COMP 382: Reasoning about Algorithms

# Reductions and NP-Completeness

Prof. Maryam Aliakbarpour

**co-instructors:** Prof. Anjum Chida & Prof. Konstantinos Mamouras

November 20, 2025

# Today's Lecture

## 1. NP-Complete Problems

1.1 3-SAT is NP-complete.

1.2 Maximum Independent Set (MIS) is NP-complete

1.3 MAX-CLIQUE Is NP-Complete

1.4 VERTEX-COVER Is NP-Complete

Reading:

- Chapter 12 of the *Algorithms* book [Erickson, 2019]

Content adapted from the same reference.

# NP-Hard and NP-Complete Problems

## NP-Hard Problem

A problem $B$ is **NP-hard** if for every problem $A \in$ NP, $A$ reduces to $B$.

# NP-Hard and NP-Complete Problems

## NP-Hard Problem

A problem $B$ is **NP-hard** if for every problem $A \in \mathrm{NP}$, $A$ reduces to $B$.

## NP-Complete Problem

A problem $B$ is **NP-complete** if:

1. $B \in \mathrm{NP}$, and
2. For every problem $A \in \mathrm{NP}$, $A$ reduces to $B$.

# The NP-Completeness Recipe

To show a new problem $B$ is NP-complete, start from a known NP-complete problem $A$. Show a polynomial-time reduction from A to B.

$$\left.\begin{array}{ll} \text{A is NP-complete:} & \text{NP} \leq_p A \\ \text{A poly-time reduction from A to B:} & A \leq_p B \end{array}\right\} \implies \text{NP} \leq_p B$$

# The NP-Completeness Recipe

To show a new problem $B$ is NP-complete, start from a known NP-complete problem $A$. Show a polynomial-time reduction from A to B.

Cook-Levin Theorem

$$\left.\begin{array}{ll} \text{A is NP-complete:} & \text{NP} \leq_p A \\ \text{A poly-time reduction from A to B:} & A \leq_p B \end{array}\right\} \implies \text{NP} \leq_p B$$

# The NP-Completeness Recipe

To show a new problem $B$ is NP-complete, start from a known NP-complete problem $A$. Show a polynomial-time reduction from A to B.

Cook-Levin Theorem

A is NP-complete: $\text{NP} \leq_p A$

A poly-time reduction from A to B: $A \leq_p B$

$$\left.\begin{array}{l} \text{NP} \leq_p A \\ A \leq_p B \end{array}\right\} \implies \text{NP} \leq_p B$$

B is NP-hard.

# The NP-Completeness Recipe

To show a new problem $B$ is NP-complete, start from a known NP-complete problem $A$. Show a polynomial-time reduction from A to B.

Cook-Levin Theorem

A is NP-complete: $\text{NP} \leq_p A$

A poly-time reduction from A to B: $A \leq_p B$

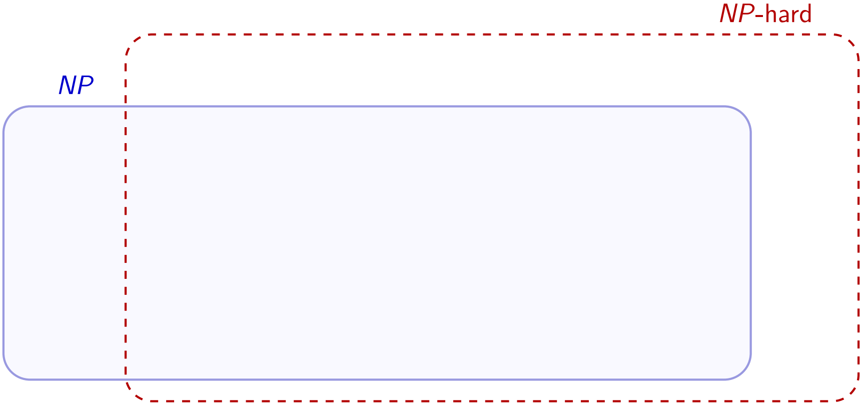
$\implies \text{NP} \leq_p B$

B is NP-hard.

If we also show that B is in NP, then $\implies$ B is NP-complete

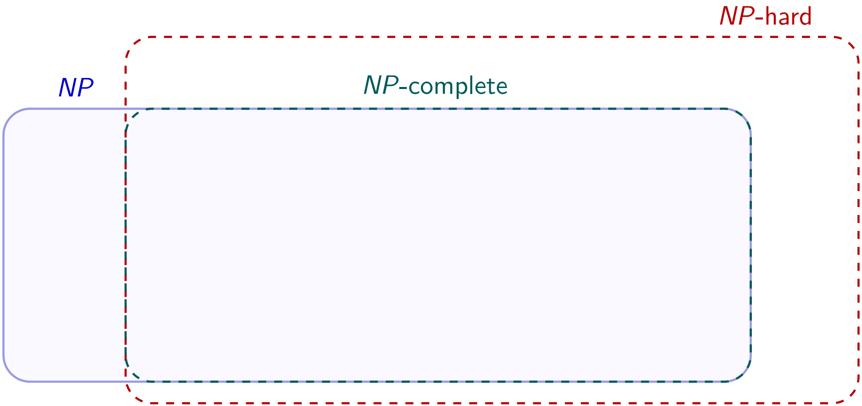
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



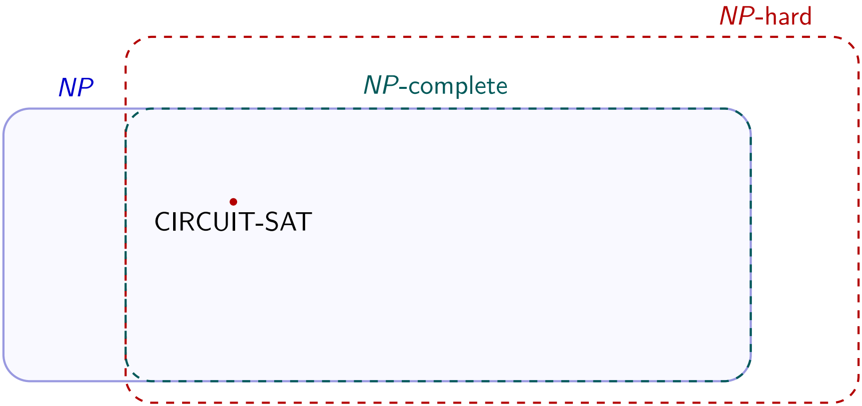
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



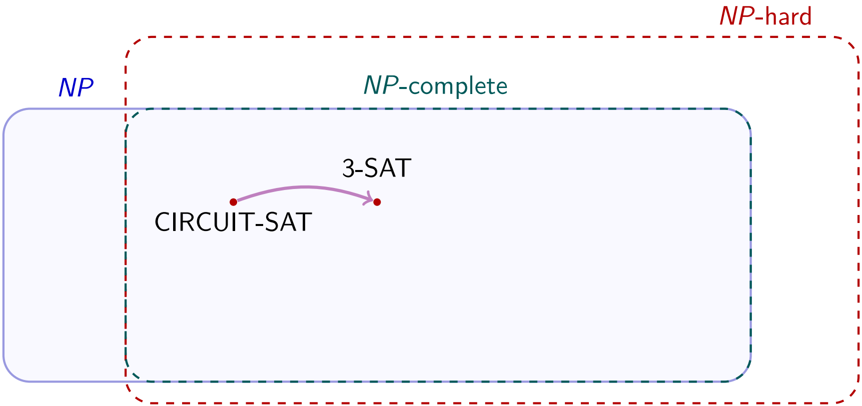
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# 3-SAT is NP-complete.

