NP-Completeness
**Objective:** Identify a class of problems that are hard to solve.

- Exponential time is hard.
- Polynomial time is easy.

**Why:** Do not try to find efficient polynomial time solutions to these problems.

- Find approximation solutions.
- Design heuristic solutions.
- Consider typical instances.
Another objective: Bundle the problems together into an equivalence class of problems.

★ Solving one of the problems in the class in polynomial time would imply a polynomial time solution to all of the other problems in the class.

★ Proving that one of the problems in the class cannot be solved in polynomial time would imply that no other problem in the class can be solved in polynomial time.
The Class of Problems $P$

**Definition:** All the problems for which there exists a polynomial time algorithm.
- $O(n^k)$-time for constant $k$ and $n \to \infty$.

**Polynomial time:** In the input size independent of the hardware and software implementation.
- Turing machines.
- Today computers.
- Parallel Computers with polynomial number of processors.
- Future computers.
\* \(O(\log n)\) for an integer \(n\).

\* In graphs with \(n\) vertices and \(m\) edges, \(\tilde{O}(n + m)\) for adjacent lists and \(\tilde{O}(n^2)\) for adjacency matrix.

**Definition:** \(\tilde{O}(f(n)) = O(f(n) \log^k n)\) for some constant \(k\).
NP stands for Non-deterministic polynomial.

**Definition I:** All the problems that can be solved by a non-deterministic polynomial time algorithm.

★ The original definition.

**Definition II:** All the problems whose solution can be verified in polynomial time.

★ A more natural definition.

**Theorem:** Both definitions are equivalent.
P vs. NP

- $P \subseteq NP$: If a problem is solvable in polynomial time its solutions are verifiable in polynomial time.

- It is strongly believed that $P \subset NP$. 
The **Halting Problem** is not in NP and definitely not in P. Actually, it is *un-solvable*.

There are many problems, though not natural problems, that are outside of NP.
Decision Problems vs. Optimization Problems

**Optimization problems:** Find a solution that optimizes – minimizes or maximizes – the objective function value.

★ Objective function values are taken from a finite ordered domain: $v_1 < v_2 < \cdots < v_h$.

**Decision problems:** Add a parameter $v$ to the input and check if there exists a solution to the original input whose value is at least $v$ (maximization problems) or at most $v$ (minimization problems).

★ A YES (TRUE) or NO (FALSE) answer.
Decision Problems vs. Optimization Problems

Optimization \Rightarrow \textbf{Decision:} Find the optimal solution and then solve the decision problem.

\textbf{Decision} \Rightarrow \textbf{Optimization:} Solve the decision problem for all possible values for the objective function.

★ Alternatively, do a binary search for the optimal value.
★ Works only if there are not too many possible values for the result.
Graph Problems

**Input:** A simple and undirected graph $G = (V, E)$.
- $V$: a set of $n$ vertices.
- $E$: a set of $m$ edges.

**Input Size:** Polynomial in both $n$ and $m$ independent on the implementation.
Independent Sets

A set $I$ of vertices is an independent set if there are no edges among them: $\forall u \neq v \in I \{(u, v) \notin E\}$. 
**Input:** A simple and undirected graph $G$ with $n$ vertices.

**Optimization Problem:** Find the independent set with the maximum size in $G$.

**Decision Problem:** Is there an independent set of size at least $K$ in $G$ for a parameter $1 \leq K \leq n$?

★ The parameter $K$ is part of the input.

**Notation:** The independent set problem is denoted by $\text{IND}$.
A set $C$ of vertices is a **clique** if all the edges among them exist: $\forall u \neq v \in C \{(u, v) \in E\}$. 
The Clique Problem

**Input:** A simple and undirected graph $G$ with $n$ vertices.

**Optimization Problem:** Find the clique with the maximum size in $G$.

**Decision Problem:** Is there a clique of size at least $K$ in $G$ for a parameter $1 \leq K \leq n$?

- The parameter $K$ is part of the input.

**Notation:** The clique problem is denoted by CLIQUE.
A set $V_C$ of vertices is a **vertex cover** if the vertices in $V_C$ cover all the edges: $\forall e \in E \exists u \in V_C \{ u \in e \}$. 

