

## NP Completeness and Reductions

### The 3-SAT Problem

Given a logical formula of  $n$  *boolean variables*  $x_1, \dots, x_n$  put together using only conjunctions, disjunctions, and nots, determine if the formula can be satisfied.

Example:

$$\phi(x_1, x_2, x_3) = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3).$$

Is this formula  $\phi$  satisfiable? Is there a satisfying assignment?

Yes:  $x_1 = T$ ,  $x_2 = T$ ,  $x_3 = F$ .

**Definition 1.** A specific instance of a variable  $x_i$  (negated or not) in the formula is referred to as a *literal*.

Example:

$$\phi = x_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_1.$$

How many variables? 3.

How many literals? 4.

Is  $\phi$  satisfiable? No.  $x_1 \wedge \bar{x}_1$  is unsatisfiable.

**Definition 2.** A formula is in *Conjunctive Normal Form (CNF)* if it can be broken down into clauses  $C_1, \dots, C_m$  such that:

- Each clause  $C_i$  is the disjunction (OR) of literals.
- The formula is  $C_1 \wedge \dots \wedge C_m$ , the conjunction (AND) of clauses.

In  $k$ -SAT (e.g., 3-SAT), each clause contains  $k$  literals.

Example:

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4).$$

Give two satisfying assignments.

Facts about 3-SAT:

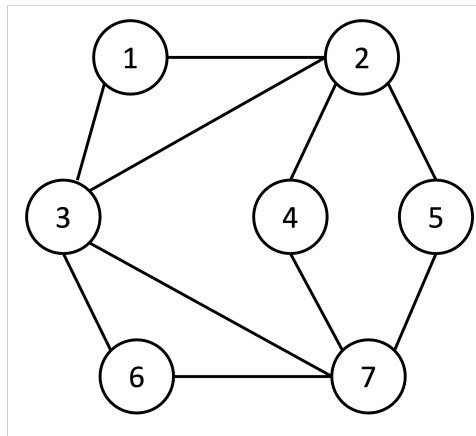
- Can represent any formula in 3-CNF
- Useful for artificial intelligence, circuit design, automatic theorem proving.
- Nobody knows how to solve it in polynomial time.
- Nobody knows how to prove that you can't, either.

Scary thing: There are thousands of problems like it.

## Independent Set

**Definition 3.** A set of vertices  $S \subseteq V$  is an *independent set* if for all  $u, v \in S$ ,  $(u, v) \notin E$ .

Give an undirected graph  $G = (V, E)$ , output an independent set of maximum size.



The maximum size independent set is of size 4:  $\{1, 4, 5, 6\}$ .

### 3-SAT $\leq_P$ Independent Set

Our goal is to show that given an algorithm to solve Independent Set (a “black box”), we could then solve 3-SAT. Hence, Independent Set is in some sense *harder* than 3-SAT.

We will do this by

- taking an instance of 3-SAT
- from it, constructing an instance for Independent Set
- showing why a solution to the Independent Set problem on this instance gives us back a solution for the 3-SAT instance in polynomial time.

This is called a polynomial-time reduction.

**Our reduction:** Given a boolean 3-CNF formula  $\phi$  with  $n$  variables,  $m$  clauses each of 3 literals, we construct the following instance of independent set:

- Construct a vertex for each literal.  
If  $v \in S$ , the independent set, set the corresponding literal to True.
- Problem: What if  $S$  includes conflicting literals?  
Solution: Add an edge between conflicting literals.  
Interpretation: We're encoding truth "consistency" in our assignment.
- Last step: connect all vertices within a clause. (Why?)  
Consequence: Every IS has  $\leq 1$  vertex per clause triangle.

What's the largest that an independent set  $S$  could be for this construction?  $m$ .

**Claim 1.**  $\phi$  is satisfiable if and only if  $\exists$  an independent set  $S$  of size  $m$ .

( $\Leftarrow$ ) Assume you have an independent set  $S$  of size  $m$ .

$S$  must contain one vertex per clause gadget. For each  $v \in S$ , set the corresponding literals to True (and set any unset variables arbitrarily).

- Every clause is satisfied: one literal is true in each.
- This truth assignment is consistent: no conflicting literals.

