COMPSCI 311: Introduction to Algorithms

Lecture 22: Reductions and NP-Complete Problems

Marius Minea

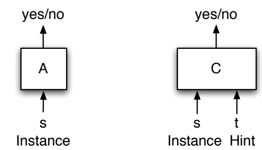
University of Massachusetts Amherst

slides credit: Dan Sheldon

17 April 2019

Review

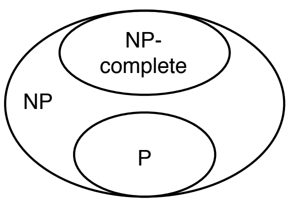
- P – class of problems with polytime **algorithm.**
- NP – class of problems with polytime **certifier**.



Example

Problem (X)	INDEPENDENT-SET
Instance (s)	Graph G and number k
Algorithm (A)	Try all subsets and check (but not poly-time)
Hint (t)	Which nodes are in the answer?
Certifier (C)	Are those nodes independent and size k?

NP-Complete



(if P $\neq$ NP)

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

To prove a new problem Q is NP-complete

- Check $Q \in$ NP.
- Choose an NP-complete problem Y (any $X \in$ NP reduces to it)
- Prove $Y \leq_P Q$ (then any $X \leq_P Y \leq_P Q$)

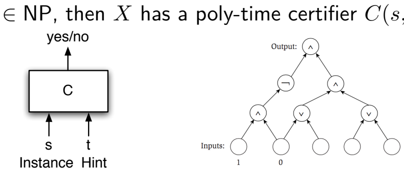
Need one NP-complete problem to bootstrap this.

CIRCUIT-SAT

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

Proof Idea: encode arbitrary certifier $C(s,t)$ as a circuit (polynomial-time algorithm $\implies$ polynomial-size 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

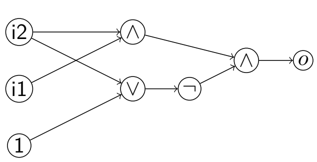
From CIRCUIT-SAT to 3-SAT

Fact: 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.

1. In NP? Yes, check satisfying assignment in poly-time.
2. 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: $x_{out} = \neg x_{in}$
  - $x_{in} \Rightarrow \neg x_{out}$
  - $\neg x_{in} \Rightarrow x_{out}$
- OR node: $x_{out} = x_1 \vee x_2$
  - $x_1 \Rightarrow x_{out}$
  - $x_2 \Rightarrow x_{out}$
  - $x_{out} \Rightarrow x_1 \vee x_2$
- AND node: $x_{out} = x_1 \wedge x_2$
  - $x_{out} \Rightarrow x_1$
  - $x_{out} \Rightarrow x_2$
  - $\neg x_{out} \Rightarrow \neg x_1 \vee \neg x_2$

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.

Clicker Question

Which of the following statements is NOT true?

A: SAT $\leq_P$ 3-SAT

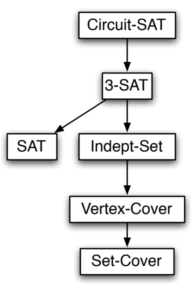
B: 3-SAT $\leq_P$ SAT

C: k-SAT $\leq_P$ SAT for all $k \geq 2$

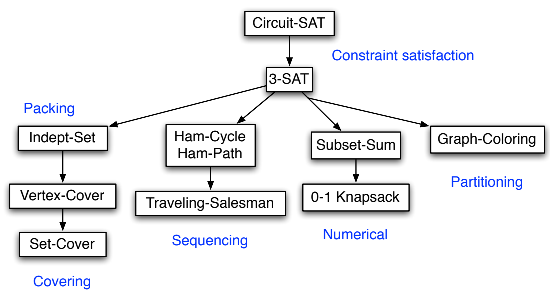
D: k-SAT is NP-complete for all $k \geq 2$

NP-Complete Problems So Far

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



NP-Complete Problems: Preview



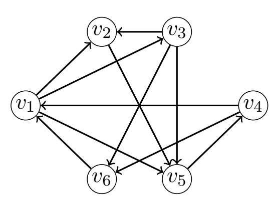
Traveling Salesman Problem

- TSP. Given n cities and distance function $d(i,j)$, is there a tour that visits all cities with total distance less than D?
  - Tour: ordering of cities $i_1, i_2, \ldots, i_n$ with $i_1 = 1$
  - Distance is $\sum_{j=1}^{n-1} d(i_j, i_{j+1}) + d(i_n, 1)$

- Applications: traveling salesperson, moving robotic arms
- Let's prove a simpler problem is NP-complete, and then use it to show TSP is NP-complete.

Hamiltonian Cycle Problem

- HamCycle – Hamiltonian Cycle. Given directed graph $G = (V, E)$, is there a cycle that visits each vertex exactly once?



- $v_1, v_3, v_2, v_5, v_4, v_6$ is a Hamiltonian Cycle

Ham-Cycle

Theorem. Ham-Cycle is NP-Complete.

