5. NP-Completeness

5.1. Some Variants of Satisfiability

Some Variants of Satisfiability

We have already encountered several versions of satisfiability problems:

- **intractable**: SAT, 3-SAT
- **tractable**: 2-SAT, HORNSAT

We shall encounter further intractable versions of satisfiability problems:

- restricted (but still intractable) versions of SAT
- CIRCUIT SAT
- Not-all-equal SAT (NAESAT)
- (MONOTONE) 1-IN-3-SAT
- strongly related problem: HITTING SET

Narrowing NP-complete languages

An NP-complete language can sometimes be narrowed down by transformations which eliminate certain features of the language but still preserve NP-completeness. Restricting SAT to formulae in CNF and a further restriction to 3-SAT are typical examples. Generally, \( k \)-SAT (i.e., formulae are restricted to CNF with exactly \( k \) literals in each clause) is NP-complete for any \( k \geq 3 \).

Here is another example of narrowing an NP-complete language:

**Proposition**

3-SAT remains NP-complete even if the Boolean expressions \( \varphi \) in 3-CNF are restricted such that

- each variable appears at most three times in \( \varphi \)
- each literal appears at most twice in \( \varphi \).
Boolean Circuits

Semantics

Let $C$ be a Boolean circuit and let $X(C)$ denote the set of variables appearing in the circuit $C$. A truth assignment for $C$ is a function $T : X(C) \rightarrow \{\text{true, false}\}$.

The truth value $T(i)$ for each gate $i$ is defined inductively:

- If $s(i) = \text{true}$, then $T(i) = \text{true}$.
- If $s(i) = \text{false}$, then $T(i) = \text{false}$.
- If $s(i) = \neg x_j \in X(C)$, then $T(i) = T(x_j)$.
- If $s(i) = \neg$, then $T(i) = \text{true}$ if $T(j) = \text{false}$, else $T(i) = \text{false}$ where $(j, i)$ is the unique edge entering $i$.
- If $s(i) = \land$, then $T(i) = \text{true}$ if $T(j) = T(j') = \text{true}$ else $T(i) = \text{false}$ where $(j, i)$ and $(j', i)$ are the two edges entering $i$.
- If $s(i) = \lor$, then $T(i) = \text{true}$ if $T(j) = \text{true}$ or $T(j') = \text{true}$ else $T(i) = \text{false}$ where $(j, i)$ and $(j', i)$ are the two edges entering $i$.
- $T(C) = T(n)$, i.e. the value of the circuit $C$.

Remark. $\{x_1, x_2, \ldots\}$ are variables whose value can be true or false.

CIRCUIT SAT

INSTANCE: Boolean circuit $C$ with variables $X(C)$

QUESTION: Does there exist a truth assignment $T : X(C) \rightarrow \{\text{true, false}\}$ such that $T(C) = \text{true}$?

Theorem

CIRCUIT SAT is NP-complete.

Proof of NP-Membership

Consider the following NP-algorithm:

1. Guess a truth assignment $T : X(C) \rightarrow \{\text{true, false}\}$.
2. Check that $T(C) = \text{true}$ holds.
### Reduction from SAT to 3-SAT

**Proof of NP-Hardness**

We prove the NP-hardness by a reduction from SAT: Let an arbitrary instance of SAT be given by a Boolean formula \( \phi \) over the variables \( X = \{x_1, \ldots, x_k\} \). We construct the following Boolean circuit \( C(\phi) \):

- The variables \( X(C) \) in \( C(\phi) \) are precisely the variables \( X \).
- For every subexpression \( \psi \) of \( \phi \), \( C(\phi) \) contains a gate \( g(\psi) \). The output gate of \( C(\phi) \) is the gate \( g(\phi) \).
- The sort and the incoming arcs of each gate \( g(\psi) \) in \( C(\phi) \) are defined inductively:
  - If \( \psi \) is a variable \( x \), then \( g(\psi) \) is an input gate of sort \( s(g(\psi)) = x \).
  - If \( \psi = \neg \psi' \) then \( s(g(\psi)) = \neg \) with an incoming arc from \( g(\psi') \).
  - If \( \psi = \psi_1 \land \psi_2 \) (resp. \( \psi = \psi_1 \lor \psi_2 \)), then \( s(g(\psi)) = \land \) (resp. \( s(g(\psi)) = \lor \)) with incoming arcs from \( g(\psi_1) \) and \( g(\psi_2) \).

