Lecture 20 Intractability II: Reductions	L 20.1 6.046 05/04/2017

Today

- Review: NP-completeness
- Reductions: cSAT → SAT → 3-SAT → Vertex Cover

NP-completeness

NP: contains all decision problems ("yes"/"no" answers) whose "yes" inputs have polynomially short and polynomial-time verifiable certificates.

examples:
- Is there a 2d-matching in a graph?
- Is there a 3d-matching in a graph?
- Is there a path from s to t of weight $\leq k$?
- Is there a path from s to t of weight $\geq k$?

Reductions: We say that $\Pi_1 \leq_p \Pi_2$ iff there is a polynomial-time algorithm R such that
- If x is an input to Π_1, then R(x) is an input to Π_2
- $\Pi_1(x) = \text{"yes"} \iff \Pi_2(R(x)) = \text{"yes"}$

In words, Π_2 is at least as hard as Π_1

NP-complete: "The hardest problems in NP"

Π is NP-complete iff (i) $\Pi \in NP$
(ii) $\forall \Pi' \in NP : \Pi' \leq_p \Pi$

This condition corresponds to NP-hard

Cook's Theorem [1971]: Circuit-SAT is NP-complete.

CSAT: Given Boolean circuit of AND, OR, NOT gates and no feedback, is there a set of {0,1} input values to produce output of 1?

Karp [1972]: Showed many other problems are NP-complete by reducing CSAT to them.

New problem: SAT Given a boolean formula, is it satisfiable (i.e., is there a set of inputs for which output is 1).

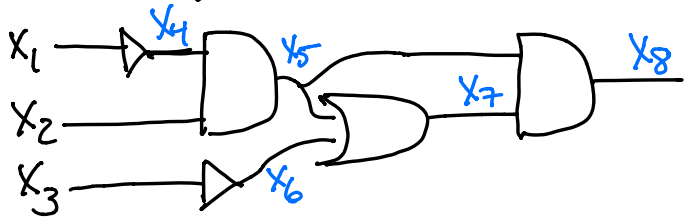
$\phi = (x_1 \vee \bar{x}_2) \wedge x_3 \wedge (\bar{x}_3 \vee x_1 \vee x_2)$ is satisfiable with $x_1 = 1$, $x_2 = 0$ or 1, $x_3 = 1$

Formulas consist of
- n boolean variables $x_1, x_2, \ldots, x_n$
- m boolean connectives $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT, also represented as $\bar{x}_i$), $\Rightarrow$ (implies), $\Leftrightarrow$ (iff)
- parentheses

We wish to show that SAT is NP-complete

- Clearly SAT is in NP ← to try all inputs is $\Omega(2^n)$.
  - Certificate of input formula plus proposed satisfying input assignment is polynomial in length and the verification algorithm would replace each variable with input value and then evaluate formula expression, which can be done in polynomial time.
- To show SAT is NP-hard, do we need to show it is at least as hard as every problem in NP? No, because we already know that cSAT is at least as hard as every problem in NP. So we just need to show SAT is at least as hard as cSAT. Then by transitivity SAT will be at least as hard as all problems in NP. Thus, we will just need to reduce cSAT to SAT.
- We might imagine that the reduction might simply convert the circuit into a formula. However, the possibility of large fan-out available in circuitry that doesn't exist for formulas causes this approach to not be polynomial in time.

- Instead, we do the following reduction, shown by example



$$\phi = x_8 \wedge (x_8 \iff (x_5 \wedge x_7)) \wedge (x_7 \iff (x_5 \vee x_6)) \wedge (x_6 \iff \neg x_3) \wedge (x_5 \iff (x_4 \wedge x_2)) \wedge (x_4 \iff \neg x_1)$$

For formula to be satisfied, need output to be 1

and each gate to correctly evaluate its inputs.

- Given any circuit, it is simple to reduce it to the appropriate formula in polytime.
- Circuit is satisfiable $\Rightarrow$ ϕ is satisfiable

  When a "yes" instance of circuit inputs is applied to the circuit, every wire has a well defined value and the output is 1. Applying those wire values in the formula ϕ causes every clause to evaluate to 1 and thus ϕ to evaluate to 1.
- ϕ is satisfiable $\Rightarrow$ Circuit is satisfiable

  Likewise, a set of inputs that satisfy ϕ, when applied to the corresponding circuit, will cause each gate to evaluate to the intermediate "wire" values in the formula and thus produce an output of 1. $\therefore$ CSAT $\leq_p$ SAT & SAT is NP-complete

New problem: 3-SAT Given a formula ϕ in CNF with 3 literals per clause, is ϕ satisfiable.