A Reduction from CIRCUIT-SAT to 3-SAT

# Boolean Logic Basics

- A literal is a variable $x$ or its negation $\neg x$.

# Boolean Logic Basics

- A literal is a variable $x$ or its negation $\neg x$.
- A disjunction (a.k.a clause) is a logical OR of two or more literals:

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k.$$

# Boolean Logic Basics

- A literal is a variable $x$ or its negation $\neg x$.
- A disjunction (a.k.a clause) is a logical OR of two or more literals:

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k.$$

- A conjunction is a logical AND of literals:

$$\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_m.$$

# Boolean Logic Basics

- A literal is a variable $x$ or its negation $\neg x$.
- A disjunction (a.k.a clause) is a logical OR of two or more literals:

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k.$$

- A conjunction is a logical AND of literals:

$$\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_m.$$

- A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions.

# Boolean Logic Basics

- A literal is a variable $x$ or its negation $\neg x$.
- A disjunction (a.k.a clause) is a logical OR of two or more literals:

$$\ell_1 \lor \ell_2 \lor \cdots \lor \ell_k.$$

- A conjunction is a logical AND of literals:

$$\ell_1 \land \ell_2 \land \cdots \land \ell_m.$$

- A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions.
- A 3-CNF formula is a CNF formula in which every clause has exactly 3 literals.

$$(a \lor \neg b \lor c) \land (\neg a \lor d \lor \neg e) \land (b \lor c \lor \neg d).$$

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \vee \neg b \vee c) \ \wedge \ (\neg a \vee d \vee \neg e) \ \wedge \ (b \vee c \vee \neg d)\,.$$

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \vee \neg b \vee c) \wedge (\neg a \vee d \vee \neg e) \wedge (b \vee c \vee \neg d).$$

**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ T $\quad e =$ F

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \lor \neg b \lor c) \ \land \ (\neg a \lor d \lor \neg e) \ \land \ (b \lor c \lor \neg d) \, .$$

**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ T $\quad e =$ F

**Why it works:**

$$(\, \text{a} \lor \neg b \lor c) \ \land \ (\neg a \lor d \lor \neg e) \ \land \ (b \lor c \lor \neg d) \, .$$

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \vee \neg b \vee c) \ \wedge \ (\neg a \vee d \vee \neg e) \ \wedge \ (b \vee c \vee \neg d)\,.$$

**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ T $\quad e =$ F

**Why it works:**

$$(\, a \vee \neg b \vee c) \ \wedge \ (\, \neg a \vee d \vee \neg e) \ \wedge \ (b \vee c \vee \neg d)\,.$$

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \vee \neg b \vee c) \wedge (\neg a \vee d \vee \neg e) \wedge (b \vee c \vee \neg d).$$

**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ T $\quad e =$ F

**Why it works:**

$$(a \vee \neg b \vee c) \wedge (\neg a \vee d \vee \neg e) \wedge (b \vee c \vee \neg d).$$

# What Is 3-SAT?

**3-SAT:** Given a 3-CNF formula $\Phi$, does there exist a truth assignment that makes $\Phi$ true?

**Example:**

$$(a \lor \neg b \lor c) \land (\neg a \lor d \lor \neg e) \land (b \lor c \lor \neg d).$$

**Assignment:** $a = \text{T}$ $\quad b = \text{F}$ $\quad c = \text{T}$ $\quad d = \text{T}$ $\quad e = \text{F}$

**Why it works:**

$$(\boxed{a} \lor \neg b \lor c) \land (\boxed{\neg a} \lor \boxed{d} \lor \neg e) \land (\boxed{\neg b} \lor \boxed{c} \lor \neg d).$$

# Why Is 3-SAT Hard? (Intuition)

Clause 1

$(a \vee \neg b \vee c)$

Clause 2

$(\neg a \vee d \vee \neg e)$

Clause 3

$(b \vee c \vee \neg d)$

# Why Is 3-SAT Hard? (Intuition)

| Clause 1 | Clause 2 | Clause 3 |
| --- | --- | --- |
| $(a \vee \neg b \vee c)$ | $(\neg a \vee d \vee \neg e)$ | $(b \vee c \vee \neg d)$ |

To satisfy the first clause, we might choose

$a =$ T, $b =$ F, or $c =$ T

# Why Is 3-SAT Hard? (Intuition)

| Clause 1 | Clause 2 | Clause 3 |
| --- | --- | --- |
| $(a \lor \neg b \lor c)$ ✓ | $(\neg a \lor d \lor \neg e)$ | $(b \lor c \lor \neg d)$ |
| | $\neg a \lor d \lor \neg e$ × | |

To satisfy the first clause, we might choose

$$a = \text{T}, \qquad b = \text{F}, \text{ or} \qquad c = \text{T}$$

# Why Is 3-SAT Hard? (Intuition)

| Clause 1 | Clause 2 | Clause 3 |
|---|---|---|
| $(a \lor \neg b \lor c)$ ✓ | $(\neg a \lor d \lor \neg e)$ | $(b \lor c \lor \neg d)$ |
| | $\not{\neg a} \lor d \lor \neg e$ × | $\not{b} \lor c \lor \neg d$ × |

To satisfy the first clause, we might choose

$$a = \text{T}, \qquad b = \text{F}, \text{ or} \qquad c = \text{T}$$

Satisfying one clause (Clause 1) can break others (Clauses 2 and 3).

This tug-of-war between clauses is what makes 3-SAT difficult.

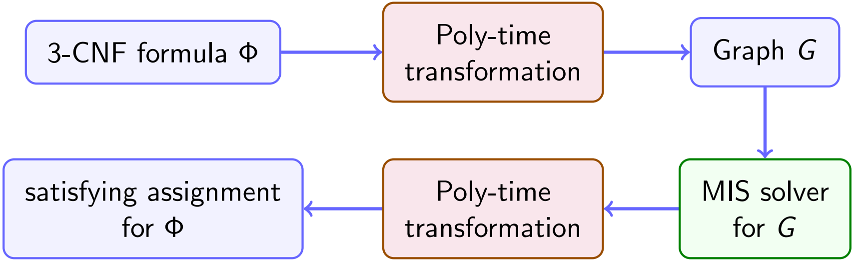
# Plan: Reduce 3-SAT to MIS

- Input on the 3-SAT side: A 3-CNF formula $\Phi$ with $k$ clauses, each with exactly three literals.

$$\Phi = C_1 \land C_2 \land \cdots \land C_k$$

- We will build a graph $G$ such that: $\Phi$ is satisfiable if and only if $G$ has an independent set of size $k$.

Reduction: 3-SAT $\leq_p$ MIS



# From CIRCUIT-SAT to 3-SAT

Goal: Show that 3-SAT is NP-complete by giving a polynomial-time reduction from Circuit-SAT to 3-SAT.