Let an arbitrary instance of CIRCUIT SAT be given by a Boolean circuit \( C \). We construct the following instance \( \phi(C) \) of SAT (\( \phi \) is in CNF with some clauses smaller than 3. The transformation into 3-CNF is obvious):

The formula \( \phi(C) \) uses all variables of \( C \). Moreover, for each gate \( g \) of \( C \), \( \phi(C) \) has a new variable \( g \) and the following clauses.

1. If \( g \) is a variable gate \( x \): \( (g \lor \neg x), (\neg g \lor x) \).
2. If \( g \) is a true (resp. false) gate: \( g \) (resp. \( \neg g \)).
3. If \( g \) is a NOT gate with a predecessor \( h \):
   \[ (\neg g \lor \neg h), (g \lor h) \]
4. If \( g \) is an AND gate with predecessors \( h, h' \):
   \[ (\neg g \lor h), (\neg g \lor h'), (g \lor \neg h \lor \neg h') \]
5. If \( g \) is an OR gate with predecessors \( h, h' \):
   \[ (\neg g \lor h \lor h'), (g \lor \neg h'), (g \lor \neg h) \]
6. If \( g \) is also the output gate: \( g \).

**Motivation**

- We have already seen how an arbitrary propositional formula \( \phi \) can be transformed efficiently into a sat-equivalent formula \( \psi \) in 3-CNF.
- This transformation (first into CNF and then into 3-CNF) is intuitive and clearly works in polynomial time. However, the log-space complexity of this transformation is not immediate.
- We now give an alternative transformation by reducing CIRCUIT SAT to 3-SAT. In total, we thus have:

\[
\text{SAT} \leq \text{L CIRCUIT SAT} \leq \text{L 3-SAT}
\]

**NAESAT**

**Not-all-equal SAT (NAESAT)**

INSTANCE: Boolean formula \( \phi \) in 3-CNF

QUESTION: Does there exist a truth assignment \( T \) appropriate to \( \phi \), such that the 3 literals in each clause do not have the same truth value?

Remark. Clearly \( \text{NAESAT} \subseteq \text{3-SAT} \).

**Theorem**

\( \text{NAESAT} \) is NP-complete.
1-IN-3-SAT

**INSTANCE:** Boolean formula \( \varphi \) in 3-CNF

**QUESTION:** Does there exist a truth assignment \( T \) appropriate to \( \varphi \), such that in each clause, exactly one literal is \textbf{true} in \( T \)?

**MONOTONE 1-IN-3-SAT**

**INSTANCE:** Boolean formula \( \varphi \) in 3-CNF, s.t. the clauses in \( \varphi \) contain only unnegated atoms.

**QUESTION:** Does there exist a truth assignment \( T \) appropriate to \( \varphi \), such that in each clause, exactly one literal is \textbf{true} in \( T \)?

**Theorem**

Both 1-IN-3-SAT and MONOTONE 1-IN-3-SAT are NP-complete.

Proof of NP-Hardness

Recall the Boolean formula \( \varphi(C) \) resulting from the reduction of CIRCUIT SAT to 3-SAT. For all one- and two-literal clauses in the resulting CNF-formula \( \varphi(C) \), we add the same literal \( z \) (possibly twice) to make them 3-literal clauses.

The resulting formula \( \varphi_z(C) \) fulfills the following equivalence:

\[ \varphi_z(C) \in \text{NAESAT} \iff C \in \text{CIRCUIT SAT}. \]

If a truth assignment \( T \) satisfies \( \varphi_z(C) \) in the sense of NAESAT, so does the complementary truth assignment \( T \). Thus, \( z \) is \textbf{false} in either \( T \) or \( T \), which implies that \( \varphi(C) \) is satisfied by either \( T \) or \( T \). Thus \( C \) is satisfiable.

Remarks

- Clearly 1-IN-3-SAT \( \subset \) NAESAT \( \subset \) 3-SAT. The instances of these 3 problems are the same, namely 3-CNF formulae. However, the positive instances of 1-IN-3-SAT are a proper subset of NAESAT, which in turn are a proper subset of the positive instances of 3-SAT.
- Note that the NP-completeness of any of these 3 problems does not immediately imply the NP-completeness of any of the other problems, since it is a priori not clear if further constraining the positive instances makes things easier or harder.
- MONOTONE 1-IN-3-SAT is a special case of 1-IN-3-SAT, i.e., the instances of the former are a proper subset of the latter while the question remains the same. The NP-hardness of the special case immediately implies the NP-hardness of the general case.
Proof of the NP-hardness of **1-IN-3-SAT**

We prove the NP-hardness by a reduction from **4-SAT**:

Let \( \varphi \) be an arbitrary instance of **4-SAT**, i.e., \( \varphi \) is in 4-CNF.

