CS 577 - Computational Intractability

Manolis Vlatakis

Department of Computer Sciences
University of Wisconsin – Madison

Fall 2024

Computational Intractability

Computational Intractability

Easy Problems

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

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.

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

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 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.

$\Longleftrightarrow$

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.

$\Longleftrightarrow$

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.

Decision Problem

Optimization:

Bipartite Matching

Given a bipartite graph G, find the largest matching.

$\iff$

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.

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.

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.

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.

INDEPENDENT SET $\iff$ VERTEX COVER

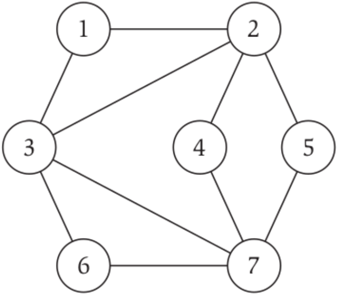
Given a graph G and a number k.

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.



🤔What is size of the largest independent set?

INDEPENDENT SET $\iff$ VERTEX COVER

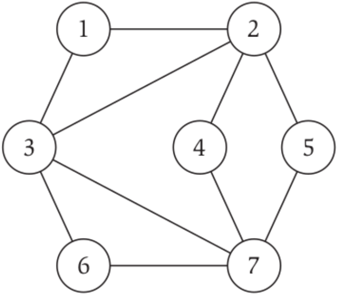
Given a graph G and a number k.

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.



🤔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.

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. $\square$

Packing and Covering Problems

Packing Problem

Independent Set

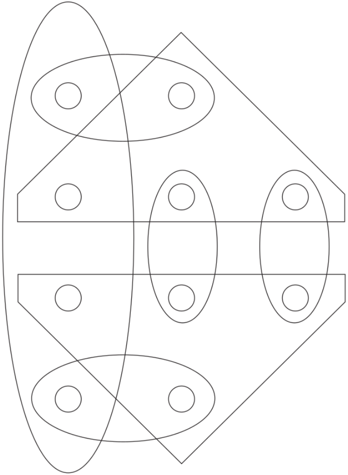
- Goal is to pack as many vertices as possible without violating edge constraints.

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: $S_1, S_2, \ldots, S_m$.
- 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

VC $\leq_p$ SC

Reduction: Vertex Cover (VC) to Set Cover (SC)

Theorem 2

VC $\leq_p$ SC

Proof.

- 🤔: For the proof, do we assume a VC or a SC black-box?

Reduction: Vertex Cover (VC) to Set Cover (SC)

Theorem 2

VC $\leq_p$ SC

Proof.

- Assume that we have a black-box solver for SC.

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

Theorem 2

VC $\leq_p$ 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.

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

Theorem 2

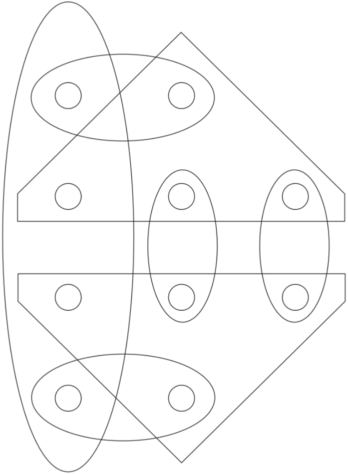
VC $\leq_p$ SC

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.
  - VC $\leq k \iff$ SC $\leq k$

□

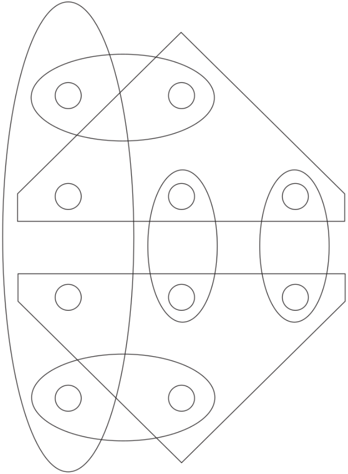
SET PACKING (SP)



Problem Definition

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

Set Packing (SP)



Exercise: Show that IS $\leq_p$ SP

Problem Definition

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

Reduction: Independent Set (IS) to Set Packing (SP)

Theorem 3

IS $\leq_p$ SP

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

Theorem 3

IS $\leq_p$ SP

Proof.

- Assume that we have a black-box solver for SP.

Reduction: Independent Set (IS) to Set Packing (SP)

Theorem 3

IS $\leq_p$ 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.

Reduction: Independent Set (IS) to Set Packing (SP)

Theorem 3

IS $\leq_p$ 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.
  - IS $\geq k \iff$ SP $\geq k$

$\square$

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: $X : x_1, \ldots, x_n$.

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: $X : x_1, \ldots, x_n$.
- For a given variable x_i, x_i is the assigned value and $\overline{x_i}$ is the negation of the assigned value.

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: $X : x_1, \ldots, x_n$.
- 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., $(x_1 \vee \overline{x_2})$.

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: $X : x_1, \ldots, x_n$.
- 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.

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: $X : x_1, \ldots, x_n$.
- 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: $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$.

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: $X : x_1, \ldots, x_n$.
- 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: $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$.
- Truth assignment function $v : X \to \{0, 1\}$, assigns values to the terms and returns the conjunction of the clauses.