Our reduction turns an arbitrary circuit $K$ into an equivalent 3-CNF formula $\Phi_3$:

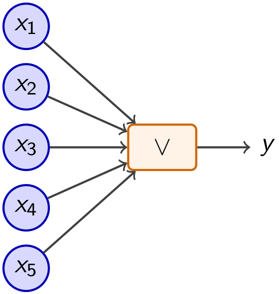
$$K \text{ is satisfiable} \iff \Phi_3 \text{ is satisfiable.}$$

# Step 1: Make the Circuit Binary

Input: an arbitrary Boolean circuit $\Phi$ built from $\wedge$, $\vee$, and $\neg$ gates.

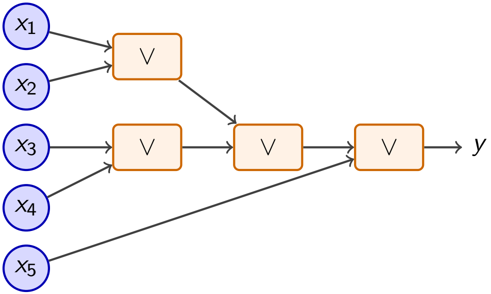
First, ensure every $\wedge$ and $\vee$ gate has exactly two inputs.

- If a gate has $k > 2$ inputs, replace it by a small binary tree of $k - 1$ binary gates.
- Call the resulting circuit $\Phi_1$.




5-input OR gate

$\Longrightarrow$

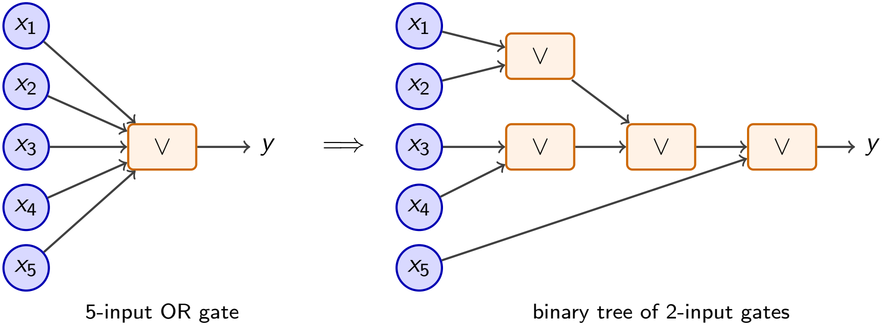



binary tree of 2-input gates

# Step 1: Make the Circuit Binary

- $\Phi$ and $\Phi_1$ are logically equivalent.
- Every satisfying input for $\Phi$ is a satisfying input for $\Phi_1$ and vice versa.

(So we can pretend from now on that every gate is binary.)



# Step 2: From Circuit to CNF $\Phi_2$

Introduce a Boolean variable for the output of every gate and input wire.

For each gate, add a constraint that relates the output variable to the input variables.

We get a formula $\Phi_2$ with one clause per gate, such that:

$$\text{assignment to inputs satisfies } \Phi_1 \iff \text{extended assignment satisfies } \Phi_2.$$

Intuitively:

- Wire variables represent the value on each wire of the circuit.
- Gate clauses enforce that each gate output is computed correctly.

# Step 2: Gate Clauses as CNF (Tseitin)

Each gate in $\Phi_1$ becomes a small CNF formula in $\Phi_2$.

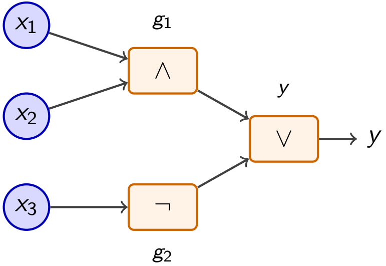
Let $a$ be the output of the gate, $b$ and $c$ be its inputs.

$$\begin{aligned}
\text{AND gate: } a = b \wedge c \quad &\Rightarrow \quad (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b) \wedge (\neg a \vee c) \\
\text{OR gate: } a = b \vee c \quad &\Rightarrow \quad (\neg a \vee b \vee c) \wedge (a \vee \neg b) \wedge (a \vee \neg c) \\
\text{NOT gate: } a = \neg b \quad &\Rightarrow \quad (a \vee b) \wedge (\neg a \vee \neg b)
\end{aligned}$$

$\Phi_1$ and $\Phi_2$ are logically equivalent, so they have exactly the same satisfying assignments.

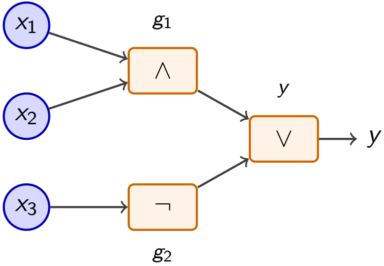
# Tseitin Encoding on a Small Circuit

Example depth-2 circuit: $y = (x_1 \wedge x_2) \vee \neg x_3.$



# Tseitin Encoding on a Small Circuit

Example depth-2 circuit: $y = (x_1 \wedge x_2) \vee \neg x_3.$

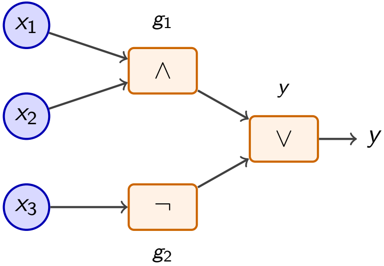



Introduce one variable per gate output:

$$g_1,\ g_2,\ y.$$

# Tseitin Encoding on a Small Circuit

Example depth-2 circuit: $y = (x_1 \wedge x_2) \vee \neg x_3.$




Introduce one variable per gate output:

$$g_1,\ g_2,\ y.$$

Tseitin constraints (CNF clauses):

$$g_1 = x_1 \wedge x_2$$
$$\Rightarrow (g_1 \vee \neg x_1 \vee \neg x_2) \wedge (\neg g_1 \vee x_1) \wedge (\neg g_1 \vee x_2)$$

$$g_2 = \neg x_3$$
$$\Rightarrow (g_2 \vee x_3) \wedge (\neg g_2 \vee \neg x_3)$$

$$y = g_1 \vee g_2$$
$$\Rightarrow (\neg y \vee g_1 \vee g_2) \wedge (y \vee \neg g_1) \wedge (y \vee \neg g_2)$$

# Step 3: Force Clauses to Have Size Exactly 3

Every clause in $\Phi_2$ has at most three literals, but 3-CNF requires exactly three.

We fix short clauses by introducing new variables.

Two-literal clause:

$$(a \vee b) \quad \Longrightarrow \quad (a \vee b \vee x) \;\wedge\; (a \vee b \vee \neg x),$$

using a new variable $x$.

One-literal clause:

$$(z) \quad \Longrightarrow \quad (z \vee x \vee y) \wedge (z \vee \neg x \vee y) \wedge (z \vee x \vee \neg y) \wedge (z \vee \neg x \vee \neg y),$$

using new variables $x, y$.

Call the final 3-CNF formula $\Phi_3$.

# Correctness of the Construction

At every step, we obtained a new formula that was logically equivalent. Thus:

$$\Phi \text{ is satisfiable} \iff \Phi_3 \text{ is satisfiable.}$$

The whole transformation runs in polynomial (in fact, linear) time in the size of $\Phi$, so we have a valid reduction from Circuit-SAT to 3-SAT.

