Chapter 34: NP-Completeness

2. Polynomial-time verification

Hamiltonian cycles

(a) is hamiltonian (b) is nonhamiltonian.
A formal language of *hamiltonian-cycle problem*:

\[ \text{HAM-CYCLE} = \{ \langle G \rangle : G \text{ is a hamiltonian graph} \} \]

**Verification algorithms**

**Example:**

- Suppose that a friend tells you that a given graph \( G \) is hamiltonian, and then offers to prove it by giving you the vertices in order along the hamiltonian cycle.
- It is to verify the proof by checking whether it is a permutation of \( V \) and whether each of the consecutive edges along the cycle actually exists in the graph.
A verification algorithm is a two-argument algorithm $A$, where one argument is an ordinary input string $x$ and the other is a binary string $y$ called a certificate, $(A(x, y))$.

An algorithm $A$ verifies a language $L$ if for any string $x \in L$, there is a certificate $y$ that $A$ can use to prove that $x \in L$. Moreover, for any string $x \notin L$, there must be no certificate proving that $x \in L$.

The complexity class NP

- A language $L$ belongs to NP if and only if there exist an algorithm $A$ that verifies $L$ in polynomial time.
- A language $L$ belongs to NP if and only if there exist a two-input polynomial-time algorithm $A$ and constant $c$ such that $^3$
\[ L = \{ x \in \{0, 1\}^* : \text{there exists a certificate } y \text{ with } |y| = O(|x|^c) \text{ such that } A(x, y) = 1 \}. \]

- **HAM-CYCLE \( \in \) NP.**
  
  \( y \) is a list of vertices in the hamiltonian cycle of \( G \) and \( A(\langle G \rangle, y) \) is to check whether consecutive edges along the cycle actually exists in \( G \).

- **If \( L \in P \), then \( L \in NP \). (Thus \( P \subseteq NP \).)**
  
  Example: for the decision problem PATH and an instance \( \langle G, u, v, k \rangle \), \( y \) is a list of vertices in a shortest \( uv \)-path and \( A(\langle G, u, v, k \rangle, y) \) is to sum the edge weights along \( y \) and check whether the summation is at most \( k \).
\[ P = \text{NP}? \text{ (unknown!)} \]

Does \( L \in \text{NP} \) imply \( \overline{L} \in \text{NP} \)?

\text{co-NP} is the set of languages \( L \) such that \( \overline{L} \in \text{NP} \).

So, \( \text{HAM-CYCLE} \in \text{co-NP} \).

\( L \in \text{NP} \) implies \( \overline{L} \in \text{co-NP} \).

\( \text{NP} = \text{co-NP}? \) (unknown!)

\[ P \subseteq \text{NP} \cup \text{co-NP}. \]

\( P = \text{NP} \cup \text{co-NP}? \) (unknown!)
Four possible scenarios:
3. NP-completeness and reducibility

Reducibility

A language $L_1$ is \textit{polynomial-time reducible} to a language $L_2$, written $L_1 \leq_P L_2$, if there exists a polynomial-time computable function $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that for all $x \in \{0, 1\}^*$,

$$x \in L_1 \text{ if and only if } f(x) \in L_2.$$ 

**Lemma 34.3**

If $L_1, L_2 \subseteq \{0, 1\}^*$ are languages such that $L_1 \leq_P L_2$, then $L_2 \in \mathbb{P}$ implies $L_1 \in \mathbb{P}$. 

![Diagram](image)
NP-completeness

► Definition: \( L \) is **NP-hard** if \( L' \leq_P L \) for every \( L \in \text{NP} \).

► Definition: A language \( L \subseteq \{0, 1\}^* \) is **NP-complete** if \( L \in \text{NP} \), and \( L' \leq_P L \) for every \( L' \in \text{NP} \).

► SAT was the first known NP-complete problem, as proved by Stephen Cook in 1971 (see Cook's theorem for the proof).

**Theorem 34.4**

If any NP-complete problem is polynomial-time solvable, then \( \text{P} = \text{NP} \).

Equivalently, if any problem in NP is not polynomial-time solvable, then no NP-complete problem is polynomial-time solvable.
4. NP-completeness proofs

In the celebrated Cook-Levin theorem (independently proved by Leonid Levin), Cook proved that the Boolean satisfiability problem is NP-complete.

In definition, $L$ is **NP-hard** if $L' \leq_P L$ for every $L \in \text{NP}$.