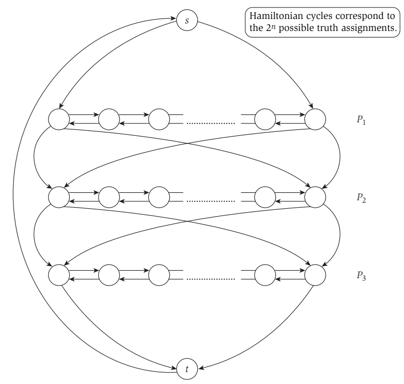
- It is in NP.
- Need to reduce from some NP-Complete problem. Which one?

Claim. 3-SAT $\leq_P$ Ham-Cycle.

Reduction has two main parts.

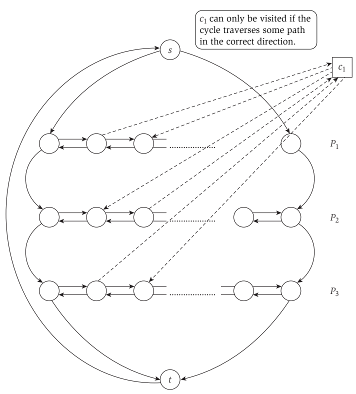
- Make a graph with 2^n Hamiltonian cycles, one per assignment.
- Augment graph with clauses to invalidate assignments.

Reduction: Graph skeleton



$x_i = 1 \iff$ traverse P_i from $L \to R$

Reduction: Clause Gadgets



$C_1 = x_1 \vee \bar{x}_2 \vee x_3$

Reduction: High-Level

- Correspondence between Hamiltonian cycles and truth assignments
  - $x_i = 1$: traverse path P_i from $L \to R$
  - $x_i = 0$: traverse path P_i from $R \to L$
- Node c_j for clause C_j must be visited in middle of *some* P_i
  - $x_i \in C_j \implies$ can visit c_j during $L \to R$ traversal of P_i.
    $x_i = 1$ satisfies C_j
  - $\bar{x}_i \in C_j \implies$ can visit c_j during $R \to L$ traveral of P_i.
    $x_i = 0$ satisfies C_j
- There is a Hamiltonian cycle
  $\iff$ can visit all clause nodes
  $\iff$ there is a truth assignment that satisfies all clauses

Reduction: Details

- n rows (bidirected paths) $P_1, \ldots, P_n$ (one per variable)
- Row has $3m + 3$ vertices, connected to neighbors in forward/backward direction
- First and last vertex of row i connected to first and last of $i + 1$.
- Source s connected to first and last of row 1.
- First and last of row n connected to t.
- Edge (t, s)
- Skeleton has 2^n possible Hamiltonian Cycles, corresponding to truth assignments to $x_1, \ldots, x_n$
  - Traverse P_i L to R $\iff$ $x_i = 1$
  - Traverse P_i R to L $\iff$ $x_i = 0$

Reduction: Clause Gadgets

For each clause C_ℓ construct gadget to restrict possible truth assignments

- New node c_ℓ
- If $x_i \in C_\ell$
  - Add edges $(v_{i,3\ell}, c_\ell)$ and $(c_\ell, v_{i,3\ell+1})$
  - c_ℓ can be visited during L to R traversal of P_i
- If $\neg x_i \in C_\ell$
  - Add edges $(v_{i,3\ell+1}, c_\ell)$ and $(c_\ell, v_{i,3\ell})$
  - c_ℓ can be visited during R to L traversal of P_i

Proof of Correctness

Given a satisfying assignment, construct Hamiltonian Cycle

- If $x_i = 1$ traverse P_i from $L \to R$, else $R \to L$.
- Each C_ℓ is satisfied, so one path P_i is traversed in the correct direction to "splice" c_ℓ into our cycle
- The result is a Hamiltonian Cycle

Given Hamiltonian cycle, construct satisfying assignment:

- If cycle visits c_ℓ from row i, it will also leave to row i because of "buffer" nodes
- Therefore, ignoring clause nodes, cycle traverses each row completely from $L \to R$ or $R \to L$
- Set $x_i = 1$ if P_i traversed $L \to R$, else $x_i = 0$
- Every node c_j visited $\Rightarrow$ every clause C_j is satisfied

Traveling Salesman

TSP. Given n cities and distance function $d(i, j)$, is there a tour that visits all cities with total distance less than D?

Theorem. TSP is NP-Complete

- Clearly in NP.
- Reduction? From Ham-Cycle

Clicker Question

We want to show that Ham-Cycle $\leq_P$ TSP. How can we do so?

Given a HamCycle instance $G = (V, E)$ make TSP instance with one city per vertex and...

A. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else 2. Tour distance: $\leq n$?

B. $d(v_i, v_j) = 2$ if $(v_i, v_j) \in E$, else 1. Tour distance: $\leq n$?

C. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else 2. Tour distance: $\leq m$?

Reduction from Ham-Cycle to TSP

Given HamCycle instance $G = (V, E)$ make TSP instance

- One city per vertex
- $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else 2

Claim: there is a tour of distance $\leq n$ if and only if G has a Hamiltonian cycle

- A Hamiltonian cycle clearly gives a tour of length n
- A tour of length n must travel n hops of length 1, which corresponds to a Hamiltonian cycle

Ham-Path

Similar to Hamiltonian Cycle, visit every vertex exactly once.

Theorem. Ham-Path is NP-Complete.

Two proofs.

- Modify 3-SAT to Ham-Cycle reduction.
- Reduce from Ham-Cycle directly.

NP-Complete Problems