# Correctness of the Construction

At every step, we obtained a new formula that was logically equivalent. Thus:

$$\Phi \text{ is satisfiable} \iff \Phi_3 \text{ is satisfiable.}$$

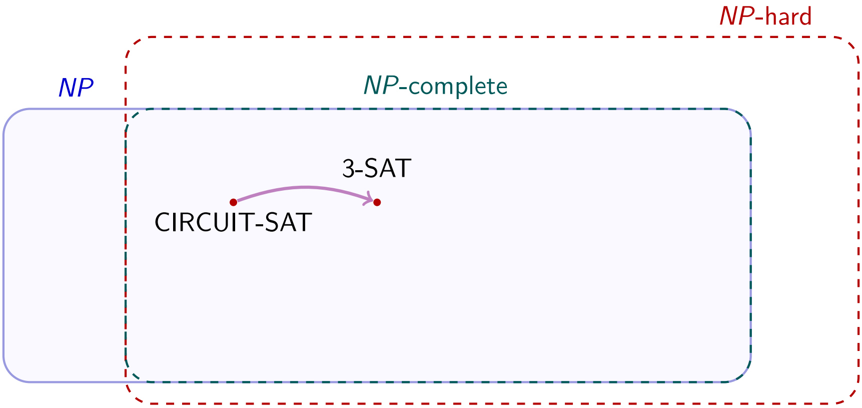
The whole transformation runs in polynomial (in fact, linear) time in the size of $\Phi$, so we have a valid reduction from Circuit-SAT to 3-SAT.

If an input assignment makes the circuit output true, that assignment serves as a polynomial-time verifiable witness for a YES-instance.

Hence CIRCUIT-SAT is in NP, and together with NP-hardness, CIRCUIT-SAT is NP-complete."

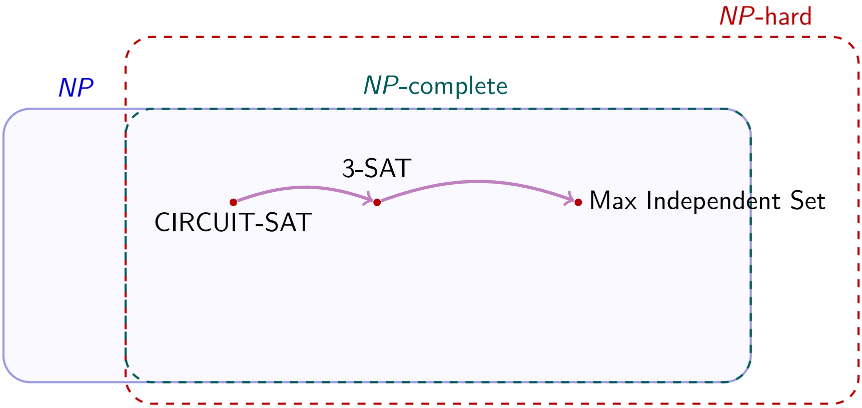
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



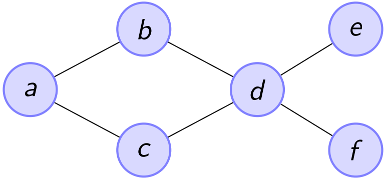
# Maximum Independent Set (MIS) is NP-complete

A reduction from 3-SAT to MIS

# Independent Sets and MIS

Let $G = (V, E)$ be a simple undirected graph.

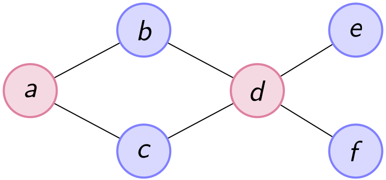
An independent set in $G$ is a subset $S \subseteq V$ such that no two vertices in $S$ are adjacent.



# Independent Sets and MIS

Let $G = (V, E)$ be a simple undirected graph.

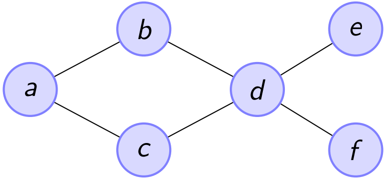
An independent set in $G$ is a subset $S \subseteq V$ such that no two vertices in $S$ are adjacent.




An independent set $S = \{a,\ d\}$ (no edges inside $S$).

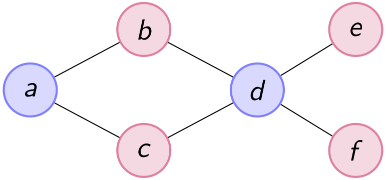
# Maximum Independent Set Problem (MIS)

**Maximum Independent Set (MIS):** Given a graph $G$ and an integer $k$, does $G$ contain an independent set of size at least $k$?



# Maximum Independent Set Problem (MIS)

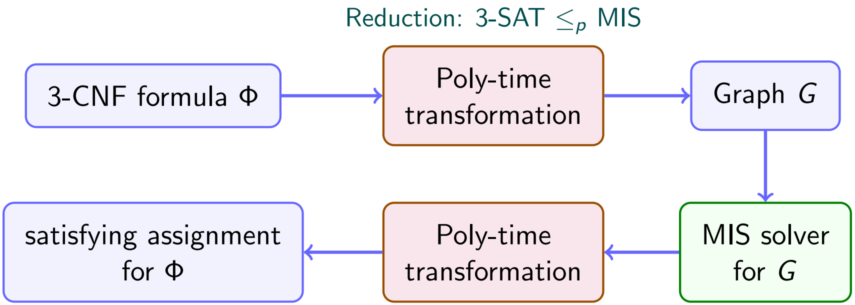
**Maximum Independent Set (MIS):** Given a graph $G$ and an integer $k$, does $G$ contain an independent set of size at least $k$?




An independent set $S = \{b,\ c,\ e,\ f\}$ with size $\geq 4$.

# Plan: Reduce 3-SAT to MIS

- Input on the 3-SAT side: A 3-CNF formula $\Phi$ with $k$ clauses, each with exactly three literals.
$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$
- We will build a graph $G$ such that: $\Phi$ is satisfiable if and only if $G$ has an independent set of size $k$.



# Constructing the Graph $G$ from $\Phi$

Let $\Phi$ be a 3-CNF formula with $k$ clauses.

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k, \qquad C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$$

# Constructing the Graph $G$ from $\Phi$

Let $\Phi$ be a 3-CNF formula with $k$ clauses.

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k, \qquad C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$$

We build a graph $G$ as follows:

1. For each clause $C_i$, create a triangle of three vertices, one for each literal $\ell_{i1}, \ell_{i2}, \ell_{i3}$.
   - The resulting graph has exactly $3k$ vertices: one vertex per literal occurrence in $\Phi$.

# Constructing the Graph $G$ from $\Phi$

Let $\Phi$ be a 3-CNF formula with $k$ clauses.

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k, \qquad C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$$

We build a graph $G$ as follows:

1. For each clause $C_i$, create a triangle of three vertices, one for each literal $\ell_{i1}, \ell_{i2}, \ell_{i3}$.
   - The resulting graph has exactly $3k$ vertices: one vertex per literal occurrence in $\Phi$.
2. For every pair of complementary literals $x$ and $\neg x$, connect all corresponding vertices in $G$ by an edge.
   - These edges enforce that we do not pick both $x$ and $\neg x$.

