

# CS 577 - Computational Intractability

Marc Renault

Department of Computer Sciences  
University of Wisconsin – Madison

Spring 2023

TopHat Section 001 Join Code: 020205  
TopHat Section 002 Join Code: 394523



# COMPUTATIONAL INTRACTABILITY

# COMPUTATIONAL INTRACTABILITY

## Easy Problems

- Problems that can be solved by efficient algorithms.
- Polynomial running time.
- Complexity class: P

## Hard Problems

- Problems for which we do not know how to solve efficiently.
- NP-hard
- NP-complete

# DECISION PROBLEM

Optimization:

## Bipartite Matching

Given a bipartite graph  $G$ ,  
find the largest matching.

Decision:

## Bipartite Matching

Given a bipartite graph  $G$ , is  
there a matching of size  $\geq k$ ?

### Decision Problem

- binary output: yes / no answer.

# DECISION PROBLEM

Optimization:

## Bipartite Matching

Given a bipartite graph  $G$ ,  
find the largest matching.



Decision:

## Bipartite Matching

Given a bipartite graph  $G$ , is  
there a matching of size  $\geq k$ ?

### Optimization to Decision

- Solve the optimization version.
- If the solution of size  $\geq k$ , return yes.

# DECISION PROBLEM

Optimization:

Bipartite Matching

Given a bipartite graph  $G$ ,  
find the largest matching.



Decision:

Bipartite Matching

Given a bipartite graph  $G$ , is  
there a matching of size  $\geq k$ ?

## Decision to Optimization

- Upper bound on maximum matching is  $N = \min(|A|, |B|)$ .
- For  $k = N$  to 0, return first  $k$  that returns yes.  
(Or, binary search between  $[0, N]$ .)

# REDUCTIONS

# POLYNOMIAL-TIME REDUCTION

## Problem Reduction: $Y \leq_p X$

- Consider any instance of problem  $Y$ .
- Assume we have a black-box solver for problem  $X$ .
- Efficiently transform an instance of problem  $Y$  into a polynomial number of instances of  $X$  that we solve (black-box solver) for problem  $X$  and aggregate efficiently to solve  $Y$ .

## $Y$ is polynomial-time reducible to $X$

Suppose  $Y \leq_p X$ . If  $X$  is solvable in polynomial time, then  $Y$  can be solved in polynomial time.

## $X$ is at least as hard as $Y$

Suppose  $Y \leq_p X$ . If  $Y$  cannot be solved in polynomial time, then  $X$  cannot be solved in polynomial time.

# INDEPENDENT SET $\iff$ VERTEX COVER

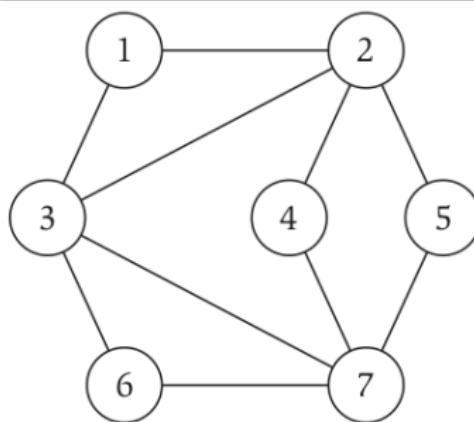
Given a graph  $G$  and a number  $k$ .

## Independent Set (IS)

- Does  $G$  contain an IS of size  $\geq k$ ?
- $S \subseteq V$  is *independent* if no 2 nodes in  $S$  are adjacent.

## Vertex Cover (VC)

- Does  $G$  contain a vertex cover of size  $\leq k$ ?
- $S \subseteq V$  is *vertex cover* if every edge is incident to at least 1 node in  $S$ .



TopHat 1: What is size of the largest independent set?

# INDEPENDENT SET $\iff$ VERTEX COVER

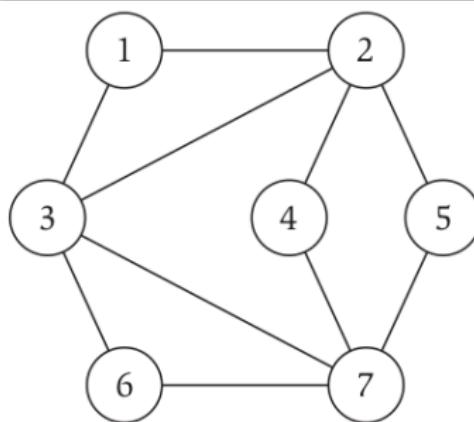
Given a graph  $G$  and a number  $k$ .

## Independent Set (IS)

- Does  $G$  contain an IS of size  $\geq k$ ?
- $S \subseteq V$  is *independent* if no 2 nodes in  $S$  are adjacent.

## Vertex Cover (VC)

- Does  $G$  contain a vertex cover of size  $\leq k$ ?
- $S \subseteq V$  is *vertex cover* if every edge is incident to at least 1 node in  $S$ .



TopHat 2: What is size of the smallest vertex cover?

# INDEPENDENT SET $\iff$ VERTEX COVER

Given a graph  $G$  and a number  $k$ .

## Independent Set (IS)

- Does  $G$  contain an IS of size  $\geq k$ ?
- $S \subseteq V$  is *independent* if no 2 nodes in  $S$  are adjacent.

## Vertex Cover (VC)

- Does  $G$  contain a vertex cover of size  $\leq k$ ?
- $S \subseteq V$  is *vertex cover* if every edge is incident to at least 1 node in  $S$ .

## Theorem 1

*Let  $G = (V, E)$  be a graph. Then  $S$  is an independent set if and only if its complement  $V \setminus S$  is a vertex cover.*

## Proof.

$\Rightarrow$ : Suppose  $S$  is an IS. For any edge  $(u, v)$ , at most one of  $\{u, v\} \in S$ . Hence, one of  $\{u, v\} \in V \setminus S$ .

# INDEPENDENT SET $\iff$ VERTEX COVER

Given a graph  $G$  and a number  $k$ .