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: $X : x_1, \ldots, x_n$.
- 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: $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$.
- Truth assignment function $v : X \to \{0, 1\}$, 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.

SATISFIABILITY PROBLEM (SAT)

: What values will satisfy the example?

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

Preliminaries

- A set of boolean terms/literals: $X : x_1, \ldots, x_n$.
- 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: $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$.
- Truth assignment function $v : X \to \{0, 1\}$, 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: $X : x_1, \ldots, x_n$, and a collection of clauses $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$, does there exist a satisfying assignment?

THREE SATIFIABILITY (3SAT)

3SAT Problem

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

THREE SATIFIABILITY (3SAT)

3SAT Problem

Given a set of literals: $X : x_1, \ldots, x_n$, and a collection of clauses $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$, 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.

THREE SATIFIABILITY (3SAT)

3SAT Problem

Given a set of literals: $X : x_1, \ldots, x_n$, and a collection of clauses $\mathcal{C} : C_1 \wedge C_2 \wedge \cdot \wedge C_k$, 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

3SAT $\leq_p$ IS

3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.

- Assume we have a black-box solver for IS.

3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.

- Assume we have a black-box solver for IS.
- Transfer any 3SAT to IS:

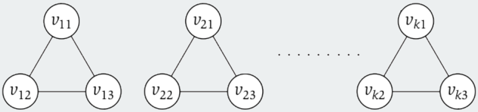
3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.

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



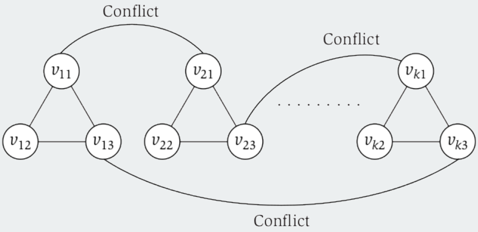
3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.

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



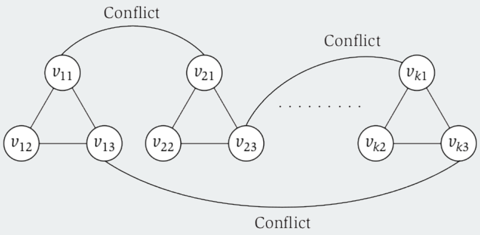
  - Add an edge between $v_{ij} = x_q$ and all $v_{i'j'} = \overline{x_q}$.

3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.



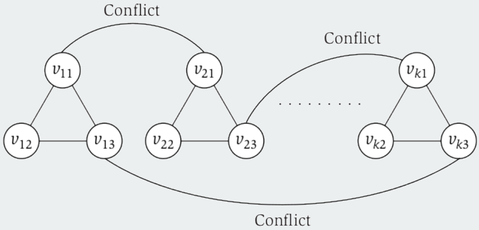
- IS of size $\geq k \iff$ 3SAT is satisfiable.

3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.



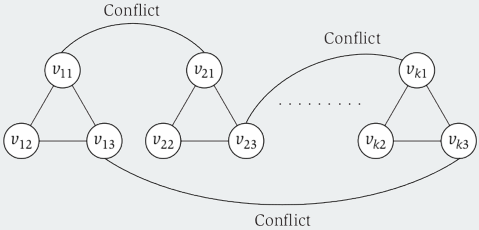
- IS of size $\geq k \iff$ 3SAT is satisfiable.
  - Each node in IS represents a 1 assignment.

3SAT to Independent Set (IS)

Theorem 4

3SAT $\leq_p$ IS

Proof.



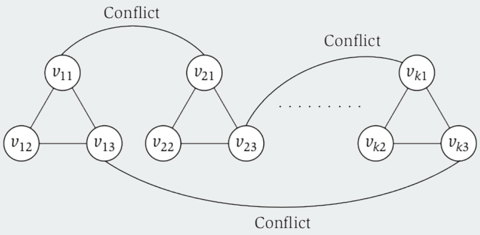
- IS of size $\geq k$ $\iff$ 3SAT is satisfiable.
    - Each node in IS represents a 1 assignment.
    - Within each gadget, only 1 node can be in IS.

3SAT TO INDEPENDENT SET (IS)

Theorem 4

3SAT $\leq_p$ 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 $\overline{x_i}$ both being assigned 1.

$\square$

Transitivity of Reductions

Observation 1

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

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.

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 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.

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.

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

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.

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

- **N**on-deterministic, **P**olynomial 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.

NP Problems

Complexity Class NP

- **N**on-deterministic, **P**olynomial 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

$P \subseteq NP$

NP Problems

Complexity Class NP

- **N**on-deterministic, **P**olynomial 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.

🤔: Which proof technique?

