NP-Hardness, Part 2

NP-Hard, NP-Complete, SAT, Cook-Levin, Reductions

Recap

- P: decision problems solvable in polynomial time.
- NP: decision problems whose yes-instances have certificates verifiable in polynomial time.
- $A \leq_p B$: a polynomial-time map f with $x \in A \iff f(x) \in B$.
- If $A \leq_p B$ and $B \in \mathrm{P}$, then $A \in \mathrm{P}$.

Today: the hardest problems in NP, and how to recognize one.

NP-Hard and NP-Complete

NP-Hard

Definition

A problem X is NP-hard if every problem in NP reduces to it:

$$A \leq_p X \quad \text{for every } A \in \text{NP}.$$

- X is at least as hard as every problem in NP.
- A poly-time algorithm for any NP-hard problem would give P = NP.
- Need not be in NP — can be strictly harder.

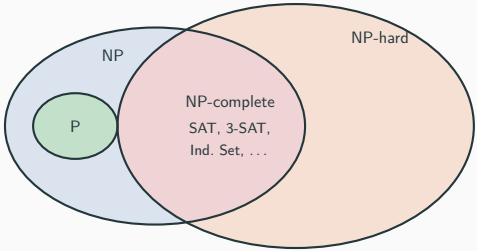
NP-Complete

Definition

X is NP-complete if

1. $X \in$ NP, and
2. X is NP-hard.

The NP-complete problems are the hardest problems *inside* NP. If any one of them is in P, then P = NP.



The picture if P $\neq$ NP.

Why Care?

NP-complete $\approx$ strong evidence no poly-time algorithm exists. Finding one would settle P vs. NP.

What to do instead:

- solve special cases (trees, small k, bounded weights),
- settle for an approximation with a guarantee,
- use heuristics that work well in practice,
- use exponential algorithms fast enough in practice (e.g. branch-and-bound, SAT solvers).

Satisfiability

SAT

- A literal is a variable x_i or its negation $\neg x_i$.
- A clause is an OR of literals: $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_m)$.
- A formula is in conjunctive normal form (CNF) if it is an AND of clauses.

SAT

- A literal is a variable x_i or its negation $\neg x_i$.
- A clause is an OR of literals: $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_m)$.
- A formula is in conjunctive normal form (CNF) if it is an AND of clauses.

SAT

Given a CNF formula ϕ, is there a true/false assignment to the variables that satisfies every clause?

Example. $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

SAT

- A literal is a variable x_i or its negation $\neg x_i$.
- A clause is an OR of literals: $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_m)$.
- A formula is in conjunctive normal form (CNF) if it is an AND of clauses.

SAT

Given a CNF formula ϕ, is there a true/false assignment to the variables that satisfies every clause?

Example. $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3)$. Setting $x_1 = x_2 = \mathrm{T}$, $x_3 = \mathrm{F}$ satisfies both clauses.

SAT

- A literal is a variable x_i or its negation $\neg x_i$.
- A clause is an OR of literals: $(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_m)$.
- A formula is in conjunctive normal form (CNF) if it is an AND of clauses.

SAT

Given a CNF formula ϕ, is there a true/false assignment to the variables that satisfies every clause?

Example. $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3)$. Setting $x_1 = x_2 = \mathrm{T}$, $x_3 = \mathrm{F}$ satisfies both clauses.

SAT $\in$ NP: certificate = satisfying assignment; verify each clause in linear time.

