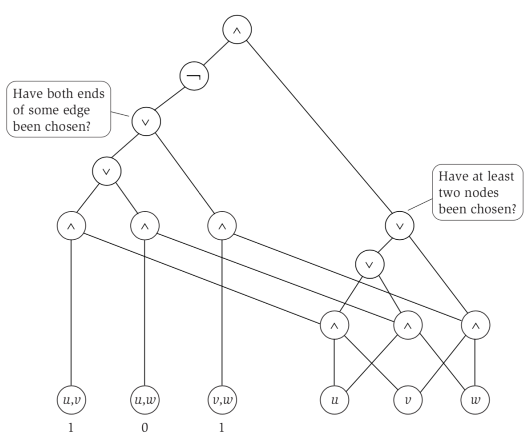
Theorem. CIRCUIT-SAT is NP-Complete.

Proof (Idea).

- A poly-time algorithm once input length is fixed can be executed on a poly-sized circuit.
- Not surprising since our hardware is circuits!
- Need to show that arbitrary $X \in \mathcal{NP}$ has $X \leq_P$ CIRCUIT-SAT.
- All we know about X is its efficient certifier $B(\cdot, \cdot)$.
- Encode $B(s, \cdot)$ as a circuit with $\text{poly}(|s|)$ inputs.
    - Satisfiable iff exists t with $|t| \leq \text{poly}(|s|)$ s.t. $B(s, t) = \text{TRUE}$ iff $s \in X$.

A CIRCUIT-SAT reduction

Independent set on 3 nodes clique:



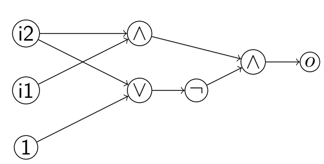
Back to 3-SAT

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

Theorem. 3-SAT is NP-Complete.

- Clearly in $\mathcal{NP}$.
- Prove by reduction from CIRCUITSAT.

Example.



The Reduction

- One variable x_v per circuit node v.
- Clauses to enforce circuit computations.
  - If v is $\neg$ then v has one input u and can add clauses $(x_v \vee x_u), (\neg x_v \vee \neg x_u)$.
  - If v is $\vee$ with u, w incoming then $(x_v \vee \neg x_u), (x_v \vee \neg x_w), (\neg x_v \vee x_u \vee x_w)$.
  - If v is $\wedge$ then $(\neg x_v \vee x_u), (\neg x_v \vee x_w), (x_v \vee \neg x_u \vee \neg x_w)$.
- Input bits get set with (x_v) if fixed to one and $(\neg x_v)$ otherwise.
- Clause (x_o) for output bit.

Final steps

- This formula satisfiable iff circuit is satisfiable.
- But not a 3-sat formula! It has clauses of size 1 and 2.
  - Fix: 4 new variables $z_1, \ldots, z_4$ where z_1, z_2 forced to be 0.
  - Include those two in any short clause.

Theorem. IndependentSet, VertexCover, SetCover, SAT, 3-SAT are all NP-Complete.