![Vertex Cover Sets Diagram]
The Vertex Cover Problem

**Input:** A simple and undirected graph $G$ with $n$ vertices.

**Optimization Problem:** Find the vertex cover with the minimum size in $G$.

**Decision Problem:** Is there a vertex cover of size at most $K$ in $G$ for a parameter $1 \leq K \leq n$?

- The parameter $K$ is part of the input.

**Notation:** The vertex cover problem is denoted by $VC$. 
A set $D$ of vertices is a dominating set if the vertices are neighbors to all other vertices: $\forall u \in V - D \exists v \in D \{(u, v) \in E\}$.
The Dominating Set Problem

**Input:** A simple and undirected graph $G$ with $n$ vertices.

**Optimization Problem:** Find the dominating set with the minimum size in $G$.

**Decision Problem:** Is there a dominating set of size at most $K$ in $G$ for a parameter $1 \leq K \leq n$?

* The parameter $K$ is part of the input.

**Notation:** The dominating set problem is denoted by $\text{DOM}$.
An assignment of colors to vertices such that neighboring vertices are assigned different colors:

$$\exists (c : V \rightarrow \{1, \ldots, \chi\}) \{ (u, v) \in E \rightarrow c(u) \neq c(v) \}.$$
The Vertex Coloring Problem

**Input:** A simple and undirected graph \( G \) with \( n \) vertices.

**Optimization Problem:** Find a coloring of \( G \) with the minimum number of colors.

**Decision Problem:** Is there a coloring of \( G \) with at most \( K \) colors for a parameter \( 1 \leq K \leq n \)?

- The parameter \( K \) is part of the input.

**Notation:** The vertex coloring problem is denoted by \( \text{COL} \).

- For \( K = 3 \) the problem is denoted by \( 3\text{-COL} \).
A simple path that visits all the vertices of the graph exactly once: $P = (v_1, v_2, \ldots, v_n) \Rightarrow V = \{v_1, \ldots, v_n\}$. 
The Hamilton Path Problem

**Input:** A simple and undirected graph $G$ with $n$ vertices.

**Optimization Problem:** Find a simple path in $G$ that visits all the vertices.

**Decision Problem:** Is there a simple path in $G$ of length $n$?

**Notation:** The Hamilton path problem is denoted by $\text{HAM}$. 
The set of all inputs of a decision problem $A$ for which the answer is **TRUE** is the language $L_A$.

The goal of a decision problem $A$ is to **identify** the language $L_A$. That is, for each input $I$ to verify if $I \in L_A$ or $I \notin L_A$.

The set NP is the set of all problems for which their language can be **verified** with a polynomial time algorithm.

The set Co-NP is the set of all problems for which their complement language can be verified with a polynomial time algorithm.
NP vs. Co-NP: Possible Relations

\[ P = NP = \text{Co-NP} \]

\[ \text{NP} \cap \text{Co-NP} \]

\[ \text{NP} \cap \text{P} \cap \text{Co-NP} \]
**Definition:** $A$ is polynomially reducible to $B$ if:

- Any input $I_A$ of $A$ can be **transformed** to an input $I_B$ of $B$ in polynomial time.
- The answer is **TRUE** to $I_B$ for problem $B$ iff the answer is **TRUE** to $I_A$ for problem $A$.

**Notation:**

- $A \rightarrow_p B$ ($A \rightarrow B$).
- $A \leq_p B$ ($A \leq B$).

**Corollary:** If $B$ is **easy** then $A$ is **easy** and if $A$ is **hard** then $B$ is **hard**.
Reducibility is Transitive

**Idea:** Polynomial times polynomial is still polynomial.

\[ x^k \cdot x^h = x^{k+h} \]

\[ k + h \text{ is constant for constants } k > 0 \text{ and } h > 0. \]

**Theorem:** \( A \rightarrow_p B \) and \( B \rightarrow_p C \) imply that \( A \rightarrow_p C \).

**Proof:** Apply first the transformation from \( I_A \) to \( I_B \) and then the transformation from \( I_B \) to \( I_C \).
Definition: Problem $A$ is an NP-Hard problem ($A \in NPH$) if all the problems in NP are polynomially reducible to $A$.

For all $B \in NP$: $B \rightarrow_p A$.