An Unsatisfiable Formula

$\phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$

Every truth assignment fails exactly one clause:

x_1	x_2	C_1	C_2	C_3	C_4
T	T	T	T	T	F
T	F	T	T	F	T
F	T	T	F	T	T
F	F	F	T	T	T

Another Unsatisfiable Formula

$$\phi = (c_1 \vee c_2) \wedge (\neg c_1 \vee \neg c_2) \wedge (c_1 \vee c_3) \wedge (\neg c_1 \vee \neg c_3) \wedge (c_2 \vee c_3) \wedge (\neg c_2 \vee \neg c_3)$$

c_1	c_2	c_3	first failing clause
T	T	T	$\neg c_1 \vee \neg c_2$
T	T	F	$\neg c_1 \vee \neg c_2$
T	F	T	$\neg c_1 \vee \neg c_3$
T	F	F	$c_2 \vee c_3$
F	T	T	$\neg c_2 \vee \neg c_3$
F	T	F	$c_1 \vee c_3$
F	F	T	$c_1 \vee c_2$
F	F	F	$c_1 \vee c_2$

3-SAT

3-SAT is SAT where every clause has exactly 3 literals:

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3).$$

- 3-SAT $\in$ NP, for the same reason as SAT.
- 3-SAT is a special case of SAT, so it is no harder.
- Rigid structure: every clause becomes the same small gadget — convenient for reductions.

(2-SAT, with two literals per clause, is in P, via strongly connected components!)

The Cook-Levin Theorem

Theorem (Cook 1971; Levin 1973)

SAT is NP-complete.

The Cook-Levin Theorem

Theorem (Cook 1971; Levin 1973)

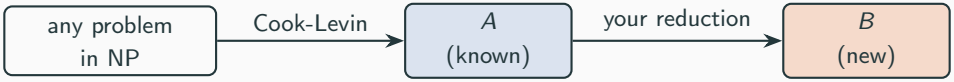
SAT is NP-complete.

Theorem (Karp 1972)

3-SAT is NP-complete.

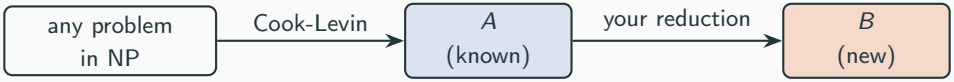
How to Prove B Is NP-Complete

1. $B \in$ NP: give a certificate and a polynomial-time verifier.
2. B **is NP-hard**: pick a known NP-complete problem A (e.g. 3-SAT) and give a reduction $A \leq_p B$.



How to Prove B Is NP-Complete

1. $B \in$ NP: give a certificate and a polynomial-time verifier.
2. B **is NP-hard**: pick a known NP-complete problem A (e.g. 3-SAT) and give a reduction $A \leq_p B$.



Why it works: reductions compose.

Direction matters: reduce *from* the known hard problem *to* the new one.

3-SAT $\leq_p$ Independent Set

Independent Set

An independent set in $G = (V, E)$ is a set $S \subseteq V$ with no edge between any two of its vertices.

Independent Set

Given G and k, does G have an independent set of size $\geq k$?

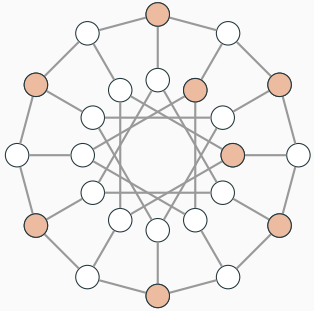
Theorem

Independent Set is NP-complete.

Step 1: in NP. Certificate: the set S. Check $|S| \geq k$ and that no edge has both endpoints in S, in $O(n + m)$ time.

Step 2: NP-hard. Reduce 3-SAT $\leq_p$ Independent Set.

Independent Set $\in$ NP: Certificate



Is there an independent set of size 8?

Certificate: the set S (orange).

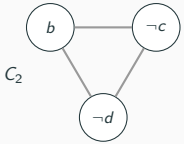
Verify: $|S| \geq 8$, and no edge has both endpoints in S. Time $O(n + m)$.

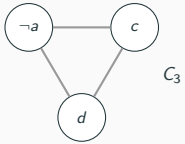
Step 1: One Vertex per Literal, Triangle per Clause

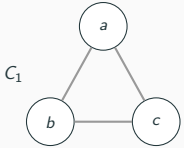
$\phi = (a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (a \vee \neg b \vee \neg d)$.