**Lemma 34.8**

If $L$ is a language such that $L' \leq_P L$ for some $L' \in \text{NPC}$, then $L$ is **NP-hard**. Moreover, if $L \in \text{NP}$, then $L \in \text{NPC}$.

In the celebrated **Cook-Levin theorem** (independently proved by **Leonid Levin**), Cook proved that the **Boolean satisfiability problem** is NP-complete.
In 1972, Richard Karp proved that several other problems were also NP-complete (see Karp's 21 NP-complete problems); thus there is a class of NP-complete problems.

Since Cook's original results, thousands of other problems have been shown to be NP-complete by reductions from other problems previously shown to be NP-complete; many of these problems are collected in Garey and Johnson's 1979 book Computers and Intractability: A Guide to the Theory of NP-Completeness.

List of NP-complete problems (Wiki):
5. NP-complete problems on graph theory (from our textbook)
(The following provides the sketch of how to prove that a decision problem is NP-complete. Many details are omitted. Please refer to our textbook for complete proofs.)

**The clique problem**

► **Definition:** a *clique* is a complete subgraph of $G$.

► The *clique problem* is the optimization problem of finding a clique of maximum size in a graph. Its **decision problem**: whether a clique of a given size $k$ exists in the graph.

**Theorem 34.11**

The clique problem is NP-complete.
Proof

• (CLIQUE ∈ NP)
  
  **Certificate:** the set \( V' \subseteq V \) of vertices in the clique for \( G \).
  Checking whether \( V' \) is a clique can be accomplished in polynomial time by checking whether, for each pair \( u, v \in V' \), the edge \((u, v)\) belongs to \( E \).

• (The clique problem is NP-hard by showing 3-CNF-SAT \( \leq_p \) CLIQUE.)

  “An example says thousand words!”

  Example: \( \varphi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \).

  We construct the corresponding graph \( G \) as follows.
A polynomial reduction:
- The input to 3-CNF-SAT can be transformed into input to CLIQUE in polynomial time.
- Suppose $\varphi$ contains $k$ clauses.
  $\varphi$ is satisfied if and only if $G$ has a clique of size $k$. 
The vertex-cover problem

- A vertex cover of an undirected graph $G = (V, E)$ is a subset $V'$ \subseteq V such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$. The size of a vertex cover is the number of vertices in it.
- Example: the graph has a vertex cover \{w, z\} of size 2.

- The vertex-cover problem is to find a vertex cover of minimum size in a given graph. Its decision problem: determine whether a graph has a vertex cover of a given size $k$. 

15
Theorem 34.12
The vertex-cover problem is NP-complete.

Proof

◆ (VERTEX-COVER ∈ NP)
Suppose we are given a graph $G = (V, E)$ and an integer $k$.
Certificate: the vertex cover $V' \subseteq V$ itself.
The verification algorithm affirms that $|V'| = k$, and then it checks, for each edge $(u, v) \in E$, that $u \in V'$ or $v \in V'$.
This verification can be performed in polynomial time.

◆ (The vertex-cover problem is NP-hard by showing that CLIQUE $\leq_P$ VERTEX-COVER)
A polynomial reduction:

- The input \( \langle G, k \rangle \) to CLIQUE can be transformed into the input \( \langle \overline{G}, |V| - k \rangle \) to VERTEX-COVER in polynomial time, where \( \overline{G} \) is the complement of \( G \).
- \( G \) has a vertex cover of size \( k \) if and only if \( \overline{G} \) has a clique of size \(|V| - k|.

**The hamiltonian-cycle problem**

**Theorem 34.13**
The hamiltonian cycle problem is NP-complete.
Proof

- **(HAM-CYCLE ∈ NP)**
  Given a graph $G = (V, E)$, our certificate is the sequence of $|V|$ vertices that makes up the hamiltonian cycle. The verification algorithm checks that this sequence contains each vertex in $V$ exactly once and that there is an edge between each pair of consecutive vertices and between the first and last vertices. This verification can be performed in polynomial time.

- **(VERTEX-COVER ≤p HAM-CYCLE, which shows that HAM-CYCLE is NP-hard.)**
  Given an undirected graph $G = (V, E)$ and an integer $k$, we construct an undirected graph $G' = (V', E')$ that has a
hamiltonian cycle if and only if $G$ has a vertex cover of size $k$.

$G'$ is constructed according to steps 1-3.