## Independent Set (IS)

- Does  $G$  contain an IS of size  $\geq k$ ?
- $S \subseteq V$  is *independent* if no 2 nodes in  $S$  are adjacent.

## Vertex Cover (VC)

- Does  $G$  contain a vertex cover of size  $\leq k$ ?
- $S \subseteq V$  is *vertex cover* if every edge is incident to at least 1 node in  $S$ .

## Theorem 1

*Let  $G = (V, E)$  be a graph. Then  $S$  is an independent set if and only if its complement  $V \setminus S$  is a vertex cover.*

## Proof.

$\Leftarrow$ : Suppose  $V \setminus S$  is a VC. Any edge  $(u, v)$  with both  $\in S$  would contradict that  $V \setminus S$  is a VC. □

# PACKING AND COVERING PROBLEMS

## Packing Problem

### Independent Set

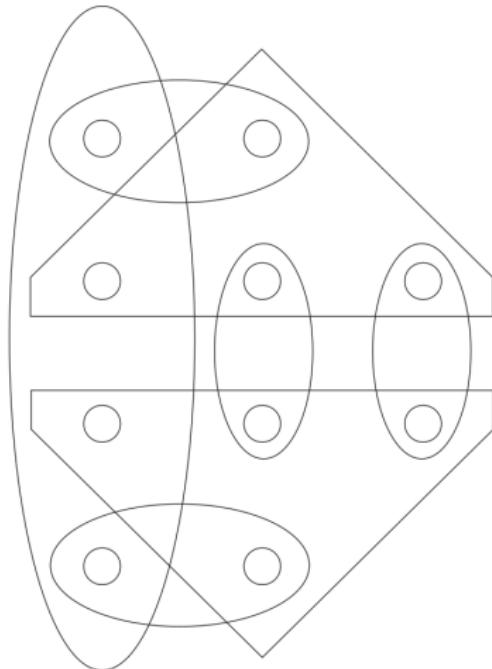
- Goal is to pack as many vertices as possible without violating edge constraints.

## Covering Problem

### Vertex Cover

- Goal is to cover all the edges in the graph using as few vertices as possible.

# SET COVER (SC)



## Problem Definition

- A universe  $U$  of  $n$  elements.
- A collection of subsets of  $U$ :  $S_1, S_2, \dots, S_m$ .
- A number  $k$ .
- Goal: Does there exist a collection of at most  $k$  of the subsets whose unions equal  $U$ .

# REDUCTION: VERTEX COVER (VC) TO SET COVER (SC)

## Theorem 2

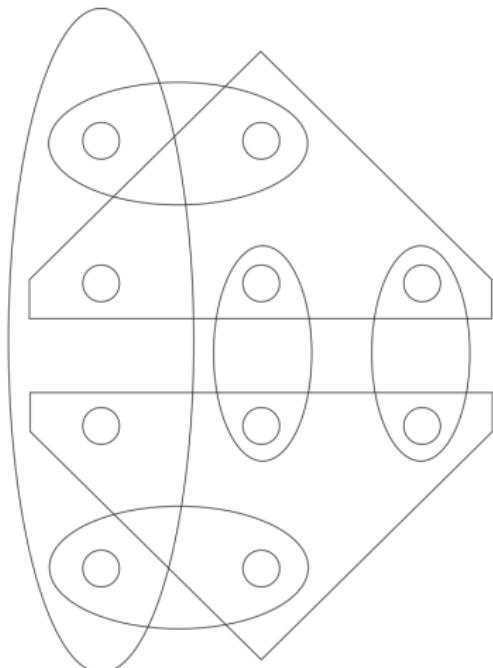
$$\text{VC} \leq_p \text{SC}$$

## Proof.

- Assume that we have a black-box solver for SC.
- Consider an arbitrary instance of VC on  $G = (V, E)$ .
  - Set  $U = E$ .
  - For each vertex  $v \in V$ :  
Create a set consisting of each edge incident to  $v$ .
- Direct correspondence between VC and SC.
  - $\text{VC} \leq k \iff \text{SC} \leq k$



# SET PACKING (SP)



## Problem Definition

- A universe  $U$  of  $n$  elements.
- A collection of subsets of  $U$ :  $S_1, S_2, \dots, S_m$ .
- A number  $k$ .
- Goal: Does there exist a collection of at least  $k$  of the subsets that don't intersect.

Exercise: Show that IS  $\leq_p$  SP

# REDUCTION: INDEPENDENT SET (IS) TO SET PACKING (SP)

## Theorem 3

$$\text{IS} \leq_p \text{SP}$$

## Proof.

- Assume that we have a black-box solver for SP.
- Consider an arbitrary instance of IS on  $G = (V, E)$ .
  - Set  $U = E$ .
  - For each vertex  $v \in V$ :  
Create a set consisting of each edge incident to  $v$ .
- Direct correspondence between IS and SP.
  - $\text{IS} \geq k \iff \text{SP} \geq k$



# SATISFIABILITY PROBLEM (SAT)

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

## Preliminaries

- A set of boolean terms/literals:  $X : x_1, \dots, x_n$ .
- For a given variable  $x_i$ ,  $x_i$  is the assigned value and  $\overline{x_i}$  is the negation of the assigned value.
- A clause  $C_j$  is a *disjunction* of (distinct) terms, e.g.,  $(x_1 \vee \overline{x_2})$ .
- Length of  $C_j$  is the # of terms in  $C_j$ .
- A collection/*conjunction* of  $k$  clauses:  $\mathcal{C} : C_1 \wedge C_2 \wedge \dots \wedge C_k$ .
- Truth assignment function  $v : X \rightarrow \{0, 1\}$ , assigns values to the terms and returns the conjunction of the clauses.
- $v$  is a *satisfying assignment* if  $\mathcal{C}$  is 1, i.e., all  $C_i$  evaluate to 1.