Remark: A problem in Np-Hard does not have to be in NP.
**The Class of Problems NP-Complete**

**Definition:** Problem $A$ is an **NP-Complete** problem ($A \in NPC$) if it is an NP-Hard problem and also an NP problem.

$\star \ NPC = NPH \cap NP$.

**Lemma:** If $A \in P$ and $A \in NPC$ then $P = NP$.

**Proof:** Every $B \in NP$ is polynomially reducible to $A$ and therefore can be solved with a polynomial time algorithm.

**Corollary:** If $NP = NPC$ then $P = NP$. 
**Conjecture:** It is strongly believed that $P \neq NP$. 

![Diagram of complexity classes: P, NP, NPH, NPC](image-url)
**Boolean Formulas**

**Input:** A boolean formula is an expression composed of boolean operations applied to boolean variables.

**Variables:** $x_1, x_2, \ldots, x_h$.

**Literals:** Variables and their negations: $x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_h, \overline{x}_h$.

**Operations:** $\lor$ (OR or SUM); $\land$ (AND or PRODUCT); $\otimes$ (XOR); $\ldots$

**Output:** A boolean formula can get the value **TRUE** or **FALSE** depending on the **TRUE** or **FALSE** values assigned to the variables.

**Size:** There are $h$ variables and polynomial in $h$ operations.
Example

\[
((x \otimes y) \land (\bar{z} \lor w)) \lor (\bar{w})
\]

- If \(w\) gets the value \text{FALSE} then the formula is \text{TRUE}.
- If \(w\) gets the value \text{TRUE} then the value of the formula is \(x \otimes y\).
- The value of \(z\) does not influence the value of the formula.
**Definition:** A **Conjunctive Normal Form (CNF)** boolean formula is a formula that is a product of sums of literals.

\[
\star (x \lor \bar{y} \lor \bar{z} \lor w) \land (\bar{y}) \land (z \lor w) \land (\bar{x} \lor y \lor w).
\]

**Theorem:** Every boolean formula can be represented as a CNF boolean formula.

**Proof:** The operations NOT, AND, and OR are complete.
The SAT Problem

Input: A CNF formula $\Phi$ with $h$ variables $\{x_1, \ldots, x_h\}$ and $k$ clauses: $\Phi = C_1 \land C_2 \land \cdots \land C_k$.

Problem: Find a TRUE and FALSE assignment to the $h$ variables, if exists, that satisfies the formula $\Phi$.

Decision problem: Is there a TRUE and FALSE assignment to the $h$ variables that satisfies the formula $\Phi$?

Observation: If $\Phi$ is satisfied then each clause contains at least one TRUE literal.

Notation: This problem is called SAT.
SAT is an NP-Complete Problem

★ The original first NP-Complete problem.

★ SAT is an NP-Hard problem since there exists a polynomial time reduction from any problem in NP to SAT.

★ SAT is an NP problem since it is possible to verify if a TRUE and FALSE assignment satisfies the formula $\Phi$. 
**Input:** A boolean circuit is a logic circuit composed of the gates **OR**, **AND**, and **NOT** whose inputs are the variables $x_1, x_2, \ldots, x_h$ or the outputs of other gates.

**Output:** Only one output.

**Structure:** No directed cycles.

**Size:** There are $h$ variables and polynomial in $h$ gates.
Example

\[( (x \lor y) \land (x \lor z)) \lor (\overline{x} \land (y \land \overline{z})) \]
The C-SAT Problem

**Input:** A boolean circuit with \( h \) variables \( \{x_1, \ldots, x_h\} \) and \( k \) OR, AND, and NOT gates.

**Problem:** Find a TRUE and FALSE assignment to the \( h \) variables, if exists, that produces a TRUE output to the circuit.

**Decision problem:** Is there a TRUE and FALSE assignment to the \( h \) variables that produces a TRUE output to the circuit?

**Notation:** This problem is called C-SAT.
C-SAT is an NP-Complete Problem

☆ The today common first NP-Hard problem.

☆ C-SAT is an NP-Hard problem since there exists a polynomial time reduction from any problem in NP to C-SAT.

☆ C-SAT is an NP problem since it is possible to verify in polynomial time if a TRUE and FALSE assignment produces a TRUE output to the circuit.
SAT vs. C-SAT

★ Each one of the problems is polynomially reducible to the other in a “natural” way.