NP Problems

Complexity Class NP

- **N**on-deterministic, **P**olynomial 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 ⊆ NP

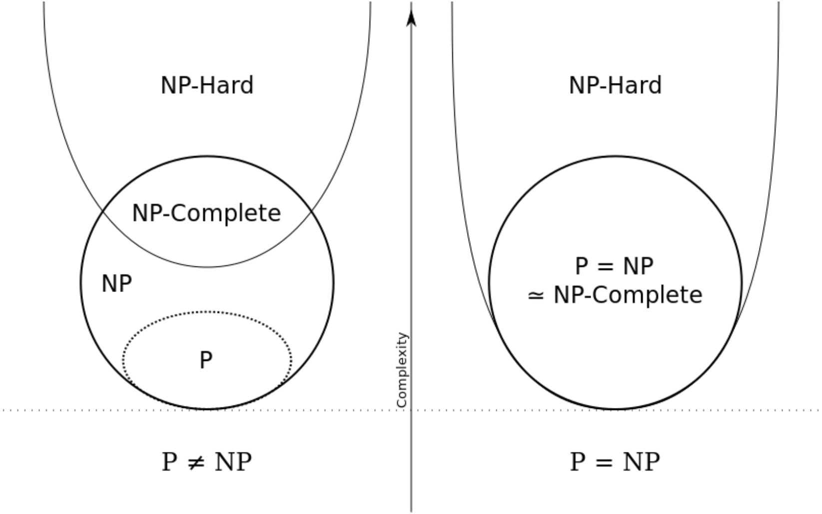
Proof.

- 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.

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.

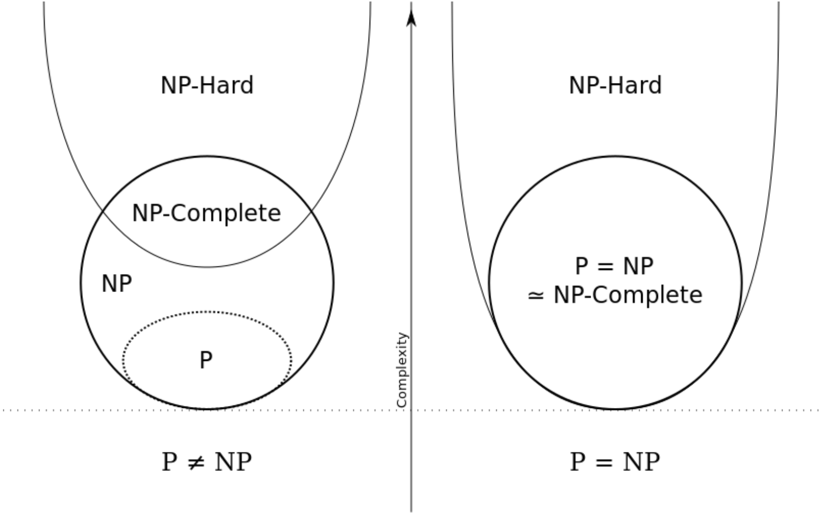
NP-Complete

Problem X is NP-Complete if:

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

Million Dollar Question: P vs NP

1 of 7 Clay Mathematics Institute Millennium Prize Problems



$P \neq NP$ $\qquad$ $P = NP$

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 $X \in$ NP-Complete. Then, X is solvable in polynomial time iff P = NP.

Hardest NP Problems

NP-Complete

Problem X is NP-Complete if:

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

Theorem 6

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

Proof.

$\Leftarrow$: Suppose $\mathrm{P} = \mathrm{NP}$, 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 $Y \in \mathrm{NP}$, $Y \leq_p X$.
- X is in NP.

Theorem 6

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

Proof.

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

FIRST NP-COMPLETE PROBLEM

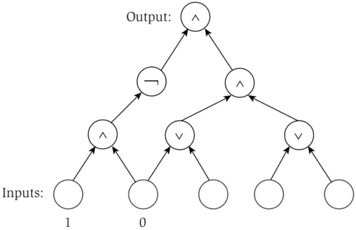
Theorem 6

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

Stephen Cook (1968)

Leonid Levin (2010)

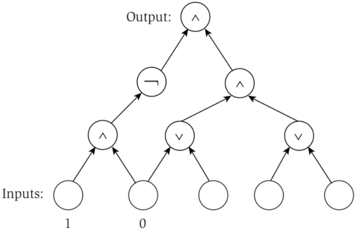
Circuit Satisfiability Problem (CSAT)



Problem Definition

- 3 types of gates: $\wedge$ (AND), $\vee$ (OR), and $\neg$ (NOT).

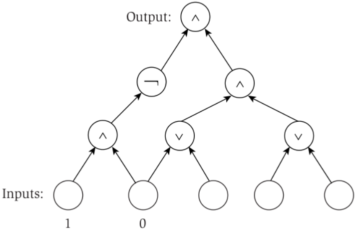
Circuit Satisfiability Problem (CSAT)



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).

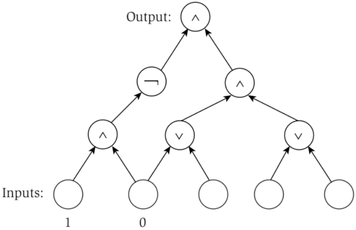
CIRCUIT SATISFIABILITY PROBLEM (CSAT)



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.

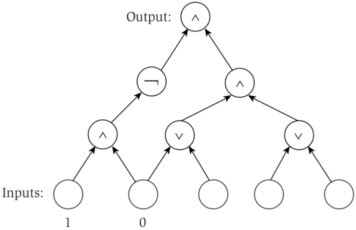
CIRCUIT SATISFIABILITY PROBLEM (CSAT)



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.

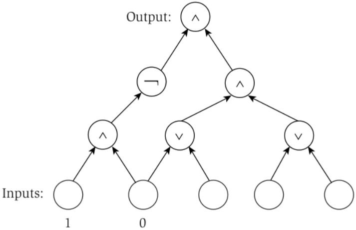
Circuit Satisfiability Problem (CSAT)



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.