# THREE SATIFIABILITY (3SAT)

## SAT Problem

Given a set of literals:  $X : x_1, \dots, x_n$ , and a collection of clauses  $\mathcal{C} : C_1 \wedge C_2 \wedge \dots \wedge C_k$ , does there exist a satisfying assignment?

# THREE SATIFIABILITY (3SAT)

## 3SAT Problem

Given a set of literals:  $X : x_1, \dots, x_n$ , and a collection of clauses  $\mathcal{C} : C_1 \wedge C_2 \wedge \dots \wedge C_k$ , each of length 3, does there exist a satisfying assignment?

# THREE SATIFIABILITY (3SAT)

## 3SAT Problem

Given a set of literals:  $X : x_1, \dots, x_n$ , and a collection of clauses  $\mathcal{C} : C_1 \wedge C_2 \wedge \dots \wedge C_k$ , each of length 3, does there exist a satisfying assignment?

## Gadgets

*Gadgets* are often used to show  $Y \leq_p X$ .

- A subset of problem  $X$  that represents a component of problem  $Y$ .
- A procedure to convert some of the components of  $Y$  to a piece of problem  $X$ .

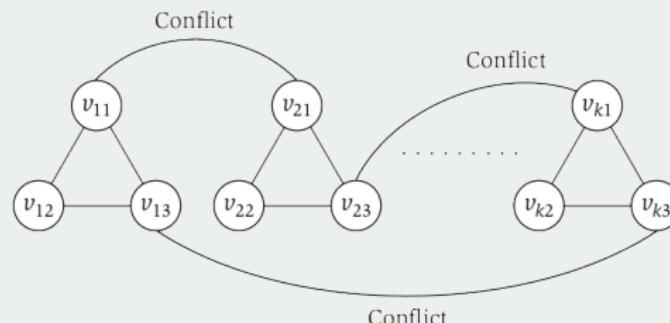
# 3SAT TO INDEPENDENT SET (IS)

## Theorem 4

$\text{3SAT} \leq_p \text{IS}$

## Proof.

- Assume we have a black-box solver for IS.
- Transfer any 3SAT to IS:
  - Clause gadget:  $k_3$  graph



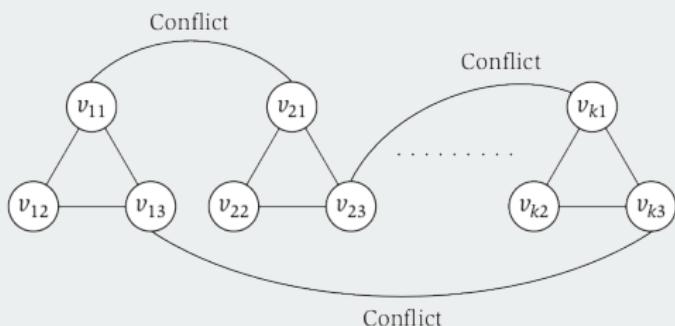
- Add an edge between  $v_{ij} = x_q$  and all  $v_{i'j'} = \bar{x}_q$ .

# 3SAT TO INDEPENDENT SET (IS)

## Theorem 4

$\text{3SAT} \leq_p \text{IS}$

## Proof.



- IS of size  $\geq k \iff$  3SAT is satisfiable.
  - Each node in IS represents a 1 assignment.
  - Within each gadget, only 1 node can be in IS.
  - Conflict edges prevent  $x_i$  and  $\bar{x}_i$  both being assigned 1.



# TRANSITIVITY OF REDUCTIONS

## Observation 1

If  $Z \leq_p Y$ , and  $Y \leq_p X$ , then  $Z \leq_p X$ .

So,

$$\text{3SAT} \leq_p \text{IS} \leq_p \text{VC} \leq_p \text{SC}$$

and

$$\text{3SAT} \leq_p \text{IS} \leq_p \text{SP}$$

and

$$\text{VC} \leq_p \text{IS} \leq_p \text{SP}.$$

# NP

# EFFICIENT CERTIFICATION

## Input Formalization

For a problem instance:

- Let  $s$  be a binary string that encodes the input.
- $|s|$  is the length of  $s$ , i.e., the # of bits in  $s$ .

## Polynomial Run-Time

Algorithm  $A$  has a *polynomial run-time* if run-time is  $O(\text{poly}(|s|))$  in the worst-case, where  $\text{poly}(\cdot)$  is a polynomial function.

## Complexity class P

P is the set of all problems for which there exists an algorithm  $A$  that solves the problem with polynomial run-time.

# EFFICIENT CERTIFICATION

## Efficient Certification

Certifier  $B(s, t)$  for a problem  $P$ :

- $s$  is an input instance of  $P$ .
- $t$  is a certificate; a proof that  $s$  is a yes-instance.
- Efficient:
  - For every  $s$ , we have  $s \in P$  iff there exists a  $t$ ,  $|t| \leq \text{poly}(|s|)$ , for which  $B(s, t)$  returns yes.
  - In other words, using  $t$ , we can check if  $s$  is a yes-instance in polynomial time.
- $B(s, t)$  returning no does not mean that  $s$  is a no-instance... only that  $t$  is not a valid proof.
- $B(s, t)$  provides a brute-force algorithm: For a given  $s$ , check every possible  $t$ .

# NP PROBLEMS

## Complexity Class NP

- Non-deterministic, Polynomial time: can be solved in polynomial time by testing every  $t$  simultaneously (non-deterministic).
- Set of all problems for which there exists an efficient certifier.