Conjunctive Normal Form (AND of ORs)

Examples:

① $\phi = (x_1 \vee \bar{x}_2) \wedge x_3 \wedge (\bar{x}_3 \vee x_1 \vee x_2)$ is not a valid input to 3-SAT because the first two clauses do not have three literals each.

variables: x_1, x_2, x_3
literals: $x_1, \bar{x}_2, x_2, x_3, \bar{x}_3$
clauses: $x_1 \vee \bar{x}_2$, x_3, $\bar{x}_3 \vee x_1 \vee x_2$

② $\phi' = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4)$ is a valid input to 3-SAT.

- Clearly 3-SAT $\in$ NP
- It is easy to see that 3-SAT $\leq_p$ SAT
  - because 3-SAT is a special case of SAT
  - but also because SAT is NP complete
- Karp showed SAT $\leq_p$ 3-SAT

$\Rightarrow$ 3-SAT is NP-complete

Summary so far: CSAT $\leq_p$ SAT $\leq$ 3-SAT

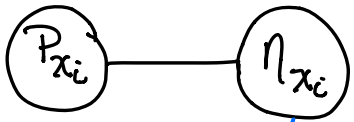
All of these are circuit or formula satisfiability problems. Can we treat a graph problem?

3-SAT $\leq_p$ Vertex Cover

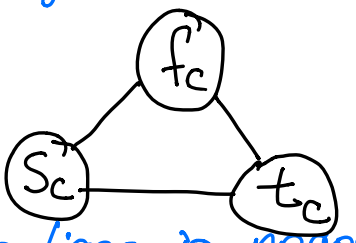
Vertex Cover (VC): Given a graph $G=(V,E)$ and an integer k, does there exist a subset $S \subseteq V$ s.t. $|S| \leq k$ and every edge $e \in E$ is incident to at least one vertex in S?

- Clearly, VC $\in$ NP: If (G,k) is a "yes" instance, then a poly-short certificate is a subset S of $\leq k$ vertices that in poly-time can be checked for incidence to all edges.

- Reduction from 3-SAT to VC:
  - Input to reduction: CNF formula ϕ
    n = # variables
    m = # clauses
  - Goal of reduction: encode ϕ via a graph $G=(V,E)$ and the number k such that
    (ϕ satisfiable) $\iff$ ($\exists$ a VC of size $\leq k$)
  - Technique: Gadget construction
    (i) for each variable, assign a gadget (i.e., a subgraph of G) representing its truth value.
    (ii) for each clause, assign a gadget representing that at least one literal must be true.
    (iii) assign edges connecting these kinds of gadgets.

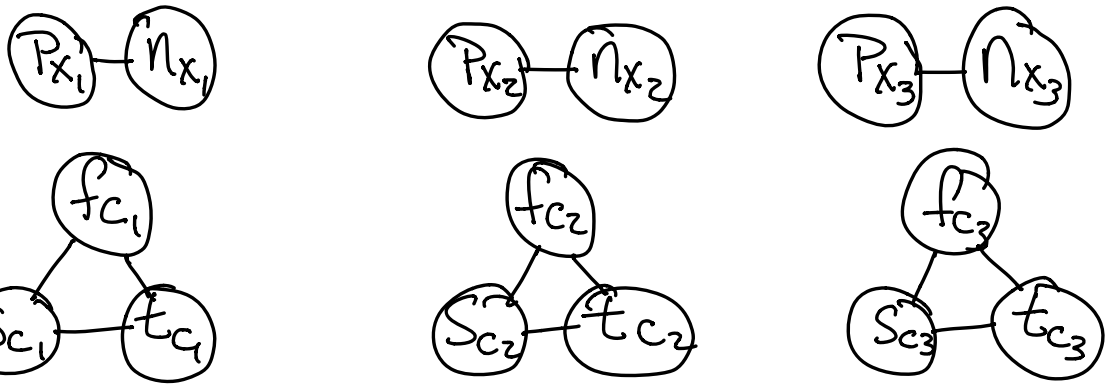
- Construction
  - each variable $x_i \longmapsto$ 

(*) choosing one vertex is necessary and sufficient to cover the edge

  - every clause $c \longmapsto$ 

(**) choosing two vertices is necessary and sufficient to cover these three edges

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



  - with no additional edges, the smallest vertex cover in our graph has size $k = n + 2m$