Step 1:

An edge $(u, v)$ of graph $G$ corresponds to widget $W_{uv}$ in the graph $G'$ created in the reduction. (a) The widget, with individual vertices labeled.
(b)-(d) The shaded paths are the only possible ones through the widget that include all vertices, assuming that the only connections from the widget to the remainder of $G'$ are through vertices $[u, v, 1]$, $[u, v, 6]$, $[v, u, 1]$, and $[v, u, 6]$. 
Neighbors of $w$ are ordered as $xyz$.
Neighbors of $x$ are ordered as $wy$.
Neighbors of $y$ are ordered as $xw$. 
Step 2:

The only other vertices in $V'$ other than those of widgets are selector vertices $s_1, s_2, \ldots, s_k$.

Step 3:

In addition to the edges in widgets, there are two other types of edges in $E'$.

**First**, for each vertex $u \in V$, we arbitrarily order the vertices adjacent to each vertex $u \in V$ as $u^{(1)}, u^{(2)}, \ldots, u^{(\text{degree}(u))}$ and then add to $E'$ the edges

$$\{(u, u^{(i)}, 6), [u, u^{(i+1)}, 1] : 1 \leq i \leq \text{degree}(u) - 1\}.$$
\textbf{Second}, we include the edges
\begin{align*}
\{(s_j, [u, u^{(1)}], 1]) : u \in V \text{ and } 1 \leq j \leq k\} \cup \\
\{(s_j, [u, u^{(\text{degree}(u))}, 6]) : u \in V \text{ and } 1 \leq j \leq k\}.
\end{align*}

- The transformation from graph $G$ to $G'$ is a reduction:
  - The input $\langle G, k \rangle$ to VERTEX-COVER can be transformed into the input $\langle G' \rangle$ to HAM-CYCLE in polynomial time.
  - $G$ as a vertex cover of size $k$ if and only if $G'$ has a hamiltonian cycle.

(1) Suppose that $G = (V, E)$ has a vertex cover $V^* = \{u_1, u_2, \ldots, u_k\}$. A hamiltonian cycle in $G$ is as
\[ s_1 - [u_1, u_1^{(1)}, 1] \quad s_2 - [u_2, u_2^{(1)}, 1] \quad \cdots \quad s_k - [u_k, u_k^{(1)}, 1] \]

(2) Suppose that \( G = (V, E') \) has a hamiltonian cycle \( C \subseteq E' \). We claim that \( V^* = \{ u \in V : (s_j, [u, u^{(1)}, 1]) \in C \text{ for some } 1 \leq j \leq k \} \) is a vertex cover.
Key observation:

- partition $C$ into maximal paths, called “cover path” as
  
  $$s_i [u, u^{(1)}, 1] \ldots s_j,$$

  without passing through any other selector vertex.

- Each cover path must start at some $s_i$, take the edge $(s_i, [u, u^{(1)}, 1])$ for some vertex $u \in V$, pass through all the widgets corresponding to edges in $E$ incident on $u$, and then end at some selector vertex $s_j$.

- Each vertex in each widget is visited by some cover path, we see that each edge in $E$ is covered by some vertex in $V^*$.
The traveling-salesman problem

- The traveling-salesman problem: given a complete graph where each edge has an integer cost, the salesman wishes to find a hamiltonian cycle of minimum cost.
- The following example has a minimum-cost tour \( \langle u, w, v, x, u \rangle \), with cost 7.
Theorem 34.14
The traveling-salesman problem is NP-complete.

Proof

- **(TSP ∈ NP)**
  Given an instance $\langle G, k \rangle$ of the problem, we use as a certificate the sequence of $n$ vertices in the tour. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge costs, and checks whether the sum is at most $k$. This process can certainly be done in polynomial time.

- **(TSP is NP-hard since HAM-CYCLE $\leq_p$ TSP).**
  - Let $G = (V, E)$ be an instance of HAM-CYCLE. We
construct an instance of TSP as follows. We form the complete graph $G' = (V, E')$, where $E' = \{(i, j) : i, j \in V \text{ and } i \neq j\}$, and we define the cost function $c$ by

$$c(i, j) = \begin{cases} 
0 & \text{if } (i, j) \in E, \\
1 & \text{if } (i, j) \notin E.
\end{cases}$$

The instance of TSP is then $(G', c, 0)$, which is easily formed in polynomial time.

$\blacklozenge$ $G$ has a hamiltonian cycle if and only if graph $G'$ has a tour of cost at most 0.