# Constructing the Graph $G$ from $\Phi$

Let $\Phi$ be a 3-CNF formula with $k$ clauses.

$$\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k, \qquad C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$$

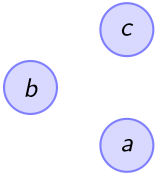
We build a graph $G$ as follows:

1. For each clause $C_i$, create a triangle of three vertices, one for each literal $\ell_{i1}, \ell_{i2}, \ell_{i3}$.
   - The resulting graph has exactly $3k$ vertices: one vertex per literal occurrence in $\Phi$.
2. For every pair of complementary literals $x$ and $\neg x$, connect all corresponding vertices in $G$ by an edge.
   - These edges enforce that we do not pick both $x$ and $\neg x$.

# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

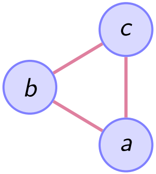
$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

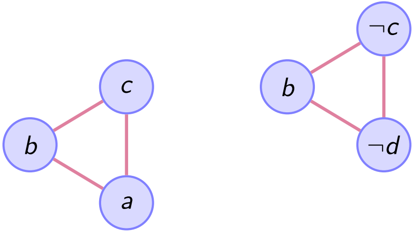
$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

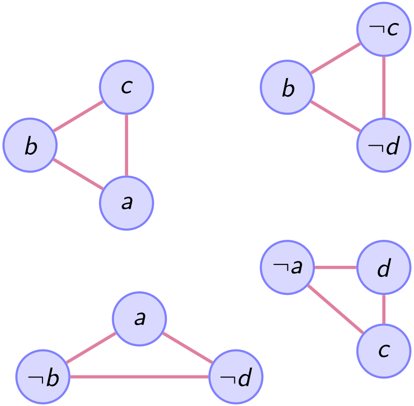
$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

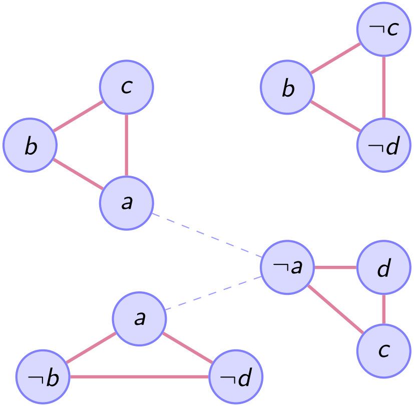
$$\Phi = (a \lor b \lor c) \land (b \lor \neg c \lor \neg d) \land (\neg a \lor c \lor d) \land (a \lor \neg b \lor \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

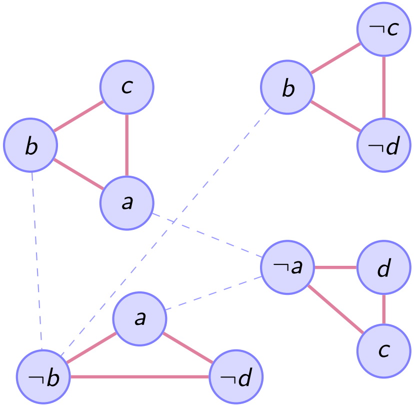
$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

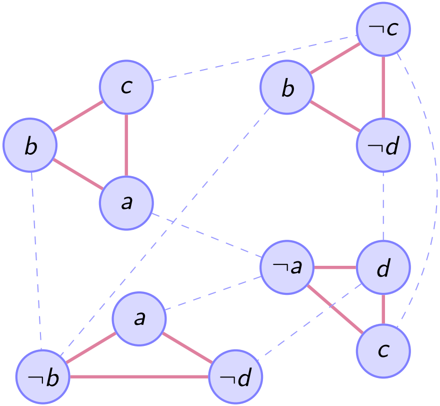
$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Example of the Construction

**Intuition:** Each selected vertex in MIS represents a literal we use to make its clause true.

$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Key Observation About Independent Sets

- Each clause corresponds to a triangle in $G$.

# Key Observation About Independent Sets

- Each clause corresponds to a triangle in $G$.
- An independent set can contain at most one vertex from each triangle, because every pair of vertices in a triangle is connected by an edge.

# Key Observation About Independent Sets

- Each clause corresponds to a triangle in $G$.
- An independent set can contain at most one vertex from each triangle, because every pair of vertices in a triangle is connected by an edge.
- Therefore, any independent set in $G$ has size at most $k$ (one vertex per clause).

# Key Observation About Independent Sets

- Each clause corresponds to a triangle in $G$.
- An independent set can contain at most one vertex from each triangle, because every pair of vertices in a triangle is connected by an edge.
- Therefore, any independent set in $G$ has size at most $k$ (one vertex per clause).
- The decision version of MIS was: $|\text{MIS}| \geq k$; The answer is YES iff $|\text{MIS}| = k$;

# Key Observation About Independent Sets

- Each clause corresponds to a triangle in $G$.
- An independent set can contain at most one vertex from each triangle, because every pair of vertices in a triangle is connected by an edge.
- Therefore, any independent set in $G$ has size at most $k$ (one vertex per clause).
- The decision version of MIS was: $|\text{MIS}| \geq k$; The answer is YES iff $|\text{MIS}| = k$;
- We will show:

$$\Phi \text{ is satisfiable} \quad \Longleftrightarrow \quad G \text{ has an independent set of size } k.$$

# Correctness: $\Phi$ is Satisfiable $\implies |\textbf{MIS}| = k$

Assume $\Phi$ is satisfiable and fix a satisfying assignment.

# Correctness: $\Phi$ is Satisfiable $\implies |\mathbf{MIS}| = k$

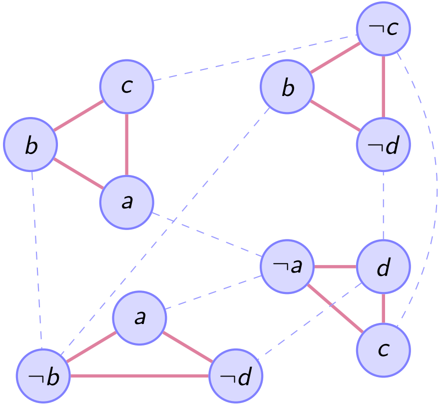
Assume $\Phi$ is satisfiable and fix a satisfying assignment.

- In each clause $C_i$, at least one literal is true under this assignment.
- For each clause $C_i$, choose exactly one vertex in the clause triangle whose literal is true.
- Let $S$ be the set of these $k$ chosen vertices.

## Correctness: $\Phi$ is Satisfiable $\implies |\text{MIS}| = k$

**3-CNF:** $\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d)$.

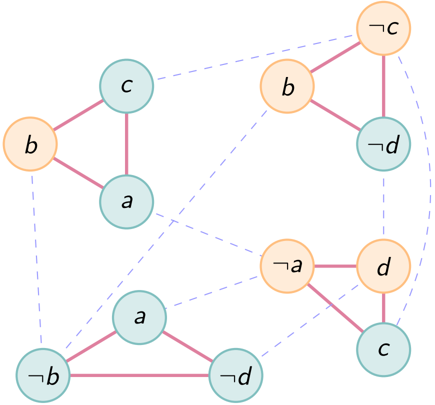
**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ F



# Correctness: $\Phi$ is Satisfiable $\implies |\text{MIS}| = k$

**3-CNF:** $\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$

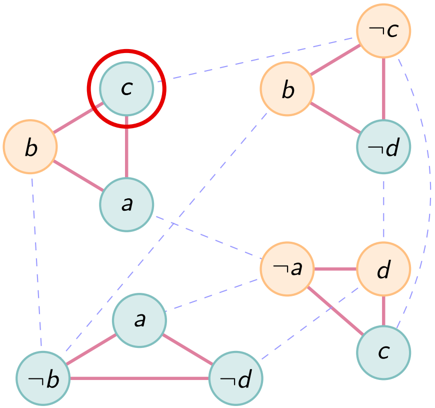
**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ F



# Correctness: $\Phi$ is Satisfiable $\implies |\mathbf{MIS}| = k$

**3-CNF:** $\Phi = (a \lor b \lor c) \land (b \lor \neg c \lor \neg d) \land (\neg a \lor c \lor d) \land (a \lor \neg b \lor \neg d)$.