★ It is “easier” to reduce all the NP problems to the C-SAT problem than to the SAT problem.
How to Prove that a Problem is NP-Complete?

⋆ Prove first that the problem is an NP problem.
⋆ Prove next that the problem is an NP-Hard problem.

Method I: Show a polynomial time reduction from any other problem in NP.

Method II: Show a polynomial time reduction from an already known NP-Hard problem.
   – Due to transitivity, all problems in NP are polynomially reducible to this problem.
The 3-SAT Problem

★ 3-SAT is SAT in which all the clauses in the formula $\Phi$ have exactly 3 literals.

★ The most “popular” NPC problem.

★ 3-SAT is an NP problem since it is possible to verify in polynomial time if a TRUE and FALSE assignment satisfies the formula $\Phi$.

★ SAT is polynomially reducible to 3-SAT and therefore 3-SAT is an NP-Hard problem.

★ 2-SAT can be solved in polynomial time.
The 3-SAT Problem

Input:

★ Variables: $x_1, x_2, \ldots, x_h$.
★ A 3-CNF formula: $\Phi = C_1 \land C_2 \land \cdots \land C_k$.
★ Clause $r$: $C_r = (\ell_{r1}, \ell_{r2}, \ell_{r3})$
★ Literals: $\ell_{ri} \in \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_h, \bar{x}_h\}$ for $1 \leq r \leq k$ and $1 \leq i \leq 3$.

Problem: Find a TRUE and FALSE assignment to the $h$ variables, if exists, such that $\Phi = \text{TRUE}$?

Decision Problem: Is there a TRUE and FALSE assignment to the variables such that $\Phi = \text{TRUE}$?
A Reduction Tree

C-SAT
  ↓
SAT
  ↓
3-SAT

VC  ↓  CLIQUE

DOM

IND

3-COL

HAM
Reduction from 3-SAT to CLIQUE

★ For a Formula $\Phi$ construct the graph $G(\Phi)$:
  - Each literal $\ell^r_i$ is a vertex for $1 \leq r \leq k$ and $1 \leq i \leq 3$.
  - Vertices $\ell^r_i$ and $\ell^s_j$ are connected by an edge if $r \neq s$ and $\ell^r_i$ is not the negation of $\ell^s_j$.

★ $G(\Phi)$ can be constructed from $\Phi$ in polynomial time. It has $3k$ vertices and at most $3k(3k - 3)/2$ edges.

Observation: Each clause is an independent set of size 3 in $G(\Phi)$. Therefore, the maximum size of a clique in $G(\Phi)$ is $k$ (number of clauses in $\Phi$).
$$\Phi = (x \lor \bar{y} \lor \bar{z}) \land (\bar{x} \lor y \lor z) \land (x \lor y \lor z)$$
**Theorem:** \( G(\Phi) \) has a clique of size \( k \) iff \( \Phi \) can be satisfied.

**Proof \( \Rightarrow \):**

- Let \( C \) be a clique of size \( k \) in \( G(\Phi) \).
- If \( x_i \in C \) then assign \( \text{TRUE} \) to \( x_i \), If \( \bar{x}_i \in C \) then assign \( \text{FALSE} \) to \( x_i \), otherwise assign \( \text{TRUE} \) or \( \text{FALSE} \) to \( x_i \),
- **Consistency:** \( x_i \) and \( \bar{x}_i \) cannot be connected by an edge and therefore cannot belong together to the clique \( C \).
- \( \Phi = \text{TRUE} \): the 3 vertices of any clause are independent set. Therefore \( C \) must contain exactly one vertex from each clause that gives this clause the \text{TRUE} value.
**Theorem:** $G(\Phi)$ has a clique of size $k$ iff $\Phi$ can be satisfied.

**Proof $\Leftarrow$:**

- Let $x_i \in \{\text{TRUE}, \text{FALSE}\}$ be a TRUE assignment to $\Phi$.
- Let $\ell_{i_1}^1, \ell_{i_2}^2, \ldots, \ell_{i_k}^k$ be the literals in the $k$ clauses that give each clause a TRUE value.
- Since the TRUE assignment is consistent, it follows that any 2 of these literals are connected by an edge to form a clique of size $k$. 