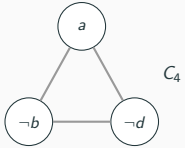
Set $k =$ number of clauses $= m$ (here $k = 4$).

For each clause, create one vertex per literal and connect them in a triangle. An independent set picks at most one vertex per triangle.



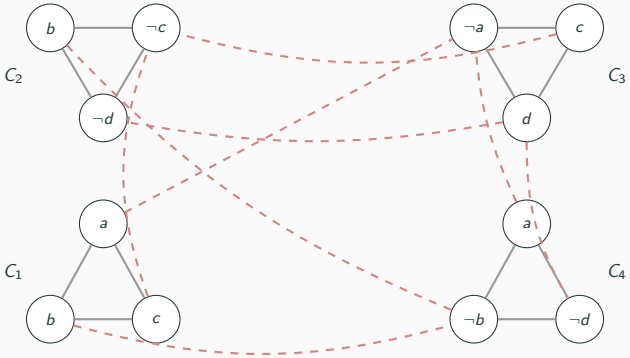






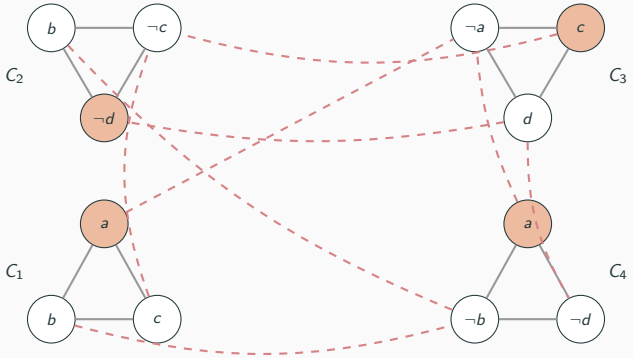
Step 2: Conflict Edges Between Complementary Literals

Add a conflict edge between every pair of vertices in *different* triangles whose literals are complementary (ℓ and $\neg\ell$). An independent set can never pick both.



Step 2: Conflict Edges Between Complementary Literals

Add a conflict edge between every pair of vertices in *different* triangles whose literals are complementary (ℓ and $\neg\ell$). An independent set can never pick both.



a=T, c=T, d=F: one true literal per clause gives an independent set of size 4 (orange).

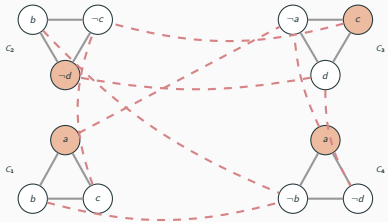
Correctness ($\Rightarrow$)

Claim. ϕ satisfiable $\Rightarrow$ G has an independent set of size k.

Take a satisfying assignment. From each clause pick one true literal's vertex; call the set S.

- $|S| = k$, one per triangle, so no triangle edge is inside S.
- A conflict edge inside S would mean ℓ and $\neg\ell$ are both true. Impossible.

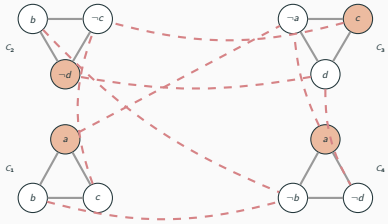
a=T, c=T, d=F: pick a from C_1, $\neg d$ from C_2, c from C_3, a from C_4.



Correctness ($\Leftarrow$)

Claim. G has an independent set of size $k \Rightarrow \phi$ is satisfiable.
Let S be an independent set of size k.

- Each triangle holds at most one vertex of S, and there are k triangles: exactly one per triangle.
- Set each literal in S to true. No conflict edge in $S \Rightarrow$ never both ℓ and $\neg\ell$ true. Set other variables freely.
- Each clause contains the true literal chosen from its triangle. $\square$



Orange vertices $\rightarrow$ set those literals true.