Circuit Satisfiability Problem (CSAT)

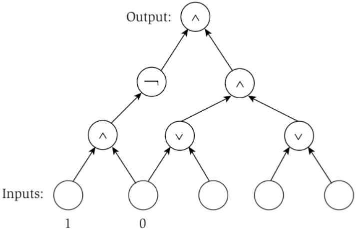


🤔: What is the output with an input of $(1, 0, 0)$?

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.

CIRCUIT SATISFIABILITY PROBLEM (CSAT)



🤔: Give an input that satisfies the example.

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 CSAT $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
    - Consider an arbitrary problem $X \in NP$.

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - We need to show $X \leq_p$ CSAT.

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - We need to show $X \leq_p$ 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 CSAT $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - Reduction to CSAT:

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - Reduction to CSAT:
     - Output is 1 when X is yes; otherwise 0.

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - Reduction to CSAT:
     - Output is 1 when X is yes; otherwise 0.
     - Sources: $|s| + |t| = n + \text{poly}(n)$ bits.

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - 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.

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 $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - 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.

First NP-Complete Problem

Theorem 6

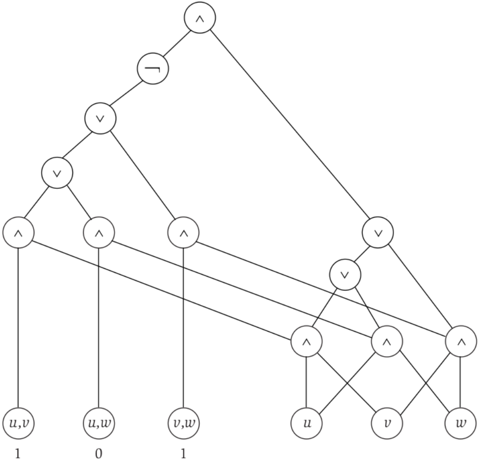
Cook (1971) – Levin (1973) Theorem [Paraphrase]: Circuit Satisfiability Problem (CSAT) *is* NP*-Complete.*

Partial Proof.

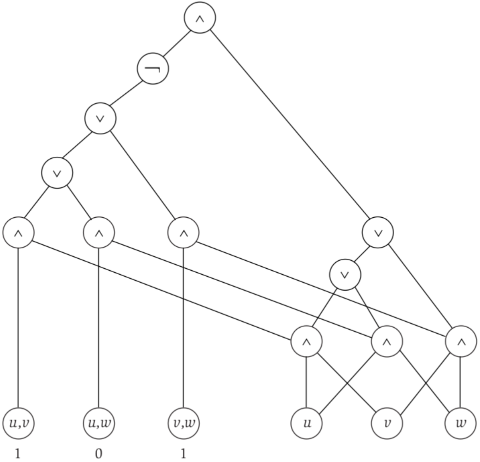
1. Show that CSAT $\in NP$:
2. Reduce every problem $\in NP$ to CSAT:
   - Consider an arbitrary problem $X \in NP$.
   - Reduction to CSAT:
     - Output is 1 when X is yes; otherwise 0.
     - Sources: $|s| + |t| = n + \text{poly}(n)$ bits.
     - The first n bits are hard-coded to the X instance input.
     - 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.



Example: Independent Set ($k \geq 2$) as Circuit Satisfiability Problem.



🤔: Draw the underlying Independent Set graph.

Strategies for Proving NP-Completeness

Showing that Problem X is NP-Complete

Cook Reduction:

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

STRATEGIES FOR PROVING NP-COMPLETENESS

Showing that Problem X is NP-Complete

Cook Reduction:

1. Prove that $X \in$ NP.
2. Choose a problem $Y \in$ 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.*

3SAT is NP-Complete

Theorem 7

3SAT *is* NP-*Complete.*

Exercise: Do step 1.

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.

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.

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:

3SAT is NP-Complete

Proof.

3. For an arbitrary circuit k:
   - Each node v is assigned a variable x_v.

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 $x_u = \overline{x_v}$.
          $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.

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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 3 clauses: $(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$.

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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 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 $x_v = x_u \wedge x_w$.
       $\rightarrow$ 3 clauses: $(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$.

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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 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 $x_v = x_u \wedge x_w$.
       $\rightarrow$ 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:
     $\rightarrow$ 1 clause: (x_s) if 1, and $(\overline{x_s})$ if 0.

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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 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 $x_v = x_u \wedge x_w$.
       $\rightarrow$ 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:
     $\rightarrow$ 1 clause: (x_s) if 1, and $(\overline{x_s})$ if 0.
   - For the output o: 1 clause (x_o).

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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 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 $x_v = x_u \wedge x_w$.
       $\rightarrow$ 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:
     $\rightarrow$ 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 $x_u = \overline{x_v}$.
       $\rightarrow$ 2 clauses: $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$.
     - OR: Let u, w be the inputs. We need $x_v = x_u \vee x_w$.
       $\rightarrow$ 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 $x_v = x_u \wedge x_w$.
       $\rightarrow$ 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:
     $\rightarrow$ 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_{CSAT} is a yes iff s_{3SAT} is a yes:
   - $\Rightarrow$: If s_{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 value will satisfy all the clauses of s_{3SAT}.
   - $\Leftarrow$: If s_{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.