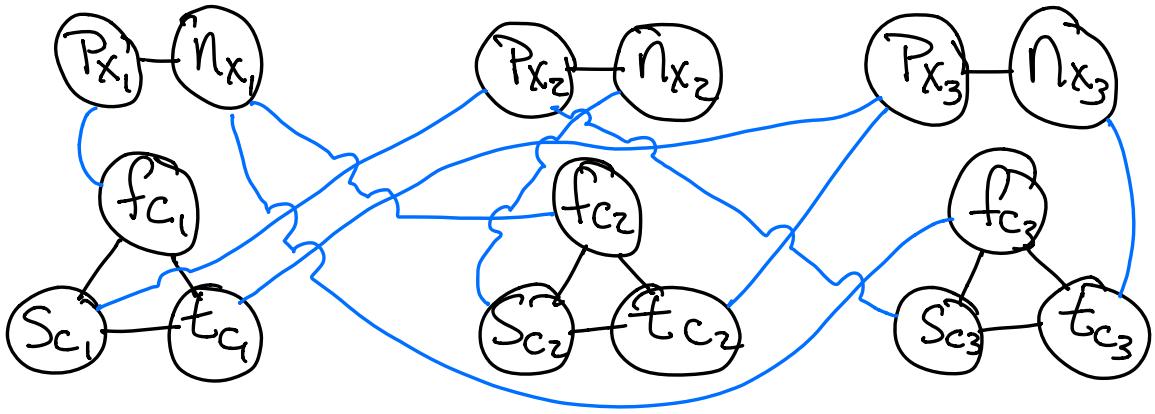
(***) [from (*) and (**)]

Trouble: This is true regardless of whether ϕ is satisfiable.

↳ we need to encode relationships between variables and clauses.

■ Connections between variable and clause gadgets

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



- formally, if first literal of clause c_i is x_j, then add edge (f_{c_i}, p_{x_j}). (positive)
- and if first literal of clause c_i is $\bar{x}_j$, then add edge (f_{c_i}, n_{x_j}). (negative)
- similarly for s_{c_i} (second) and t_{c_i} (third).

If a clause ("triangle") is satisfied, then at least one of its blue incident edges is covered. The remaining 2 blue edges are covered by picking 2 nodes from the triangle. Covering a blue edge from a variable node $\Rightarrow$ satisfying corresponding literal in clause.

Correctness:

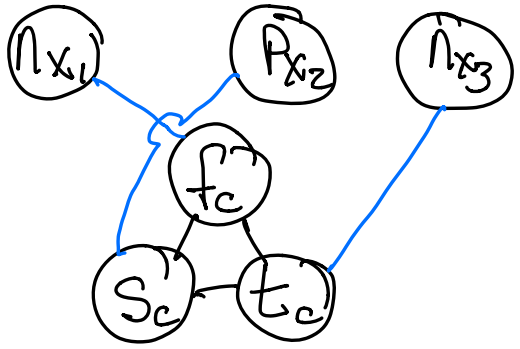
Claim 1: $\exists$ Vertex Cover of size $k = 2m + n \Rightarrow \phi$ is satisfiable

Proof: . If there exists VC of size k, it must use exactly one vertex from each variable gadget and exactly two vertices from each clause gadget.

- If P_{x_i} is in VC, then set $x_i = 1$. Otherwise set $x_i = 0$.
- Want to show that the resulting assignment $x_1, \ldots, x_n$ satisfies every clause.

- Pick any clause c of ϕ (e.g., $\bar{x}_1 \vee x_2 \vee \bar{x}_3$)

subclaim: If none of n_{x_1}, p_{x_2}, and n_{x_3} is in VC, then at least one edge adjacent to clause gadget is uncovered.



proof: Suppose not. Then exactly two of f_c, s_c, t_c are in VC (by ***). Then the blue edge adjacent to the vertex not in VC will be uncovered

Thus, at least one of n_{x_1}, p_{x_2}, n_{x_3} is in VC
$\Rightarrow$ assignment satisfies clause.

Claim 2: ϕ is satisfiable $\Rightarrow \exists$ VC of size k

Proof: If $\vec{x}$ is a satisfying assignment and $x_i = 1$, then include P_{x_i} in VC. Otherwise include n_{x_i}. Next pick another 2 vertices from each of the m clause gadgets to cover all edges.

This completes the proof of $3\text{-SAT} \leq_p \text{VC}$

$\Rightarrow$ VC is NP-complete

So far: $\text{cSAT} \leq_p \text{SAT} \leq_p 3\text{-SAT} \leq_p \text{VC} \leq_p \text{cSAT}$

from Cook [1971]

$\Rightarrow$ All of these NP-complete problems are equally hard.

$\Rightarrow$ If we had a solution to any one of them, by reduction we would have a solution to all of them.

Chain of reductions

cSAT → SAT → 3-SAT → VC → Hamiltonian Cycle → TSP

3-SAT → Clique

3-SAT → 3d-matching → subset sum → Partition → Rectangle Packing

Beyond NP-completeness

- approximation algorithms (next lecture)
- intelligent exponential search
- average case analysis (input may not be worst case)
- special input cases
  - vertex cover in bipartite vs. general graphs