We construct an instance \( \psi \) of **1-IN-3-SAT** as follows:

For every clause \( l_1 \lor l_2 \lor l_3 \) in \( \varphi \), let \( a_1, a_2, a_3, b_1, b_2, c_1, c_2, d \) be 9 fresh propositional variables. Then \( \psi \) contains the following 7 clauses:

\[
\begin{align*}
(1) & \quad l_1 \lor a_1 \lor b_1 \\
(2) & \quad l_2 \lor a_2 \lor b_1 \\
(3) & \quad l_1 \lor a_2 \lor c_1 \\
(4) & \quad l_2 \lor a_3 \lor b_2 \\
(5) & \quad l_3 \lor a_3 \lor c_2 \\
(6) & \quad b_1 \lor b_2 \lor d \\
(7) & \quad b_1 \lor b_2 \lor v \\
\end{align*}
\]

**Idea.** These seven clauses guarantee that in a legal 1-in-3 assignment of \( \psi \), the clause \( l_1 \lor \cdots \lor l_4 \) must be **true**:

By (1) – (3): If \( l_1 \) and \( l_2 \) are **false**, then \( b_1 \) must be **true**.

By (4) – (6): If \( l_3 \) and \( l_4 \) are **false**, then \( b_2 \) must be **true**.

However, by (7), it is not allowed that both \( b_1 \) and \( b_2 \) are **true**.

**HITTING SET**

**INSTANCE:** Set \( T = \{ t_1, \ldots, t_p \} \), family \( (V_i)_{1 \leq i \leq n} \) of subsets of \( T \), i.e., for all \( i \in \{1, \ldots, n\} \), \( V_i \subseteq T \).

**QUESTION:** Does there exist a set \( W \subseteq T \), s.t. \( |W \cap V_i| = 1 \) for all \( i \in \{1, \ldots, n\} \)? (A set \( W \) with this property is called a “hitting set”.)

**Corollary**

**HITTING SET** is NP-complete.

**Proof of the NP-hardness**

By reduction from **MONOTONE 1-IN-3-SAT**: Let an instance of **MONOTONE 1-IN-3-SAT** be given by the 3-CNF formula \( \varphi \), over the variables \( X \). We define the following instance of **HITTING SET**:

\( T = X \). Moreover, suppose that \( \varphi \) contains \( n \) clauses. Then there are \( n \) sets \( (V_i)_{1 \leq i \leq n} \). If the \( i \)-th clause in \( \varphi \) is \( l_1 \lor l_2 \lor l_3 \), then \( V_i = \{ l_1, l_2, l_3 \} \).

**MONOTONE 1-IN-3-SAT**

We show how an arbitrary instance \( \varphi \) of **1-IN-3-SAT** can be transformed into an equivalent instance \( \psi \) of **MONOTONE 1-IN-3-SAT**:

Let \( X = \{ x_1, \ldots, x_n \} \) be the variables in \( \varphi \). Then the variables in \( \psi \) are \( X \cup \{ x'_i \mid 1 \leq i \leq n \} \cup \{ a, b, c \} \). In \( \varphi \), we replace every negative literal of the form \( x_i \) (for some \( i \)) by the unnegated atom \( x'_i \).

Moreover, for every \( i \in \{1, \ldots, n\} \), we add the following 3 clauses:

\[
\begin{align*}
(1) & \quad x_i \lor x'_i \lor a \\
(2) & \quad x_i \lor x'_i \lor b \\
(3) & \quad a \lor b \lor c \\
\end{align*}
\]

**Idea.** These three clauses guarantee that in a legal 1-in-3 assignment of \( \psi \), the variables \( x_i \) and \( x'_i \) have complementary truth values. Hence, \( x'_i \) indeed encodes \( \neg x_i \).

**Some Graph Problems**

We have already proved the NP-completeness of the following graph problems:

- **INDEPENDENT SET**
- **CLIQUE**
- **VERTEX COVER**

We shall now show the following results:

- **3-COLORABILITY** is NP-complete.
- **HAMILTON-PATH** \( \leq_L \) **HAMILTON-CYCLE** \( \leq_L \) **TSP(D)**
**INDEPENDENT SET**

INSTANCE: Undirected graph \( G = (V, E) \) and integer \( K \).