$\square$

3SAT is NP-Complete

From our previous reductions

$$3SAT \leq_p IS \leq_p VC \leq_p SC$$

and

$$3SAT \leq_p IS \leq_p SP$$

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)

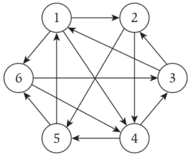
- A salesperson must visit n cities $v_1, v_2, \ldots, v_n$.
- Starting at some v_1, visit all cities and return to v_1.
- Distance function: $d(\cdot, \cdot)$ for all pairs of cities (not necessarily symmetric nor metric).
- Optimization: What is the shortest tour?

SEQUENCING PROBLEMS

Travelling Salesperson Problem (TSP)

- A salesperson must visit n cities $v_1, v_2, \ldots, v_n$.
- 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).
- Decision: Is there a tour of length D?

SEQUENCING PROBLEMS



Travelling Salesperson Problem (TSP)

- A salesperson must visit n cities $v_1, v_2, \ldots, v_n$.
- 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).

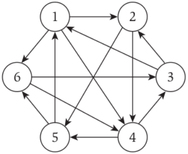
Hamiltonian Cycle

- 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 $v_1, v_2, \ldots, v_n$.
- 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).



: 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.

3SAT $\leq_p$ Hamiltonian

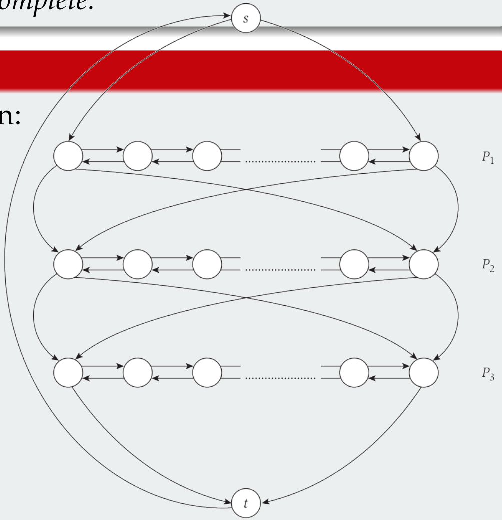
Theorem 8

Hamiltonian Cycle is NP-*complete.*

Proof.

3. 3SAT $\leq_p$ Hamiltonian:

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



3SAT $\le_p$ Hamiltonian

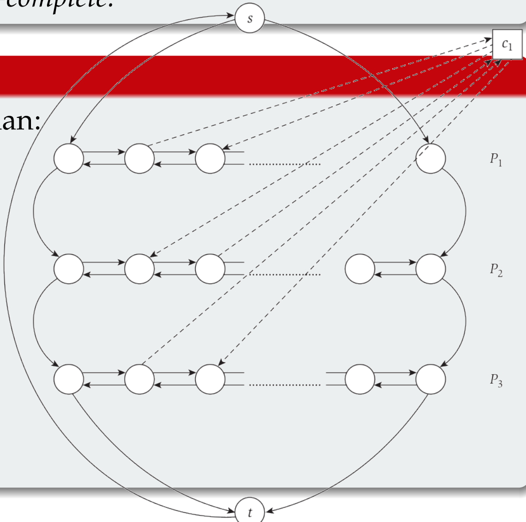
Theorem 8

Hamiltonian Cycle is NP*-complete.*

Proof.

3. 3SAT $\le_p$ Hamiltonian:

- C_i for each clause i: Connect based on x_i or $\overline{x_i}$.



3SAT $\leq_p$ Hamiltonian

Theorem 8

Hamiltonian Cycle is NP*-complete.*

Proof.

3. s_{3SAT} is a yes iff $s_{\text{Hamiltonian}}$ is a yes:
   - $\Rightarrow$: If s_{3SAT} is a yes, then each clause node can be visited from one of the paths corresponding to one of the variables when the path is traversed in the direction of the satisfying assignment.
   - $\Leftarrow$: If $s_{\text{Hamiltonian}}$ is a yes, then every clause node is visited, and the direction of each path traversal gives a value assignment for the corresponding variable in s_{3SAT}. The reduction guarantees that value assignment for a variable the path used to traverse the clause node will be the assignment of a variable that satisfies the corresponding clause.

$\square$

Travelling Salesperson

Theorem 9

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

Proof.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. 🤔Which *NP*-complete problem?

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. Hamiltonian Cycle $\leq_p$ TSP:
   Exo: Try to come up with the reduction.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. Hamiltonian Cycle $\leq_p$ TSP:
   Given a graph $G = (V, E)$:

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. Hamiltonian Cycle $\leq_p$ TSP:
   Given a graph $G = (V, E)$:
   - For each v, make a city.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. 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.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. 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 $d(u, v) = 2$.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. 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.

TRAVELLING SALESPERSON

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. 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.
   - $\Rightarrow$ With a Hamiltonian Cycle in G, the shortest tour will be length n.

Travelling Salesperson

Theorem 9

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

Proof.

1. In NP: Certificate that is a tour of the cities.
2. Use Hamiltonian Cycle.
3. 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.
   - $\Leftarrow$ If the shortest tour is length n, then no $d(u, v) = 2$ is used, so only edges from the graph are used implying a Hamiltonian cycle in G.