## Theorem 5

$$P \subseteq NP$$

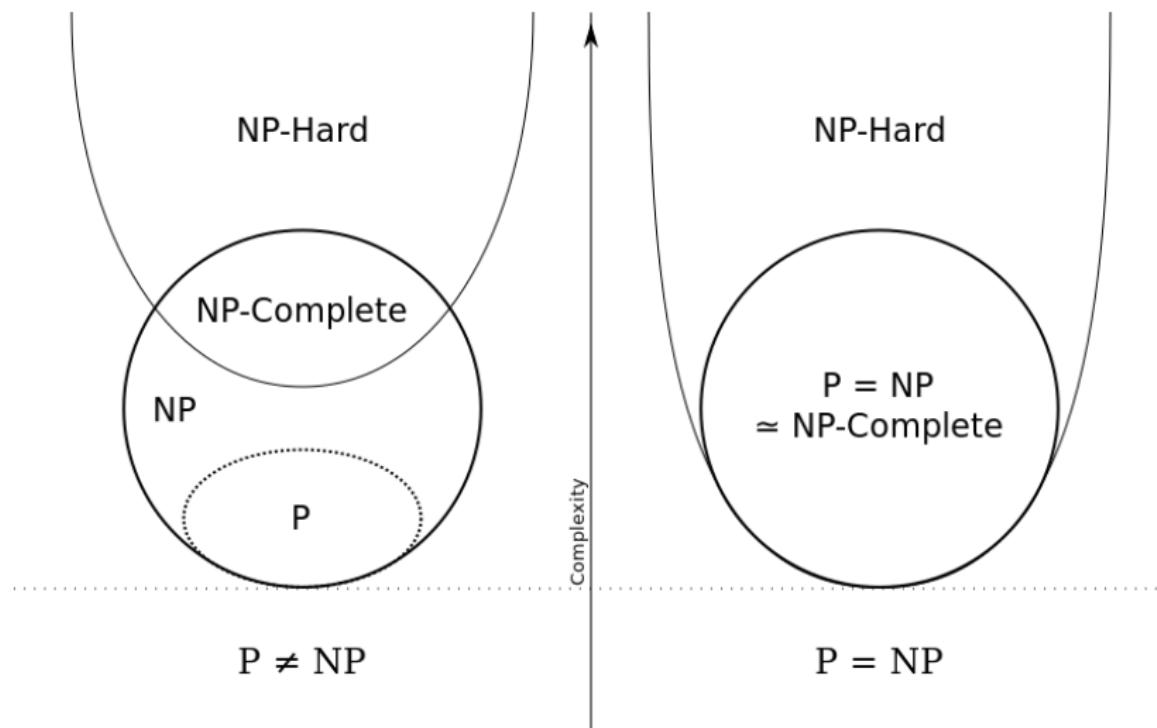
## Proof.

- For every  $p \in P, \exists$  an algorithm  $A$  that runs in polynomial time.
- $B(s, t)$  for any  $t$  returns  $A(s)$ .



# MILLION DOLLAR QUESTION: P vs NP

1 OF 7 CLAY MATHEMATICS INSTITUTE MILLENNIUM PRIZE PROBLEMS



# NP-COMPLETE

# HARDEST NP PROBLEMS

## NP-Hard

Problem  $X$  is NP-Hard if:

- For all  $Y \in \text{NP}$ ,  $Y \leq_p X$ .
- NP-Hard problem may or may not be in NP.

## NP-Complete

Problem  $X$  is NP-Complete if:

- For all  $Y \in \text{NP}$ ,  $Y \leq_p X$ .
- $X$  is in NP.

# HARDEST NP PROBLEMS

## NP-Complete

Problem  $X$  is NP-Complete if:

- For all  $Y \in \text{NP}$ ,  $Y \leq_p X$ .
- $X$  is in NP.

## Theorem 6

*Suppose  $X \in \text{NP-Complete}$ . Then,  $X$  is solvable in polynomial time iff  $P = \text{NP}$ .*

## Proof.

$\Leftarrow$ : Suppose  $P = \text{NP}$ , then by definition of  $P$ ,  $X$  can be solved in polynomial time.

# HARDEST NP PROBLEMS

## NP-Complete

Problem  $X$  is NP-Complete if:

- For all  $Y \in \text{NP}$ ,  $Y \leq_p X$ .
- $X$  is in NP.

## Theorem 6

*Suppose  $X \in \text{NP-Complete}$ . Then,  $X$  is solvable in polynomial time iff  $P = \text{NP}$ .*

## Proof.

$\Rightarrow$ : Suppose  $X$  can be solved in polynomial time. Then, by definition of NP-Complete, all problems  $\in \text{NP} \leq_p X$ . Hence, solvable in polynomial time and  $\in P$ . □

# FIRST NP-COMPLETE PROBLEM

## Theorem 6

*Cook (1971) – Levin (1973) Theorem [Paraphrase]: Circuit Satisfiability Problem (CSAT) is NP-Complete.*

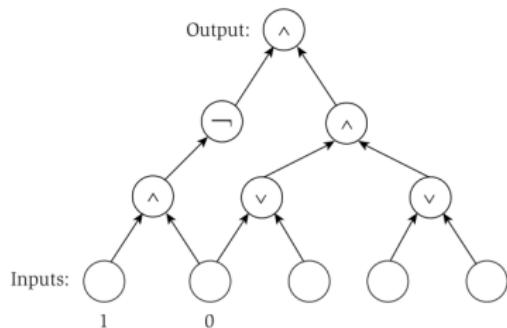


Stephen Cook  
(1968)



Leonid Levin  
(2010)

# CIRCUIT SATISFIABILITY PROBLEM (CSAT)



## Problem Definition

- 3 types of gates:  $\wedge$  (AND),  $\vee$  (OR), and  $\neg$  (NOT).
- Circuit  $k$ :
  - A DAG (nodes have 0, 1, or 2 incoming edges).
  - Source: Nodes with no incoming edges; may have a preset binary value.
  - Every other node is labelled with a gate.
  - Output: Result of the node with no outgoing edges.

# FIRST NP-COMPLETE PROBLEM

## Theorem 6

*Cook (1971) – Levin (1973) Theorem [Paraphrase]: Circuit Satisfiability Problem (CSAT) is NP-Complete.*

## Partial Proof.

① Show that CSAT  $\in$  NP:

- Input size is  $\Omega(|V|)$ .
- A single gate can be evaluated in constant time.
- Evaluate a certificate of the inputs can be verified in  $O(|V|)$  time.