**Proof $\Rightarrow$:**

The proof of the converse direction ($\Rightarrow$) is omitted for brevity.
CLIQUE is NPC

**CLIQUE** is **NP-Hard**: 3-SAT $\rightarrow_p$ CLIQUE.

* Is there a **TRUE** assignment to the formula $\Phi$?
* Does $G(\Phi)$ have a clique of size $K = k$?

**CLIQUE** is **NP-Complete**: Can verify in polynomial time that a set of size $K$ is a clique.

For a Formula $\Phi$ construct the graph $G(\Phi)$:

- Each variable $x_i$ is represented by 2 vertices $x_i$ and $\bar{x}_i$ that are connected by an edge.
- Each clause $(\ell_1^r \lor \ell_2^r \lor \ell_3^r)$ is represented by a triangle on the 3 vertices $\ell_1^r, \ell_2^r, \ell_3^r$.
- Each vertex $\ell_j^r$ in a triangle is connected to the variable vertex that represents its value.

$G(\Phi)$ can be constructed from $\Phi$ in polynomial time. It has $2h + 3k$ vertices and $h + 6k$ edges.
Example

\[ \Phi = (x \lor y \lor z) \land (\bar{x} \lor y \lor \bar{z}) \land (\bar{y} \lor \bar{z} \lor \bar{w}) \]
Theorem: $G(\Phi)$ has a vertex cover of size $h + 2k$ iff $\Phi$ can be satisfied.

Observation: Any vertex cover in $G(\Phi)$ must contain at least $h + 2k$ vertices:

- $h$ vertices to cover the $h$ variable edges.
- $2k$ vertices to cover the $k$ triangles.
Reduction from 3-SAT to Vertex Cover

Proof \( \Rightarrow \):

★ Let \( C \) be a vertex cover of size \( h + 2k \) in \( G(\Phi) \).
★ \( C \) contains exactly one variable vertex and 2 triangle vertices.
★ If \( x_i \in C \) then assign $\text{TRUE}$ to \( x_i \). If \( \overline{x_i} \in C \) then assign $\text{FALSE}$ to \( x_i \).
★ The 2 triangle vertices can cover only 2 of the edges connecting the triangle to the variable vertices. The third edge must be covered by a variable vertex.
★ This variable vertex will give the corresponding triangle the true value.
Proof $\Leftarrow$:

- Let $x_i \in \{\text{TRUE}, \text{FALSE}\}$ be a TRUE assignment to $\Phi$.
- If $x_i = \text{TRUE}$ then add $x_i$ to the vertex cover $C$. If $x_i = \text{FALSE}$ then add $\overline{x_i}$ to the vertex cover $C$.
- Each clause has a TRUE literal. The variable vertex corresponding to this literal covers the edge connecting this clause triangle to its variables. Add the other 2 vertices from the triangle to the vertex cover $C$.
- These 2 vertices cover the triangle edges and the other 2 edges connecting the triangle to the variable vertices.
- All together, $C$ contains $h + 2k$ vertices.
VC is NP-Hard: $3$-SAT $\rightarrow_p$ VC.
- Is there a TRUE assignment to the formula $\Phi$?
- Does $G(\Phi)$ have a vertex cover of size $K = h + 2k$?

VC is NP-Complete: Can verify in polynomial time that a set of size $K$ is a vertex cover.
**Observation:** \( I \) is an independent set in a graph iff \( V - I \) is a vertex cover of the graph.
IND is NP-Hard: VC →^p IND.

★ Is there a vertex cover of size $K_{vc}$ in a graph $G$?
★ Is there an independent set of size $K_{ind} = n - K_{vc}$ in $G$?

IND is NP-Complete: Can verify in polynomial time that a set of size $K_{ind}$ is an independent set.
VC is an NP-Complete Problem

**VC is NP-Hard:** \(\text{IND} \rightarrow_p \text{VC}.

- Is there an independent set of size \(K_{ind}\) in a graph \(G\)?
- Is there a vertex cover of size \(K_{vc} = n - K_{ind}\) in \(G\)?

**VC is NP-Complete:** Can verify in polynomial time that a set of size \(K\) is a vertex cover.
Observation: $C$ is a clique in a graph iff $C$ is an independent set in the complement graph.
IND is NP-Hard: CLIQUE $\rightarrow_p$ IND.