( $\Rightarrow$ ) Assume you have a satisfying assignment for  $\phi$ .

Each clause is satisfied by at least one literal. Take one literal per clause, and add its vertex to the independent set  $S$ . This has size  $m$  (one per clause), and is independent:

- No conflicting literals.
- Never more than one literal per clause.

Algorithm for 3-SAT:

- Construct an instance of Independent Set (a graph).
- Call a black box algorithm for IS, finding a max IS  $S$ .
- Return SAT if  $|S| = m$ , UNSAT otherwise.

# P vs. NP

## Decision Problems

**Definition 4.** A *decision problem* is an algorithmic problem where the desired output is either *yes* or *no*.

Examples:

- 3-SAT
- Independent Set: Is the max IS of size  $\geq k$ ?
- Knapsack: Is the set of value  $\geq k$ ?
- Shortest Path: Is the path from  $s$  to  $t$  of length  $\leq k$ ?

## P: “Easy to Solve”

**Definition 5.** P (polynomial time) is a complexity class of decision problems. A decision problem is in P if there is an algorithm which solves it in polynomial time.

What are some problems in P?

## NP: “Easy to Check”

Idea: If I claim to you that there’s an independent set in  $G$  with  $\geq k$  vertices, how could I convince you of that fact quickly?

What about for 3-SAT?

For all these problems, there’s some information I can give you that you can use to quickly check that there’s a yes instance. We call this information a *certificate* and the process you use to check correctness *certification*.

Formally: Given a decision problem  $A$ , a certifier algorithm ALG for  $A$ :

- Takes an input  $X$  for  $A$  and a “certificate”  $C$ .  
(“Here’s a yes input, and here’s proof that it’s a yes input.”)
- Returns True or False.  
(“Yep! Sure enough.”)
- For every input  $X$ :  
( $X$  is a yes instance)  $\Leftrightarrow$  (There’s some certificate  $C$  such that ALG returns true on  $(X, C)$ .)
  - Can always find a certificate to convince me of a yes input.
  - Can’t convince me that no inputs are actually yes inputs.

Examples:

- Independent Set (Is there an independent set of size  $\geq k$ ?)
  - **Certificate:** The set.
  - **Certifier:** Check that set has size  $k$ , is independent.
- Knapsack (Is there a set of value at least  $k$  with weight at most  $W$ ?)
  - **Certificate:** The set.
  - **Certifier:** Check that its value  $\geq k$  and weight is  $\leq W$ .
- 3-SAT (Is  $\phi$  satisfiable?)
  - **Certificate:** Variable assignment.
  - **Certifier:** Check each clause for satisfaction.

**Definition 6.** NP (non-deterministic polynomial time) is a complexity class of decision problems. A decision problem is in NP if there exists a polynomial-time certifier algorithm (that takes polynomial-size certificates).

$\Rightarrow$  all of the above problems are in NP.

We will only care about verifying “yes” instances. We don’t care if it’s hard to convince us that there’s *no* independent set of size  $\geq k$ .

Not every problem is obviously NP: “Is this 3-SAT instance unsatisfiable?”

## P vs. NP

**Claim 1:**  $P \subseteq NP$

Ex. Weighted Interval Scheduling:

- I give you an instance of WIS, and claim that there’s a schedule with weight  $\geq k$ .
- What certificate do I need to give you for you confirm this in polytime?
- Can just find WIS yourself!

To certify problems in P:

- Throw away the certificate.
- Solve the problem yourself.

Question: Does  $P = NP$ ? (Or is there a problem in NP with provably no poly-time algorithm?)

**Theorem 1** (Cook-Levin '71). *For any problem  $A \in NP$ ,  $A \leq_P 3\text{-SAT}$ .*

**Corollary 2.** *If there's a polytime algorithm for 3-SAT,  $P = NP$ .*

**Definition 7.** A problem  $B$  is *NP-hard* if for all  $A \in NP$ ,  $A \leq_P B$ .

**Definition 8.** A problem is *NP-complete* if it is NP-hard and also in NP.

**Proving a problem is NP-Complete:**

- Prove it's NP-hard (reduce from a known NP-hard problem)
- Prove it's in NP. (describe the certification algorithm)