3-SAT $\leq_p$ Integer Linear Programming

ILP Feasibility

Given integer coefficients c_{ji} and d_j, are there integers $y_1, \ldots, y_n$ with

$$\sum_{i=1}^{n} c_{ji}\, y_i \geq d_j \quad \text{for every constraint } j?$$

Identical to LP except the variables must take integer values. That single constraint makes it NP-hard (LP is in P).

ILP: Yes Instance

$$y_1 + y_2 \geq 3$$
$$2y_1 + y_2 \geq 4$$
$$y_1,\ y_2 \in \mathbb{Z}_{\geq 0}$$

Certificate: $y_1 = 1,\ y_2 = 2$.

Check:

$$1 + 2 = 3 \geq 3 \qquad 2 \cdot 1 + 2 = 4 \geq 4$$

Both constraints satisfied. This is a yes instance.

ILP: No Instance

$$
\begin{aligned}
y_1 + y_2 &\geq 5 \\
-y_1 &\geq -2 \quad \text{(i.e. } y_1 \leq 2\text{)} \\
-y_2 &\geq -2 \quad \text{(i.e. } y_2 \leq 2\text{)} \\
y_1, y_2 &\in \mathbb{Z}_{\geq 0}
\end{aligned}
$$

Constraints 2 and 3 force $y_1 \leq 2$ and $y_2 \leq 2$, so

$$y_1 + y_2 \leq 4 < 5.$$

Constraint 1 can never be met. This is a no instance.

ILP is NP-Complete

Theorem

ILP Feasibility is NP-complete.

Step 1: in NP. Certificate: the integers y_i; verify each constraint in polynomial time.

(Subtle: feasible instances have solutions with polynomially many bits — true but non-obvious; immediate for 0/1 variables.)

Step 2: NP-hard. Reduce 3-SAT $\leq_p$ ILP.

Step 1: Encode Variables

For each Boolean variable x_i, introduce an integer y_i with

$$0 \leq y_i \leq 1, \quad y_i \in \mathbb{Z}.$$

So $y_i \in \{0, 1\}$: interpret $y_i = 1$ as true, $y_i = 0$ as false.

Example. ϕ over x_1, x_2, x_3 creates three integer variables y_1, y_2, y_3.

The assignment $x_1 = \mathit{true},\ x_2 = \mathit{false},\ x_3 = \mathit{true}$ corresponds to $y_1 = 1,\ y_2 = 0,\ y_3 = 1$.

Step 2: Encode Each Clause

For clause C_j, let P_j (N_j) be the indices of its positive (negated) literals. Add the constraint:

$$\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \;\geq\; 1.$$

The left side counts true literals in C_j; it must be ≥ 1 — at least one literal true.

Example. $C = (x_1 \vee \neg x_2 \vee x_3)$: $P = \{1, 3\}$, $N = \{2\}$, constraint:

$$y_1 + (1 - y_2) + y_3 \;\geq\; 1.$$

y_1=1, y_2=0, y_3=1: $1 + 1 + 1 = 3 \geq 1$ (*satisfied*). y_1=0, y_2=1, y_3=0: $0 + 0 + 0 = 0 < 1$ (*fails*).

The Full Construction

Given ϕ with n variables and m clauses:

1. Create integer variables $y_1, \ldots, y_n$.
2. For each clause C_j: add one constraint $\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq 1$.
3. For each i: add bounds $y_i \geq 0$ and $-y_i \geq -1$.

Total: n variables, $m + 2n$ constraints, built in polynomial time.

Correctness: y is feasible $\iff$ every clause has at least one true literal $\iff$ ϕ is satisfied. $\square$

Construction: Example

Let $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2)$.

Encode each clause:

$$C_1 = (x_1 \vee \neg x_2): \quad y_1 + (1 - y_2) \geq 1$$
$$C_2 = (\neg x_1 \vee x_2): \quad (1 - y_1) + y_2 \geq 1$$
$$C_3 = (x_1 \vee x_2): \quad y_1 + y_2 \geq 1$$

Plus integrality: $0 \leq y_1 \leq 1,\ 0 \leq y_2 \leq 1,\ y_1, y_2 \in \mathbb{Z}$.