- Is there a clique of size $K_{\text{clique}}$ in a graph $G$?
- Is there an independent set of size $K_{\text{ind}} = K_{\text{clique}}$ in the complement graph $\tilde{G}$ of the graph $G$?

IND is NP-Complete: Can verify in polynomial time that a set of size $K_{\text{ind}}$ is an independent set.
**CLIQUE** is an NP-Complete Problem

**CLIQUE is NP-Hard:** \( \text{IND} \rightarrow_p \text{CLIQUE} \).

- Is there an **independent set** of size \( K_{\text{ind}} \) in a graph \( G \)?
- Is there a **clique** of size \( K_{\text{clique}} = K_{\text{ind}} \) in the complement graph \( \tilde{G} \) of the graph \( G \)?

**CLIQUE is NP-Complete:** Can verify in polynomial time that a set of size \( K_{\text{clique}} \) is a **clique**.
**Observation:** $VC$ is a vertex cover in a graph iff $V - VC$ is a clique in the complement graph.
VC is an NP-Complete Problem

**VC is NP-Hard:** \( \text{CLIQUE} \rightarrow_p \text{VC} \).

- Is there a clique of size \( K_{\text{clique}} \) in a graph \( G \)?
- Is there a vertex cover of size \( K_{vc} = n - K_{\text{clique}} \) in the complement graph \( \tilde{G} \) of the graph \( G \)?

**VC is NP-Complete:** Can verify in polynomial time that a set of size \( K_{vc} \) is a vertex cover.
CLIQUE is an NP-Complete Problem

CLIQUE is NP-Hard: $\text{VC} \xrightarrow{p} \text{CLIQUE}$.

- Is there a vertex cover of size $K_{vc}$ in a graph $G$?
- Is there a clique of size $K_{clique} = n - K_{vc}$ in the complement graph $\tilde{G}$ of the graph $G$?

CLIQUE is NP-Complete: Can verify in polynomial time that a set of size $K_{clique}$ is a clique.
Reduction from Vertex Cover to Dominating Set

For a graph $G = (V, E)$ with $n$ vertices and $m$ edges construct the graph $G' = (V', E')$:

- All the vertices of $G$ are in $G'$: $V \subset V'$.
- All the edges of $G$ are in $G'$: $E \subset E'$.
- For an edge $e = (u, v) \in E$ add a vertex $e \in V'$.
- Add the edges $(u, e)$ and $(e, v)$ to $E'$.

$G'$ can be constructed from $G$ in polynomial time. $G'$ has $n + m$ vertices and $3m$ edges.
Example
**Theorem:** $G$ has a vertex cover of size at most $K$ iff $G'$ has a dominating set of size at most $K$.

**Observation:** Every dominating set in $G'$ must contain one of the vertices $e, u, v$ for any edge $e = (u, v) \in E$ to dominate the vertex $e \in V'$. 
**Theorem:** $G$ has a vertex cover of size at most $K$ iff $G'$ has a dominating set of size at most $K$.

**Proof $\Rightarrow$:**

★ Let the vertex cover of $G$ be the dominating set for $G'$.
★ Every vertex cover is a dominating set and therefore all the $G$ vertices in $G'$ are dominated.
★ Every edge in $G$ is covered and therefore all the new vertices are dominated.
Reduction from Vertex Cover to Dominating Set

**Theorem:** \( G \) has a vertex cover of size at most \( K \) iff \( G' \) has a dominating set of size at most \( K \).

**Proof \( \Leftarrow \):**

* Let \( D \) be a dominating set in \( G' \).

* If \( u \in V' \) is in the dominating set for \( G' \), then \( u \in V \) is in the vertex set for \( G \).

* If \( e = (u, v) \) is in the dominating set for \( G' \), then select either \( u \) or \( v \) to be in the vertex set for \( G \).

* Since every vertex \( e \in V' \) is dominated, it follows that every edge \( e \in E \) is covered.
DOM is NP-Hard: VC $\rightarrow_p$ DOM.

⋆ Is there a vertex cover of size $K_{vc}$ in a graph $G$?
⋆ Is there a dominating set of size $K_{dom} = K_{vc}$ in the graph $G'$?

DOM is NP-Complete: Can verify in polynomial time that a set of size $K_{dom}$ is a dominating set.