□

Exercise: Show that Hamiltonian Path is NP-complete

Hamiltonian Path

- A simple path in a digraph G that contains all nodes.
- Another sequencing problem.

Exercise: Show that Hamiltonian Path is NP-complete

Theorem 10

Hamiltonian Path is NP*-complete*

Proof.

1. In NP: Certificate is a path in G which can be verified in polynomial time.
2. 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.

3. Hamiltonian Cycle $\leq_p$ Hamiltonian Path:
   For $G = (V, E)$ create G':
   - Choose an arbitrary $v \in V$: $V' = V \setminus \{v\} \cup \{v', v''\}$.

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':
   - 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 $(u, v) \in E$: $E' \setminus \{(u, v)\} \cup \{(u, v'')\}$.

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':
   - 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 $v' \to v''$ means Hamiltonian Cycle.

$\square$

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.

1. In NP: Certificate is a set of trebles which can be verified in polynomial time.
2. 🤔Which NP-complete problem?

3-D MATCHING IS NP-COMPLETE

Theorem 11

3-D Matching is NP*-Complete.*

Proof.

1. In NP: Certificate is a set of trebles which can be verified in polynomial time.
2. Use 3SAT.

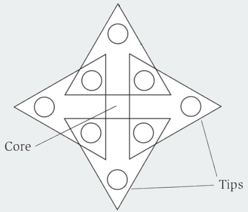
3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP*-Complete.*

Proof.

3. $3SAT \leq_p$ 3-D Matching:
   Consider an arbitrary 3SAT:
   - Variable x_i gadget:
     - Core:
       $A_i = \{a^i_1, \ldots, a^i_{2k}\}$.
     - Tips:
       $B_i = \{b^i_1, \ldots, b^i_{2k}\}$.
     - $t^i_j = (a^i_j, a^i_{j+1}, b^i_j)$ for $j = 1, 2, \ldots, 2k$ (add mod $2k$).



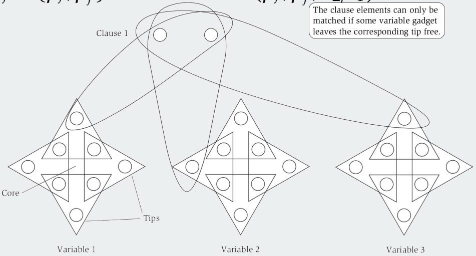
3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP-*Complete.*

Proof.

3. 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^i_{2i-1}) if x_i and (p_j, p'_j, b^i_{2j}) if $\overline{x_i}$.



3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP*-Complete.*

Proof.

3. 3SAT $\leq_p$ 3-D Matching:
   Consider an arbitrary 3SAT:
   - Counting cores: covered by even/odd choice.

3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP*-Complete.*

Proof.

3. 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.

3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP-Complete.

Proof.

3. 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.

3. 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.
   - What are the 3 sets?
     $X = \{a^i_{j\text{ even}}\} \cup \{p_j\} \cup \{q_i\}, Y = \{a^i_{j\text{ odd}}\} \cup \{p_j'\} \cup \{q_i'\}, Z = \{b^i_j\}$.

3-D Matching is NP-Complete

Theorem 11

3-D Matching is NP*-Complete.*

Proof.

3. 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 match to a clean-up gadget.

3-D Matching is NP-Complete

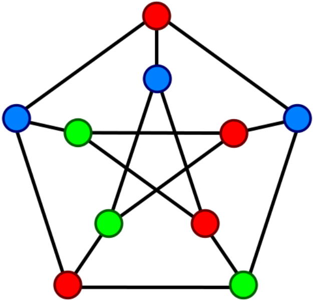
Theorem 11

3-D Matching is NP-*Complete.*

Proof.

3. 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 match 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.

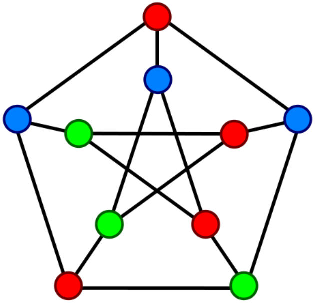
Graph Colouring



Problem

Given a graph G and a bound k, does G have a k-colouring?

Graph Colouring



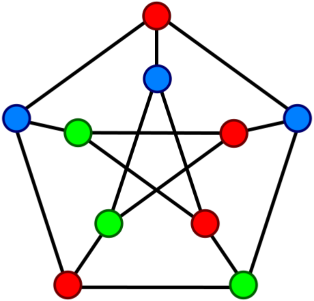
Problem

Given a graph G and a bound k, does G have a k-colouring?

***k*-Colour**

- Colouring of the nodes of a graph such that no adjacent nodes have the same colour, using at most k colours.

Graph Colouring



Problem

Given a graph G and a bound k, does G have a k-colouring?

k-Colour

- Colouring of the nodes of a graph such that no adjacent nodes have the same colour, using at most k colours.
- Labelling (partitioning) function $f : V \to \{1, \ldots, k\}$ such that, for every $(u, v) \in E, f(u) \neq f(v)$.

3-COLOURING IS NP-COMPLETE

Theorem 12

3-Colouring is NP*-Complete.*

Proof.

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. 🤔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.

3. 3SAT $\leq_p$ 3 Colouring:

3-COLOURING IS NP-COMPLETE

Theorem 12

3-Colouring is NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ 3 Colouring:
   - For each literal: Nodes v_i and $\overline{v_i}$.