# FIRST NP-COMPLETE PROBLEM

## Theorem 6

*Cook (1971) – Levin (1973) Theorem [Paraphrase]: Circuit Satisfiability Problem (CSAT) is NP-Complete.*

## Partial Proof.

- ① Show that  $\text{CSAT} \in \text{NP}$ :
- ② Reduce every problem  $\in \text{NP}$  to CSAT:
  - Consider an arbitrary problem  $X \in \text{NP}$ .
  - We need to show  $X \leq_p \text{CSAT}$ .
  - By definition for X:
    - X has an input of  $|s|$  bits.
    - Produces 1 bit of output (yes/no).
    - $\exists$  an efficient certifier  $B_X(\cdot, \cdot)$ .

# FIRST NP-COMPLETE PROBLEM

## Theorem 6

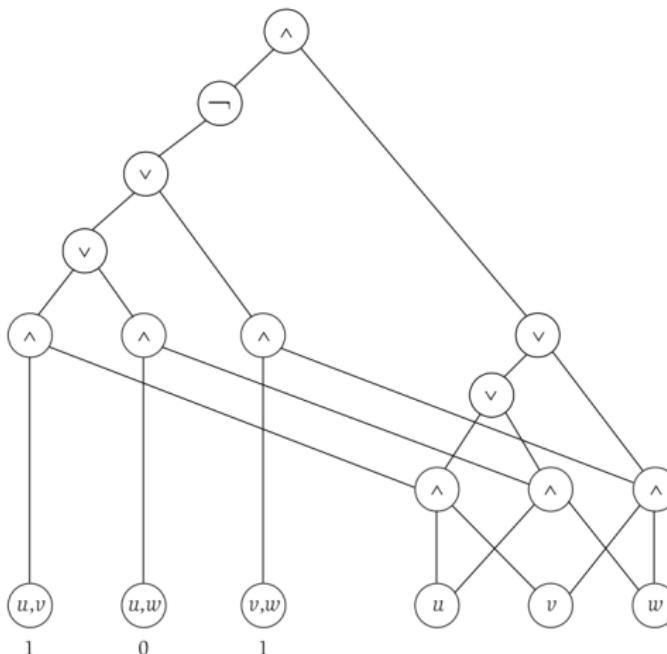
*Cook (1971) – Levin (1973) Theorem [Paraphrase]: Circuit Satisfiability Problem (CSAT) is NP-Complete.*

### Partial Proof.

- ① Show that  $\text{CSAT} \in \text{NP}$ :
- ② Reduce every problem  $\in \text{NP}$  to CSAT:
  - Consider an arbitrary problem  $X \in \text{NP}$ .
  - Reduction to CSAT:
    - Output is 1 when  $X$  is yes; otherwise 0.
    - Sources:  $|s| + |t| = n + \text{poly}(n)$  bits.
    - The first  $n$  bits are hard-coded to the  $X$  instance input.
    - The  $\text{poly}(n)$  bits are free and used to find a  $t$  such that  $B_X(s, t)$  is yes.
    - The gates of the circuit are a translation of algorithm  $B_X$ .



## EXAMPLE: INDEPENDENT SET ( $k \geq 2$ ) AS CIRCUIT SATISFIABILITY PROBLEM.



TH8: Draw the underlying Independent Set graph.

# STRATEGIES FOR PROVING NP-COMPLETENESS

## Showing that Problem X is NP-Complete

Cook Reduction:

- ① Prove that  $X \in \text{NP}$ .
- ② Choose a problem  $Y \in \text{NP-Complete}$ .
- ③ Prove  $Y \leq_p X$ .

## Typical Step 3

- ③ Karp Reduction: For an arbitrary instance  $s_Y$  of  $Y$ , show how to construct, in polynomial time, an instance  $s_X$  of  $X$  such that  $s_Y$  is a yes iff  $s_X$  is a yes.

Steps:

- ① Provide efficient reduction.
- ② Prove  $\Rightarrow$ : if  $s_Y$  is a yes,  $s_X$  is a yes.
- ③ Prove  $\Leftarrow$ : if  $s_X$  is a yes, then  $s_Y$  had to have been a yes.

# 3SAT is NP-COMPLETE

## Theorem 7

3SAT is NP-Complete.

Show that Problem 3SAT is NP-Complete

Cook Reduction:

- ① Prove that 3SAT  $\in$  NP.
- ② Choose a problem  $Y \in$  NP-Complete.
- ③ Prove  $Y \leq_p$  3SAT.

Proof.

- ① Use a truth assignment of the literals as a certificate. This can be verified in polynomial time.
- ② The only NP-Complete problem we know is CSAT.

# 3SAT is NP-COMPLETE

## Proof.

- ③ For an arbitrary circuit  $k$ :

- Each node  $v$  is assigned a variable  $x_v$ .
- For each gate:
  - NOT: Let  $u$  be the input. We need  $x_u = \overline{x_v}$ .  
 → 2 clauses:  $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$ .
  - OR: Let  $u, w$  be the inputs. We need  $x_v = x_u \vee x_w$ .  
 → 3 clauses:  $(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$ .
  - AND: Let  $u, w$  be the inputs. We need  $x_v = x_u \wedge x_w$ .  
 → 3 clauses:  $(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$ .
- For each constant source  $s$ :  
 → 1 clause:  $(x_s)$  if 1, and  $(\overline{x_s})$  if 0.
- For the output  $o$ : 1 clause  $(x_o)$ .
- Convert clauses to length 3:
  - We need 2 variables  $z_1$  and  $z_2$  that are always 0 in a satisfying assignment.
  - To ensure this, we need 4 variables:  $z_1, z_2, z_3, z_4$ .