QUESTION: Does there exist an independent set \( I \) of size \( \geq K \)?

i.e., \( I \subseteq V \), s.t. for all \( i, j \in I \) with \( i \neq j \), \([i,j] \notin E \).

**CLIQUE**

INSTANCE: Undirected graph \( G = (V, E) \) and integer \( K \).

QUESTION: Does there exist a clique \( C \) of size \( \geq K \)?

i.e., \( C \subseteq V \), s.t. for all \( i, j \in C \) with \( i \neq j \), \([i,j] \in E \).

**VERTEX COVER**

INSTANCE: Undirected graph \( G = (V, E) \) and integer \( K \).

QUESTION: Does there exist a vertex cover \( N \) of size \( \leq K \)?

i.e., \( N \subseteq V \), s.t. for all \( [i,j] \in E \), either \( i \in N \) or \( j \in N \).

---

**Decision Problems**

**3-COLORABILITY**

INSTANCE: Undirected graph \( G = (V, E) \)

QUESTION: Does \( G \) have a 3-coloring? i.e., an assignment of one of 3 colors to each of the vertices in \( V \) such that any two vertices \( i, j \) connected by an edge \([i,j] \in E \) do not have the same color?

**k-COLORABILITY (for fixed value \( k \))**

INSTANCE: Undirected graph \( G = (V, E) \)

QUESTION: Does \( G \) have a \( k \)-coloring? i.e., an assignment of one of \( k \) colors to each of the vertices in \( V \) such that any two vertices \( i, j \) connected by an edge \([i,j] \in E \) do not have the same color?

---

**NP-Hardness Proof of 3-COLORABILITY**

By reduction from NAESAT: Let an arbitrary instance of NAESAT be given by a Boolean formula \( \varphi = c_1 \land \ldots \land c_m \) in 3-CNF with variables \( x_1, \ldots, x_n \). We construct the following graph \( G(\varphi) \):

Let \( V = \{a\} \cup \{x_i, \neg x_i \mid 1 \leq i \leq n\} \cup \{l_1, l_2, l_3 \mid 1 \leq i \leq m\}, \)

i.e. \( |V| = 1 + 2n + 3m \).

For each variable \( x_i \) in \( \varphi \), we introduce a triangle \([a, x_i, \neg x_i]\), i.e., all these triangles share the node \( a \).

For each clause \( c_i \) in \( \varphi \), we introduce a triangle \([l_1, l_2, l_3]\). Moreover, each of these vertices \( l_j \) is further connected to the node corresponding to this literal, i.e., if the \( j \)-th literal in \( c_i \) is of the form \( x_a \) (resp. \( \neg x_a \)) then we introduce an edge between \( l_j \) and \( x_a \) (resp. \( \neg x_a \))
Example

The 3-CNF formula \( \varphi = (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_3 \lor \neg x_4) \) is reduced to the following graph:

```
\begin{tikzpicture}
  \node (a) at (0,0) [circle, draw] {a};
  \node (x1) at (-2,-1) [circle, draw] {x_1};
  \node (x2) at (-1,-1) [circle, draw] {x_2};
  \node (x3) at (0,-1) [circle, draw] {x_3};
  \node (x4) at (1,-1) [circle, draw] {x_4};
  \node (h1) at (-2,-2) [circle, draw] {h_1};
  \node (h2) at (-1,-2) [circle, draw] {h_2};
  \node (h3) at (0,-2) [circle, draw] {h_3};

  \path
    (a) edge (x1)
    (a) edge (x2)
    (a) edge (x3)
    (a) edge (x4)
    (x1) edge (h1)
    (x1) edge (h2)
    (x1) edge (h3)
    (x2) edge (h1)
    (x2) edge (h2)
    (x2) edge (h3)
    (x3) edge (h1)
    (x3) edge (h2)
    (x3) edge (h3)
    (x4) edge (h1)
    (x4) edge (h2)
    (x4) edge (h3);
\end{tikzpicture}
```

Correctness of the Problem Reduction

Proof (continued)

"\( \Rightarrow \)" Suppose that \( G \) has a 3-coloring with colors \( \{0, 1, 2\} \). W.l.o.g., the node \( a \) has the color 2. This induces a truth assignment \( T \) via the colors of the nodes \( x_i \): if the color is 1, then \( T(x_i) = \text{true} \) else \( T(x_i) = \text{false} \).