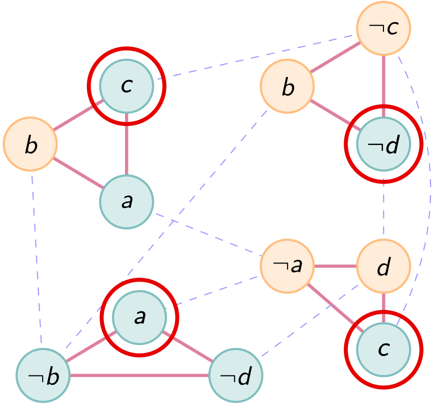
**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ F



# Correctness: $\Phi$ is Satisfiable $\implies |\text{MIS}| = k$

**3-CNF:** $\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d)$.

**Assignment:** $a =$ T $\quad b =$ F $\quad c =$ T $\quad d =$ F



# Correctness: $\Phi$ is Satisfiable $\implies |\text{MIS}| = k$

Why is $S$ an independent set?

- $S$ contains at most one vertex from each triangle, so no triangle edge connects two vertices in $S$.
- All literals in $S$ are true, so $S$ cannot contain both $x$ and $\neg x$. Hence no negation edge connects two vertices in $S$.

Thus $S$ is an independent set of size $k$ in $G$.

# Correctness: $|\textbf{MIS}| = k \implies \Phi$ is Satisfiable

Now assume $G$ has an independent set $S$ of size $k$.

# Correctness: $|\mathbf{MIS}| = k \implies \Phi$ is Satisfiable

Now assume $G$ has an independent set $S$ of size $k$.

- $S$ can contain at most one vertex from each clause triangle.
- But $|S| = k$ and there are $k$ triangles, so $S$ must contain exactly one vertex from each clause triangle.

# Correctness: $|\textbf{MIS}| = k \implies \Phi$ is Satisfiable

Now assume $G$ has an independent set $S$ of size $k$.

- $S$ can contain at most one vertex from each clause triangle.
- But $|S| = k$ and there are $k$ triangles, so $S$ must contain exactly one vertex from each clause triangle.

Use $S$ to build a truth assignment:

- For each literal in $S$, set that literal to true (that is, set the underlying variable accordingly).
- Because $S$ is independent, it never contains both $x$ and $\neg x$, so this assignment is consistent.
- Any variable not appearing in $S$ can be set arbitrarily.
- Each clause has one vertex in $S$, so each clause has at least one true literal.

# Correctness: $|\textbf{MIS}| = k \implies \Phi$ is Satisfiable

Now assume $G$ has an independent set $S$ of size $k$.

- $S$ can contain at most one vertex from each clause triangle.
- But $|S| = k$ and there are $k$ triangles, so $S$ must contain exactly one vertex from each clause triangle.

Use $S$ to build a truth assignment:

- For each literal in $S$, set that literal to true (that is, set the underlying variable accordingly).
- Because $S$ is independent, it never contains both $x$ and $\neg x$, so this assignment is consistent.
- Any variable not appearing in $S$ can be set arbitrarily.
- Each clause has one vertex in $S$, so each clause has at least one true literal.

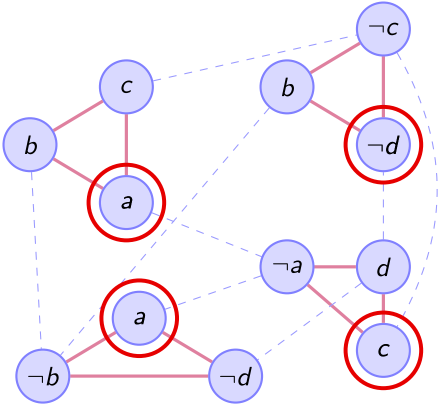
Therefore $\Phi$ is satisfiable.

# Correctness: $|\text{MIS}| = k \implies \Phi$ is Satisfiable

**Independent set:** $\{a_1, \neg d_2, c_3, a_4\}$

**Assignment:** $a =$ T $b =$ ? $c =$ T $d =$ F

$$\Phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d).$$



# Concluding the Reduction

- We transformed a 3-CNF formula $\Phi$ with $k$ clauses into a graph $G$ with $3k$ vertices.
- The transformation can be carried out in time polynomial in $|\Phi|$.
- We proved:

$$\Phi \text{ is satisfiable} \quad \iff \quad G \text{ has an independent set of size } k.$$

# Concluding the Reduction

- We transformed a 3-CNF formula $\Phi$ with $k$ clauses into a graph $G$ with $3k$ vertices.
- The transformation can be carried out in time polynomial in $|\Phi|$.
- We proved:

$$\Phi \text{ is satisfiable} \iff G \text{ has an independent set of size } k.$$

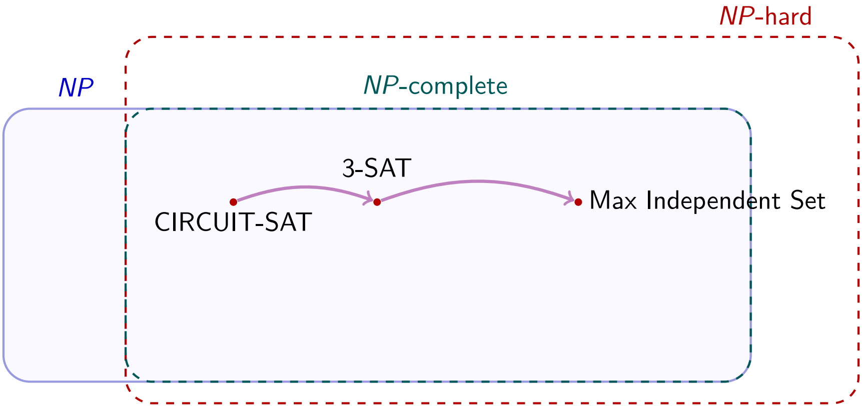
Therefore, if we could solve the MIS decision problem in polynomial time, we could solve 3-SAT in polynomial time via this reduction.

So MIS is NP-hard.

If an independent set of size at least $k$ exists, that set serves as a polynomial-time verifiable witness for a YES-instance. Hence MIS is in NP, and together with NP-hardness, MIS is NP-complete.