# 3SAT is NP-COMPLETE

## Proof.

- ③ For an arbitrary circuit  $k$ :

- Each node  $v$  is assigned a variable  $x_v$ .
- For each gate:
  - NOT: Let  $u$  be the input. We need  $x_u = \overline{x_v}$ .  
→ 2 clauses:  $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$ .
  - OR: Let  $u, w$  be the inputs. We need  $x_v = x_u \vee x_w$ .  
→ 3 clauses:  $(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$ .
  - AND: Let  $u, w$  be the inputs. We need  $x_v = x_u \wedge x_w$ .  
→ 3 clauses:  $(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$ .
- For each constant source  $s$ :  
→ 1 clause:  $(x_s)$  if 1, and  $(\overline{x_s})$  if 0.
- For the output  $o$ : 1 clause  $(x_o)$ .
- Convert clauses to length 3:
  - 4 variables:  $z_1, z_2, z_3, z_4$ , and 8 clauses for  $i \in \{1, 2\}$ :  
 $(\overline{z_i} \vee z_3 \vee z_4) \wedge (\overline{z_i} \vee \overline{z_3} \vee z_4) \wedge (\overline{z_i} \vee z_3 \vee \overline{z_4}) \wedge (\overline{z_i} \vee \overline{z_3} \vee \overline{z_4})$ .

# 3SAT is NP-COMPLETE

## Proof.

③  $s_{\text{CSAT}}$  is a yes iff  $s_{\text{3SAT}}$  is a yes:

- $\Rightarrow$ : If  $s_{\text{CSAT}}$  is a yes, then the satisfying assignment to the circuit inputs can be used to calculate the value of each gate. By the reduction, these values will satisfy all the clauses of  $s_{\text{3SAT}}$ .
- $\Leftarrow$ : If  $s_{\text{3SAT}}$  is a yes, then the assignment of the variables give the satisfying assignment of the circuit inputs, and the reduction guarantees that the assigned values for the nodes match the gate calculations.



# 3SAT is NP-COMPLETE

From our previous reductions

$$\text{3SAT} \leq_p \text{IS} \leq_p \text{VC} \leq_p \text{SC}$$

and

$$\text{3SAT} \leq_p \text{IS} \leq_p \text{SP}$$

and the fact that 3SAT is NP-Complete:

## Corollary 7

*The following problems are NP-Complete:*

3SAT, IS, VC, SC, SP .

# TAXONOMY OF NP-COMPLETENESS

# SEQUENCING PROBLEMS

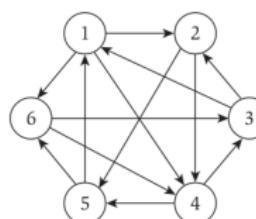
## Travelling Salesperson Problem (TSP)

- A salesperson must visit  $n$  cities  $v_1, v_2, \dots, v_n$ .
- Starting at some  $v_1$ , visit all cities and return to  $v_1$ .
- Distance function:  $d(\cdot, \cdot)$  for all pairs of cities (not necessarily symmetric nor metric).
- Optimization: What is the shortest tour?

# SEQUENCING PROBLEMS

## Travelling Salesperson Problem (TSP)

- A salesperson must visit  $n$  cities  $v_1, v_2, \dots, v_n$ .
- Starting at some  $v_1$ , visit all cities and return to  $v_1$ .
- Distance function:  $d(\cdot, \cdot)$  for all pairs of cities (not necessarily symmetric nor metric).



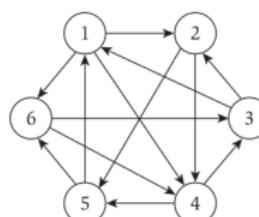
## Hamiltonian Cycle

- Graph analogue of TSP.
- *Hamiltonian cycle*: a tour of the nodes of  $G$  that visits each node once.
- Given a digraph  $G$ , does it contain a Hamiltonian cycle?

# SEQUENCING PROBLEMS

## Travelling Salesperson Problem (TSP)

- A salesperson must visit  $n$  cities  $v_1, v_2, \dots, v_n$ .
- Starting at some  $v_1$ , visit all cities and return to  $v_1$ .
- Distance function:  $d(\cdot, \cdot)$  for all pairs of cities (not necessarily symmetric nor metric).



TH9: Does this graph contain a Hamiltonian cycle?

## Hamiltonian Cycle

- Graph analogue of TSP.
- *Hamiltonian cycle*: a tour of the nodes of  $G$  that visits each node once.
- Given a digraph  $G$ , does it contain a Hamiltonian cycle?

# 3SAT $\leq_p$ Hamiltonian

## Theorem 8

*Hamiltonian Cycle is NP-complete.*

## Proof.

- ① In NP: A certificate would be a sequence of vertices which can be verified in polynomial time.
- ② Choose an NP-complete problem: 3SAT.

# $3\text{SAT} \leq_p \text{Hamiltonian}$

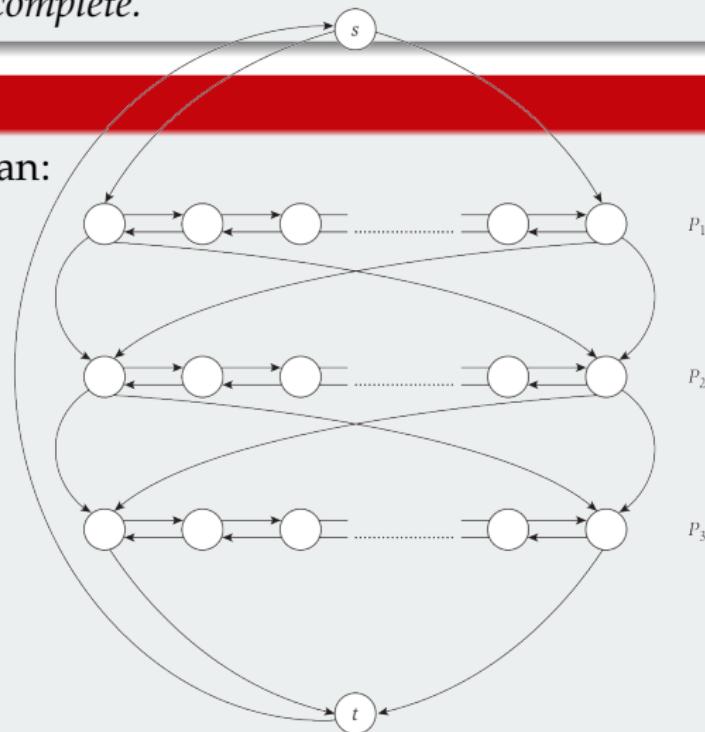
## Theorem 8

*Hamiltonian Cycle is NP-complete.*

## Proof.

- ⑤  $3\text{SAT} \leq_p \text{Hamiltonian}:$

- $P_i$  (containing  $3k + 2$  nodes) for each  $X_i$ : left traversal for 1 and right traversal for 0.