We claim that \( T \) is a legal \textit{NAESAT}-assignment. Indeed, if in some clause, all literals had the value \textit{false} (resp. \textit{true}), then we could not use the color 0 (resp. 1) for coloring the triangle \([l_1, l_2, l_3]\), a contradiction.

"\( \Leftarrow \)" Suppose that there exists an \textit{NAESAT}-assignment \( T \) of \( \varphi \).

Then we can extract a 3-coloring for \( G \) from \( T \) as follows:

(i) Node \( a \) is colored with color 2.
(ii) If \( T(x_i) = \text{true} \), then color \( x_i \) with 1 and \( \neg x_i \) with 0 else vice versa.
(iii) From each \([l_1, l_2, l_3]\), color two literals having opposite truth values with 0 (\textit{true}) and 1 (\textit{false}). Color the third with 2.

Example

The 3-CNF formula \( \varphi = (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_3 \lor \neg x_4) \) is reduced to the following graph:

```
\begin{tikzpicture}
  \node (x1) at (0,0) [circle, draw] {x_1};
  \node (x2) at (1,0) [circle, draw] {x_2};
  \node (x3) at (2,0) [circle, draw] {x_3};
  \node (x4) at (3,0) [circle, draw] {x_4};
  \node (l11) at (0,-1) [circle, draw] {l_11};
  \node (l12) at (-1,-1) [circle, draw] {l_12};
  \node (l13) at (-2,-1) [circle, draw] {l_13};
  \node (l21) at (1,-1) [circle, draw] {l_21};
  \node (l22) at (2,-1) [circle, draw] {l_22};
  \node (l23) at (3,-1) [circle, draw] {l_23};

  \path
    (x1) edge (l11)
    (x1) edge (l12)
    (x1) edge (l13)
    (x2) edge (l11)
    (x2) edge (l12)
    (x2) edge (l13)
    (x3) edge (l11)
    (x3) edge (l12)
    (x3) edge (l13)
    (x4) edge (l11)
    (x4) edge (l12)
    (x4) edge (l13);
\end{tikzpicture}
```

Let red = \textit{false} and green = \textit{true}. The above 3-coloring corresponds to \( T(x_1) = T(\neg x_2) = T(\neg x_3) = T(\neg x_4) = \text{true} \).

HAMILTON-PATH

INSTANCE: (directed or undirected) graph \( G = (V, E) \)
QUESTION: Does \( G \) have a Hamilton path?

i.e., a path visiting all vertices of \( G \) exactly once.

HAMILTON-CYCLE

INSTANCE: (directed or undirected) graph \( G = (V, E) \)
QUESTION: Does \( G \) have a Hamilton cycle?

i.e., a cycle visiting all vertices of \( G \) exactly once.

TSP(D)

INSTANCE: \( n \) cities \( 1, \ldots, n \) and a nonnegative integer distance \( d_{ij} \) between any two cities \( i \) and \( j \) (such that \( d_{ij} = d_{ji} \)), and an integer \( B \).

QUESTION: Is there a tour through all cities of length at most \( B \)?

i.e., a permutation \( \pi \) s.t. \( \sum_{i=1}^{n} d_{\pi(i)\pi(i+1)} \leq B \) with \( \pi(n+1) = \pi(1) \).
HAMILTON-PATH, HAMILTON-CYCLE, and TSP(D) are NP-complete.

Proof
We shall show the following chain of reductions:

HAMILTON-PATH $\leq_L$ HAMILTON-CYCLE $\leq_L$ TSP(D)

It suffices to show NP-membership for the hardest problem:
1. Guess a tour $\pi$ through the $n$ cities.
2. Check that $\sum_{i=1}^{n} d_{\pi(i)\pi(i+1)} \leq B$ with $\pi(n+1) = \pi(1)$.

Likewise, it suffices to prove the NP-hardness of the easiest problem. The NP-hardness of HAMILTON-PATH (by a reduction from 3-SAT) is quite involved and is therefore omitted here (see Papadimitriou's book).
Learning Objectives

- The concept of NP-completeness and its characterizations in terms of succinct certificates.
- You should now be familiar with the intuition of NP-completeness (and recognize NP-complete problems).
- Basic techniques to prove problems NP-complete.
- A basic repertoire of NP-complete problems (in particular, versions of SAT and some graph problems) to be used in further NP-completeness proofs.
- Reductions, reductions, reductions, ...