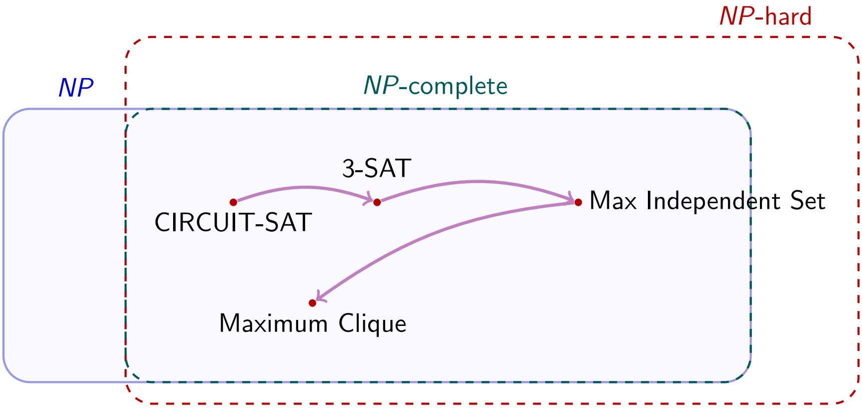
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



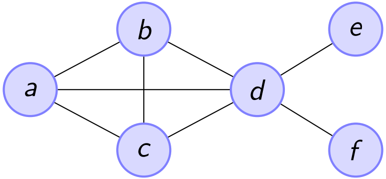
# MAX-CLIQUE Is NP-Complete

A reduction from MIS to MAX-CLIQUE

# Cliques

Let $G = (V, E)$ be a simple undirected graph.

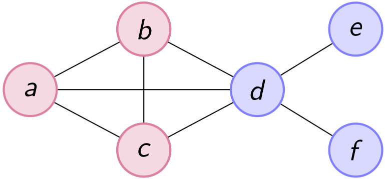
A clique in $G$ is a subset $C \subseteq V$ such that every two distinct vertices in $C$ are adjacent.



# Cliques

Let $G = (V, E)$ be a simple undirected graph.

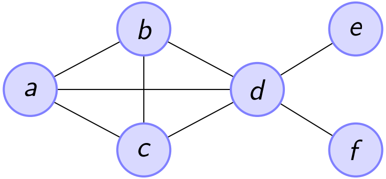
A clique in $G$ is a subset $C \subseteq V$ such that every two distinct vertices in $C$ are adjacent.




A clique $C = \{a,\ b,\ c\}$ (all vertices connected to each other).

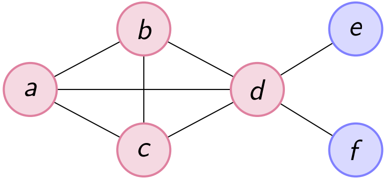
# Maximum Clique Problem

**Maximum Clique (decision):** Given a graph $G$ and an integer $k$, does $G$ contain a clique of size at least $k$?



# Maximum Clique Problem

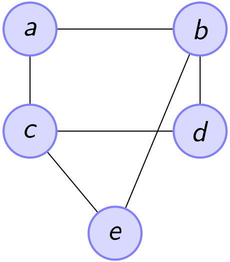
**Maximum Clique (decision):** Given a graph $G$ and an integer $k$, does $G$ contain a clique of size at least $k$?



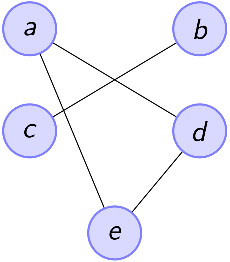

A clique $C = \{a,\ b,\ c,\ d\}$ with size $\geq 4$.

# Observation: Complement Graph

**Graph** $G$

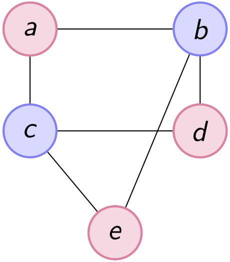



**Complement** $\overline{G}$



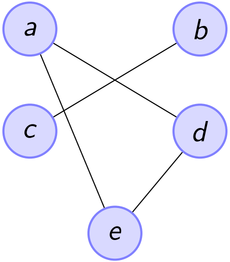
# Observation: Complement Graph

**Graph $G$**

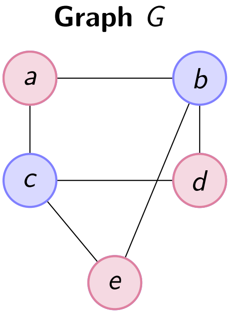



In $G$, $\{a, d, e\}$ is an independent set
(no internal edges).

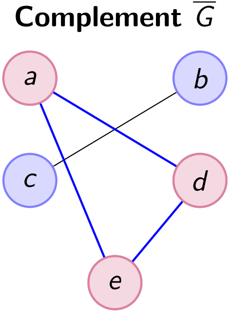
**Complement $\overline{G}$**



# Observation: Complement Graph




In $G$, $\{a, d, e\}$ is an independent set (no internal edges).

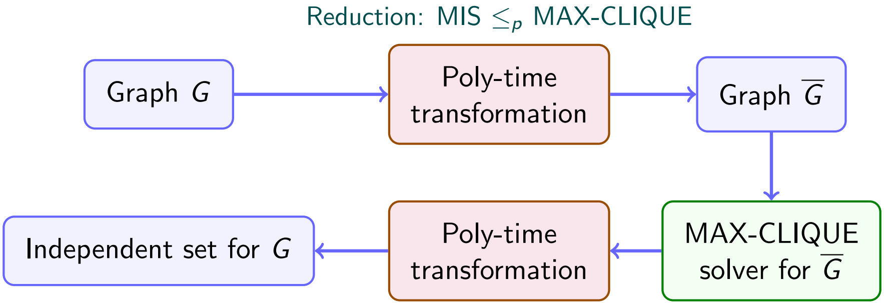



In $\overline{G}$, $\{a, d, e\}$ is a clique (triangle formed).

# Reduction from MIS to MAX-CLIQUE

Let $\overline{G} = (V, \overline{E})$ be the complement of $G = (V, E)$.

$$\text{A set } S \text{ is independent in } G \iff S \text{ is a clique in } \overline{G}.$$



# Reduction from MIS to MAX-CLIQUE

- **Mapping:** The problem of finding the maximum independent set in $G$ is *identical* to finding the maximum clique in $\overline{G}$.

# Reduction from MIS to MAX-CLIQUE

- **Mapping:** The problem of finding the maximum independent set in $G$ is *identical* to finding the maximum clique in $\overline{G}$.

- **Complexity:** We can construct $\overline{G}$ from $G$ in $O(|V|^2)$ time.

# Reduction from MIS to MAX-CLIQUE

- **Mapping:** The problem of finding the maximum independent set in $G$ is *identical* to finding the maximum clique in $\overline{G}$.

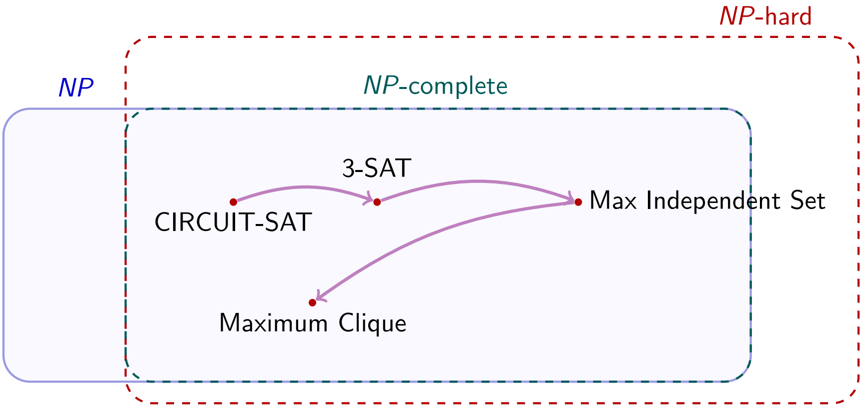
- **Complexity:** We can construct $\overline{G}$ from $G$ in $O(|V|^2)$ time.