# TRAVELLING SALESPERSON

## Theorem 9

*Travelling Salesperson (TSP) is NP-complete.*

## Proof.

- ➊ In NP: Certificate that is a tour of the cities.
- ➋ Use Hamiltonian Cycle.
- ➌ Hamiltonian Cycle  $\leq_p$  TSP:  
Given a graph  $G = (V, E)$ :
  - For each  $v$ , make a city.
  - For each edge  $(u, v) \in E$ , define  $d(u, v) = 1$ .

# TRAVELLING SALESPERSON

## Theorem 9

*Travelling Salesperson (TSP) is NP-complete.*

## Proof.

- ➊ In NP: Certificate that is a tour of the cities.
- ➋ Use Hamiltonian Cycle.
- ➌ Hamiltonian Cycle  $\leq_p$  TSP:

Given a graph  $G = (V, E)$ :

- For each  $v$ , make a city.
- For each edge  $(u, v) \in E$ , define  $d(u, v) = 1$ .
- For each pair  $(u, v) \notin E$ , define  $d(u, v) = 2$ .
- Set the tour bound to be  $n$ .

# EXERCISE: SHOW THAT HAMILTONIAN PATH IS NP-COMPLETE

## Theorem 10

*Hamiltonian Path is NP-complete*

### Proof.

- ➊ In NP: Certificate is a path in  $G$  which can be verified in polynomial time.
- ➋ NP-complete problem: Hamiltonian Cycle.

# EXERCISE: SHOW THAT HAMILTONIAN PATH IS NP-COMPLETE

## Theorem 10

*Hamiltonian Path is NP-complete*

## Proof.

- ③ Hamiltonian Cycle  $\leq_p$  Hamiltonian Path:  
For  $G = (V, E)$  create  $G'$ :

# EXERCISE: SHOW THAT HAMILTONIAN PATH IS NP-COMPLETE

## Theorem 10

*Hamiltonian Path is NP-complete*

## Proof.

- ➊ Hamiltonian Cycle  $\leq_p$  Hamiltonian Path:

For  $G = (V, E)$  create  $G'$ :

- Choose an arbitrary  $v \in V$ :  $V' = V \setminus \{v\} \cup \{v', v''\}$ .
- Initialize  $E' = E$ :
  - For each edge  $(v, w) \in E$ :  $E' \setminus \{(v, w)\} \cup \{(v', w)\}$ .
  - For each edge  $(u, v) \in E$ :  $E' \setminus \{(u, v)\} \cup \{(u, v'')\}$ .
- A path  $v' \rightarrow v''$  means Hamiltonian Cycle.



# PARTITIONING PROBLEMS

## 3-D Matching

- Given 3 disjoint sets:  $X, Y, Z$  (each of size  $n$ ).
- A set of  $m \geq n$  trebles  $T \subseteq X \times Y \times Z$ .
- Does there exist a set of  $n$  trebles from  $T$  so that each item is in exactly one of these trebles?

# 3-D MATCHING IS NP-COMPLETE

## Theorem 11

*3-D Matching is NP-Complete.*

## Proof.

# 3-D MATCHING IS NP-COMPLETE

## Theorem 11

*3-D Matching is NP-Complete.*

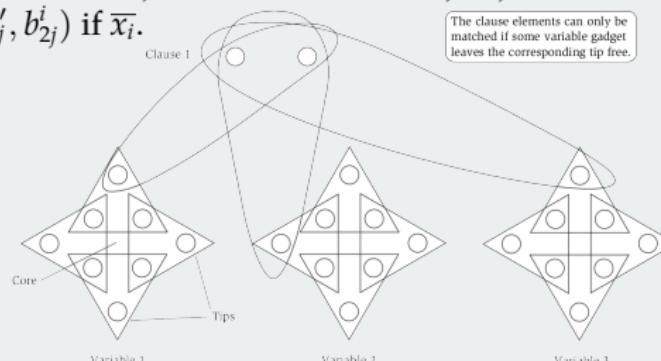
## Proof.

### ⑤ 3SAT $\leq_p$ 3-D Matching:

Consider an arbitrary 3SAT:

- Clause  $C_j$  gadget:

- Add  $P_j = \{p_j, p'_j\}$  with trebles:  $(p_j, p'_j, b_{2j-1}^i)$  if  $x_i$  and  $(P_j, P'_j, b_{2j}^i)$  if  $\bar{x}_i$ .



# 3-D MATCHING IS NP-COMPLETE

## Theorem 11

*3-D Matching is NP-Complete.*

## Proof.

### ⑤ 3SAT $\leq_p$ 3-D Matching:

Consider an arbitrary 3SAT:

- Counting cores: covered by even/odd choice.
- Counting tips:  $n2k$ 
  - Even/odd tips cover  $nk$ .
  - Clauses cover  $k$ .
  - $(n - 1)k$  uncovered.
- $(n - 1)k$  clean-up gadgets:
  - $Q_i = \{q_i, q'_i\}$  with treble  $(q_i, q'_i, b)$  for every tip  $b$ .