Construction: Example

Let $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2)$.

Encode each clause:

$$C_1 = (x_1 \vee \neg x_2): \quad y_1 + (1 - y_2) \geq 1$$
$$C_2 = (\neg x_1 \vee x_2): \quad (1 - y_1) + y_2 \geq 1$$
$$C_3 = (x_1 \vee x_2): \quad y_1 + y_2 \geq 1$$

Plus integrality: $0 \leq y_1 \leq 1,\ 0 \leq y_2 \leq 1,\ y_1, y_2 \in \mathbb{Z}$.

Certificate: $y_1 = 1,\ y_2 = 1$.

$$1 + 0 = 1 \geq 1 \qquad 0 + 1 = 1 \geq 1 \qquad 1 + 1 = 2 \geq 1$$

Correctness: y feasible $\iff$ every clause has ≥ 1 true literal $\iff$ ϕ is satisfied. $\square$

Examples

Satisfiable

$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$

$$y_1 + y_2 + (1 - y_3) \geq 1$$
$$(1 - y_1) + y_2 + y_3 \geq 1$$
$$y_1 + (1 - y_2) + y_3 \geq 1$$
$$0 \leq y_i \leq 1,\ y_i \in \mathbb{Z}$$

$y_1 = y_2 = y_3 = 1$: the left sides are $2, 2, 2$. Feasible.

Unsatisfiable

$(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$

$$3y_1 \geq 1$$
$$3(1 - y_1) \geq 1$$
$$0 \leq y_1 \leq 1,\ y_1 \in \mathbb{Z}$$

The first forces $y_1 = 1$, the second $y_1 = 0$. Infeasible.

Without integrality, $y_1 = \frac{1}{2}$ works — LP is in P. Integrality is what makes ILP hard.

Exercise

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_1 \vee x_2)$$

1. Build the Independent Set instance. How many vertices, triangle edges, and conflict edges does it have, and what is k?
2. Find a satisfying assignment and the corresponding independent set.
3. Write the ILP instance.
4. A friend proves Independent Set $\leq_p$ SAT. Does that show Independent Set is NP-hard?

Solution

(a) 9 vertices, 9 triangle edges, $k = 3$. Conflict edges (6): x_1 in C_1 and C_3 each with $\neg x_1$ in C_2; x_2 in C_1 and C_3 each with $\neg x_2$ in C_2; x_3 in C_1 and C_2 each with $\neg x_3$ in C_3.

(b) $x_1 = \mathrm{T}$, $x_2 = \mathrm{F}$, $x_3 = \mathrm{T}$. Pick x_1 from C_1, $\neg x_2$ from C_2, x_1 from C_3. No conflict edge joins them.

(c) $y_1 + y_2 + y_3 \geq 1, \quad (1 - y_1) + (1 - y_2) + y_3 \geq 1, \quad (1 - y_3) + y_1 + y_2 \geq 1, \quad 0 \leq y_i \leq 1,\ y_i \in \mathbb{Z}.$

(d) No — wrong direction. It shows IS $\leq_p$ SAT (IS is no harder than SAT), not that SAT $\leq_p$ IS. For hardness we need 3-SAT $\leq_p$ Independent Set.

Summary

- **NP-hard**: every NP problem reduces to it. **NP-complete**: NP-hard and in NP.
- **Cook-Levin**: SAT is NP-complete; 3-SAT is too.
- To show B is NP-complete: $B \in$ NP, and known NP-complete $\leq_p B$.
- 3-SAT $\leq_p$ Independent Set: a triangle per clause, conflict edges between complementary literals, $k =$ number of clauses.
- 3-SAT $\leq_p$ ILP: 0/1 variables, one "at least one true literal" constraint per clause.