- **MAX-CLIQUE is in NP-complete.**
  - The clique of size $k$ can serve as a witness. Thus, MAX-CLIQUE is in NP.
  - Since Maximum Independent Set (MIS) is NP-hard, Maximum Clique must also be NP-hard.

$$\text{MIS} \leq_p \text{MAX-CLIQUE}$$

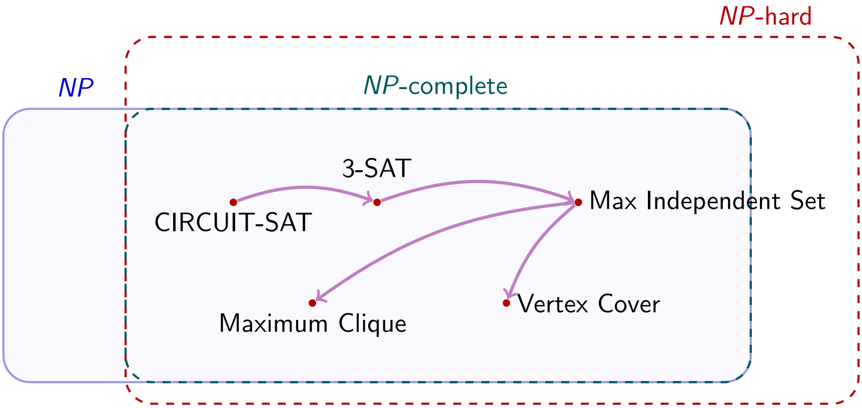
# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# A Small NP-Completeness Family Portrait

Once one natural problem is shown NP-complete, the others follow by reductions.



# VERTEX-COVER Is NP-Complete

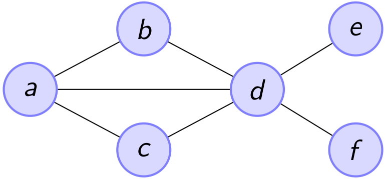
A reduction from MIS to VERTEX-COVER

# Vertex Covers

Let $G = (V, E)$ be a simple undirected graph.

A vertex cover in $G$ is a subset $C \subseteq V$ such that every edge of $G$ has at least one endpoint in $C$.

Equivalently: every edge is "touched" by $C$.

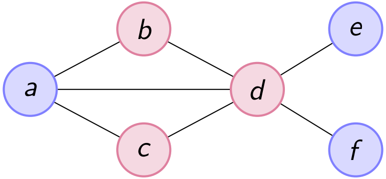


# Vertex Covers

Let $G = (V, E)$ be a simple undirected graph.

A vertex cover in $G$ is a subset $C \subseteq V$ such that every edge of $G$ has at least one endpoint in $C$.

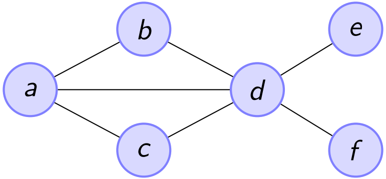
Equivalently: every edge is "touched" by $C$.




A vertex cover $C = \{b, \ c, \ d\}$ touches all edges.

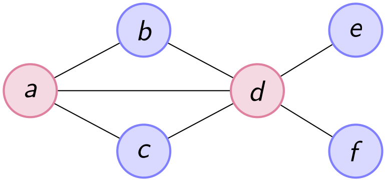
# Minimum Vertex Cover Problem

**Vertex Cover (decision):** Given a graph $G$ and an integer $k$, does $G$ contain a vertex cover of size at most $k$?



# Minimum Vertex Cover Problem

**Vertex Cover (decision):** Given a graph $G$ and an integer $k$, does $G$ contain a vertex cover of size at most $k$?

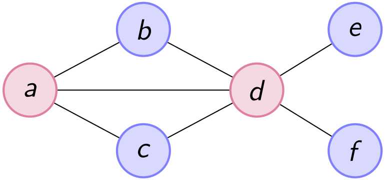



Here $\{a,\ d\}$ is a vertex cover of size 2.

# Independent Sets vs Vertex Covers

If we remove the vertices in a vertex cover from the graph, all edges disappear. The remaining vertices form an independent set.

**Vertex cover**

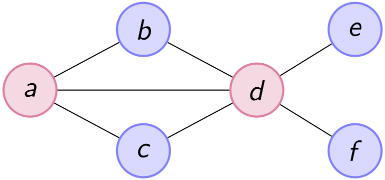



$\{a, d\}$ is a vertex cover.

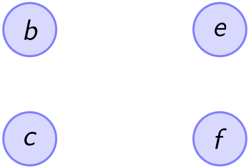
# Independent Sets vs Vertex Covers

If we remove the vertices in a vertex cover from the graph, all edges disappear. The remaining vertices form an independent set.

**Vertex cover**




$\{a, d\}$ is a vertex cover.

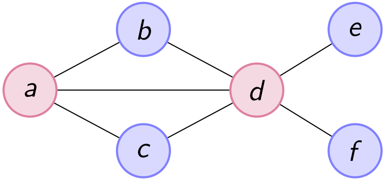



By removing $\{a, d\}$ form $G$, we obtain $\{b, \ c, \ e, \ f\}$

# Independent Sets vs Vertex Covers

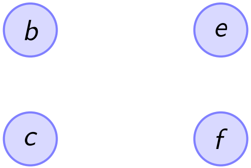
If we remove the vertices in a vertex cover from the graph, all edges disappear. The remaining vertices form an independent set.

**Vertex cover**




$\{a, d\}$ is a vertex cover.

**Independent set**




By removing $\{a, d\}$ form $G$, we obtain $\{b,\ c,\ e,\ f\}$ which is an independent set.

# Key Relationship

Let $G = (V, E)$ be a graph with $n = |V|$.

Key relationship:

$$I \text{ is an independent set in } G \iff V \setminus I \text{ is a vertex cover of } G.$$

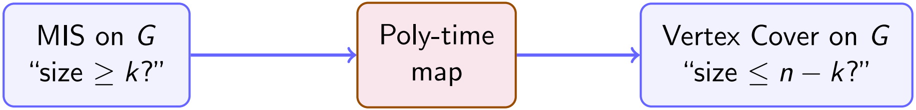
So finding a largest independent set is equivalent to finding a smallest vertex cover:

$$\max |\text{independent set}| + \min |\text{vertex cover}| = n,$$

# Reduction from MIS to VERTEX-COVER

**Mapping:** Given $(G, k)$ for MIS ("is there an independent set of size at least $k$?"), map it to $(G, n-k)$ for Vertex Cover ("is there a vertex cover of size at most $n-k$?"),

Reduction: MIS $\leq_p$ VERTEX-COVER



# VERTEX-COVER Is NP-Complete

- A vertex cover of size $\leq k$ is a polynomial-time verifiable witness, so Vertex Cover is in NP.
- Since MIS is NP-hard and MIS $\leq_p$ VERTEX-COVER, Vertex Cover is NP-complete.

# References

Erickson, J. (2019).
*Algorithms*.
Self-published.