# 3-D MATCHING IS NP-COMPLETE

## Theorem 11

*3-D Matching is NP-Complete.*

## Proof.

### ⑤ 3SAT $\leq_p$ 3-D Matching:

Consider an arbitrary 3SAT:

- $\Rightarrow$  For a yes 3SAT, there is a matching that takes the even/odd tip trebles, leaving at least one tip as part of each clause gadget treble. The remaining unmatched tips are matched to a clean-up gadget.
- $\Leftarrow$  A yes for 3-D Matching from the reduction means that each clause gadget is part of a selected treble, each variable gadget has selected the odd or even tips, and the remaining tips are matched to a clean-up gadget. Each clause will be satisfied by the tip matched by the clause gadget. The even/odd selection for each variable guarantees all variables are assigned 1 or 0.

# 3-COLOURING IS NP-COMPLETE

## Theorem 12

*3-Colouring is NP-Complete.*

## Proof.

- ① In NP: Certificate is a colouring of the nodes which can be verified in polynomial time.
- ② TH12: Which NP-complete problem?

# 3-COLOURING IS NP-COMPLETE

## Theorem 12

*3-Colouring is NP-Complete.*

## Proof.

- ① In NP: Certificate is a colouring of the nodes which can be verified in polynomial time.
- ② NP-complete problem: 3SAT.

# 3-COLOURING IS NP-COMPLETE

## Theorem 12

*3-Colouring is NP-Complete.*

## Proof.

⑥ 3SAT  $\leq_p$  3 Colouring:

- For each literal: Nodes  $v_i$  and  $\bar{v}_i$ .

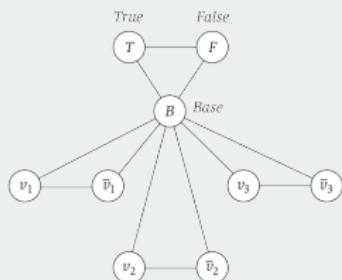
# 3-COLOURING IS NP-COMPLETE

## Theorem 12

*3-Colouring is NP-Complete.*

## Proof.

### ⑥ 3SAT $\leq_p$ 3 Colouring:



- For each literal: Nodes  $v_i$  and  $\bar{v}_i$ .
- Nodes  $T$  (true),  $F$  (false), and  $B$  (base).
- Edges:  $(v_i, \bar{v}_i), (v_i, B), (\bar{v}_i, B)$ .
- Edges:  $(T, F), (F, B), (T, B)$ .

# SUBSET SUM IS NP-COMPLETE

## Theorem 13

*Subset Sum is NP-Complete.*

## Proof.

- ➊ 3-D Matching  $\leq_p$  Subset Sum: Exercise: Try it, but tough.

# SUBSET SUM IS NP-COMPLETE

## Theorem 13

*Subset Sum is NP-Complete.*

## Proof.

- ❸ 3-D Matching  $\leq_p$  Subset Sum:
  - 3-D Matching: Subsets can be viewed as length  $3n$  bit vectors with a 1 indicating that item is in the set.

# SUBSET SUM IS NP-COMPLETE

## Theorem 13

*Subset Sum is NP-Complete.*

## Proof.

### ③ 3-D Matching $\leq_p$ Subset Sum:

- 3-D Matching: Subsets can be viewed as length  $3n$  bit vectors with a 1 indicating that item is in the set.
- For each treble  $(i, j, k)$  from  $X \times Y \times Z$  construct a  $w_t$ :

# SUBSET SUM IS NP-COMPLETE

## Theorem 13

*Subset Sum is NP-Complete.*

## Proof.

### ③ 3-D Matching $\leq_p$ Subset Sum:

- 3-D Matching: Subsets can be viewed as length  $3n$  bit vectors with a 1 indicating that item is in the set.
- For each treble  $(i, j, k)$  from  $X \times Y \times Z$  construct a  $w_t$ :
  - A digits with 1 at  $i, n + j$ , and  $2n + k$ .

# SUBSET SUM IS NP-COMPLETE

## Theorem 13

*Subset Sum is NP-Complete.*

## Proof.

### ③ 3-D Matching $\leq_p$ Subset Sum:

- 3-D Matching: Subsets can be viewed as length  $3n$  bit vectors with a 1 indicating that item is in the set.
- For each treble  $(i, j, k)$  from  $X \times Y \times Z$  construct a  $w_t$ :
  - A digits with 1 at  $i, n + j$ , and  $2n + k$ .
  - For base  $d$ ,  $w_t = d^{i-1} + d^{n+j-1} + d^{2n+k-1}$ .
  - Set base  $d = m + 1$  to avoid addition carry overs.
- Set  $W = \sum_0^{3n-1} (m + 1)^i$  which corresponds to have each item exactly once.



# ASYMMETRY OF NP

## Efficient Certifier Asymmetry

Given an instance  $s$  of problem  $X$ :

- For any  $t$ ,  $B(s, t) = \text{yes}$  implies yes-instance.
- For all  $t$ ,  $B(s, t) = \text{no}$  implies no-instance.

## Complimentary Problem

For every problem  $X$ , there is a complementary problem  $\overline{X}$ :

- For all input  $s$ ,  $s \in X$  iff  $s \notin \overline{X}$ .
- Note that, if  $X \in P$ , then  $\overline{X} \in P$ .

# coNP

## Complexity Class coNP

A problem  $X \in \text{coNP}$  iff  $\bar{X} \in \text{NP}$ .

# coNP

## Complexity Class coNP

A problem  $X \in \text{coNP}$  iff  $\bar{X} \in \text{NP}$ .

## Open Question

Does  $\text{NP} = \text{coNP}$ ?

# coNP

## Complexity Class coNP

A problem  $X \in \text{coNP}$  iff  $\bar{X} \in \text{NP}$ .

## Open Question

Does  $\text{NP} = \text{coNP}$ ?

## Theorem 14

*If  $\text{NP} \neq \text{coNP}$ , then  $\text{P} \neq \text{NP}$ .*

## Open Question

Does  $\text{P} = \text{NP} \cap \text{coNP}$ ?

# BEYOND TIME

## Complexity Class PSPACE

Set of all problems that can be solved using polynomial space.

### Theorem 15

$P \subseteq PSPACE$