3-COLOURING IS NP-COMPLETE

Theorem 12

3-Colouring is NP*-Complete.*

Proof.

3. 3SAT $\leq_p$ 3 Colouring:
   - For each literal: Nodes v_i and $\overline{v_i}$.
   - Nodes T (true), F (false), and B (base).

3-COLOURING IS NP-COMPLETE

Theorem 12

3-Colouring is NP*-Complete.*

Proof.

3. 3SAT $\leq_p$ 3 Colouring:

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

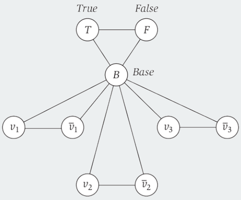
3-COLOURING IS NP-COMPLETE

Theorem 12

3-Colouring is NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ 3 Colouring:



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

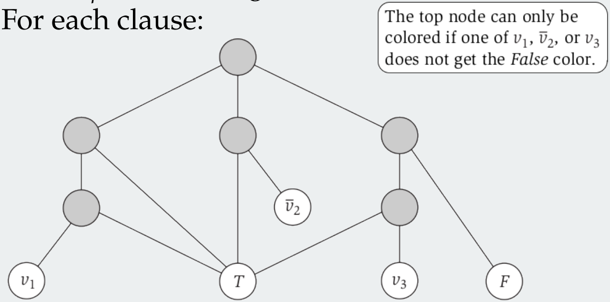
3-Colouring is NP-Complete

Theorem 12

3-Colouring is NP*-Complete.*

Proof.

3. 3SAT $\leq_p$ 3 Colouring:
For each clause:



Numerical Problems

Subset Sum Problem

Given a set of n natural numbers $\{w_1, \ldots, w_n\}$ and a target W, is there a subset of the numbers that add up to W?

Numerical Problems

Subset Sum Problem

Given a set of n natural numbers $\{w_1, \ldots, w_n\}$ and a target W, is there a subset of the numbers that add up to W?

Dynamic Programming Approach

- We saw an $O(nW)$ algorithm.
- Pseudo-polynomial: W is unbounded, e.g., 2^n.

Subset Sum is NP-Complete

Theorem 13

Subset Sum is NP*-Complete.*

Proof.

Subset Sum is NP-Complete

Theorem 13

Subset Sum is NP*-Complete.*

Proof.

1. In NP: Certificate is a subset of the numbers which can be verified in polynomial time.
2. 🤔Which NP-complete problem?

Subset Sum is NP-Complete

Theorem 13

Subset Sum is NP*-Complete.*

Proof.

1. In NP: Certificate is a subset of the numbers which can be verified in polynomial time.
2. NP-complete problem: 3-D Matching.

Subset Sum is NP-Complete

Theorem 13

Subset Sum is NP-*Complete.*

Proof.

3. 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.

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.

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, $w_t = d^{i-1} + d^{n+j-1} + d^{2n+k-1}$.

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.

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.

□

Constraint Satisfaction Problems

Not All Equal 4SAT (NAE 4SAT)

Given a 4SAT formula, is there an assignment to the literals such that every clause contains at least one true term and at least one false term.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

1. In NP:
2.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

1. In NP: Certificate is an assignment of values to the literals which can be verified in polynomial time.
2.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

1. In NP: Certificate is an assignment of values to the literals which can be verified in polynomial time.
2. NP-complete problem: 3SAT.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:

NAE 4SAT IS NP-COMPLETE

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

NAE 4SAT is NP-Complete

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

   Reduction correctness:

NAE 4SAT IS NP-COMPLETE

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

   Reduction correctness:
   - $\Rightarrow$: If 3SAT Φ is true, $\exists$ an assignment where every clause in Φ has ≥ 1 true value. Set $v = 0$ and Φ' is satisfied.

NAE 4SAT IS NP-COMPLETE

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

   Reduction correctness:
   - $\Rightarrow$: If 3SAT Φ is true, $\exists$ an assignment where every clause in Φ has ≥ 1 true value. Set $v = 0$ and Φ' is satisfied.
   - $\Leftarrow$: If NAE 4SAT is true:

NAE 4SAT IS NP-COMPLETE

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

   Reduction correctness:
   - $\Rightarrow$: If 3SAT Φ is true, $\exists$ an assignment where every clause in Φ has ≥ 1 true value. Set $v = 0$ and Φ' is satisfied.
   - $\Leftarrow$: If NAE 4SAT is true:
     - Case 1: $v = 0$. Each clause in Φ' has at least 1 term that is not v set to true $\implies$ Φ is satisfied.

NAE 4SAT IS NP-COMPLETE

Theorem 14

NAE 4SAT *is* NP-*Complete.*

Proof.

3. 3SAT $\leq_p$ NAE 4SAT:
   - Add a literal v to every 3SAT clause of Φ to create a NEA – 4SAT formula Φ'.

   Reduction correctness:
   - $\Rightarrow$: If 3SAT Φ is true, $\exists$ an assignment where every clause in Φ has ≥ 1 true value. Set $v = 0$ and Φ' is satisfied.
   - $\Leftarrow$: If NAE 4SAT is true:
     - Case 1: $v = 0$. Each clause in Φ' has at least 1 term that is not v set to true $\implies$ Φ is satisfied.
     - Case 2: $v = 1$. Each clause in Φ' has at least 1 term that is not v set to false $\implies$ Φ is satisfied by the complement of the assignment that satisfies Φ'.

$\square$

Taxonomy of Hard Problems

Packing Problems

- **Independent Set**
- Set Packing
- **Clique** (in discussion)

Covering Problems

- **Vertex Cover**
- Set Cover

Sequencing Problems

- TSP
- Hamiltonian Cycle
- Hamiltonian Path

Taxonomy of Hard Problems

Partitioning Problems

- 3-D Matching
- Graph Colouring

Numerical Problems

- **Subset Sum**
- Knapsack

Constraint Satisfaction Problems

- **3SAT**
- *CSAT*
- *NAE 4SAT*

coNP

Asymmetry of NP

Efficient Certifier Asymmetry

Given an instance s of problem X:

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

ASYMMETRY OF NP

Efficient Certifier Asymmetry

Given an instance s of problem X:

- For any t, $B(s,t) =$ yes implies yes-instance.
- For all t, $B(s,t) =$ 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 $s \notin \overline{X}$.

Asymmetry of NP

Efficient Certifier Asymmetry

Given an instance s of problem X:

- For any t, $B(s,t) =$ yes implies yes-instance.
- For all t, $B(s,t) =$ 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 $X \in \mathrm{P}$, then $\overline{X} \in \mathrm{P}$.

coNP

Complexity Class coNP

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

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

Theorem 15

If NP $\neq$ coNP, *then* P $\neq$ NP.

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

Theorem 15

If NP ≠ coNP, *then* P ≠ NP.

Proof.

Contra-positive: Prove it!

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

Theorem 15

If NP $\neq$ coNP, *then* P $\neq$ NP.

Proof.

Contra-positive: Assume P = NP:

- $X \in \text{NP} \rightarrow X \in \text{P} \rightarrow \overline{X} \in P \rightarrow \overline{X} \in \text{NP} \rightarrow X \in \text{coNP}$.

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

Theorem 15

If NP $\neq$ coNP, *then* P $\neq$ NP.

Proof.

Contra-positive: Assume P = NP:

- $X \in \text{NP} \rightarrow X \in \text{P} \rightarrow \overline{X} \in P \rightarrow \overline{X} \in \text{NP} \rightarrow X \in \text{coNP}$.
- $X \in \text{coNP} \rightarrow \overline{X} \in \text{NP} \rightarrow \overline{X} \in \text{P} \rightarrow X \in \text{P} \rightarrow X \in \text{NP}$.

$\square$

coNP

Complexity Class coNP

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

Open Question

Does NP = coNP?

Theorem 15

If NP $\neq$ coNP, *then* P $\neq$ NP.

Open Question

Does P = NP $\cap$ coNP?

PSPACE

Beyond Time

Complexity Class PSPACE

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

Beyond Time

Complexity Class PSPACE

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

Theorem 16

$\mathrm{P} \subseteq \mathrm{PSPACE}$

Beyond Time

Complexity Class PSPACE

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

Theorem 16

P $\subseteq$ PSPACE

Theorem 17

NP $\subseteq$ PSPACE

Beyond Time

Complexity Class PSPACE

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

Theorem 16

P ⊆ PSPACE

Theorem 17

NP ⊆ PSPACE

Proof.

- For 3SAT, a bit vector can encode an assignment.
- We can try all bit vectors with one n-length vector in memory:
  - Start with 0 until $2^n - 1$, adding 1 at each iteration.

Beyond Time

Complexity Class PSPACE

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

Theorem 16

P $\subseteq$ PSPACE

Theorem 17

NP $\subseteq$ PSPACE

Proof.

- For 3SAT, a bit vector can encode an assignment.
- We can try all bit vectors with one n-length vector in memory:
  - Start with 0 until $2^n - 1$, adding 1 at each iteration.
- Since 3SAT $\in$ PSPACE and is NP-complete, for any $Y \in$ NP, $Y \leq_p$ 3SAT and solve in PSPACE. □

PROTOTYPICAL PSPACE PROBLEM

Let $\Phi(x_1, \ldots, x_n)$ be a conjunction of k disjunction of n variables (like SAT).

Prototypical PSPACE Problem

Let $\Phi(x_1, \ldots, x_n)$ be a conjunction of k disjunction of n variables (like SAT).

Quantified SAT

- $\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \Phi(x_1, \ldots, x_n)$ (Prenex normal form).
- Contingency planning.

Prototypical PSPACE Problem

Let $\Phi(x_1, \ldots, x_n)$ be a conjunction of k disjunction of n variables (like SAT).

Quantified SAT

- $\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \Phi(x_1, \ldots, x_n)$ (Prenex normal form).
- Contingency planning.

Theorem 18

QSAT *is* PSPACE-*complete.*

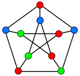
Appendix

References

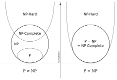
Image Sources I

https://en.wikipedia.org/wiki/Leonid_Levin

https://en.wikipedia.org/wiki/Stephen_Cook



https://en.wikipedia.org/wiki/Graph_coloring



https://en.wikipedia.org/wiki/NP-completeness#/media/File:P_np_np-complete_np-hard.svg

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

https://brand.wisc.edu/web/logos/