Computational Complexity 4: NP-Completeness of the Clique Decision Problem

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
Recall:

The **NP-complete class** — denoted as **NPC** — is the set of decision problems $\pi$ such that

- $\pi$ is in NP;
- if $\pi$ can be *solved* in polynomial time, then every problem in NP can be *solved* in polynomial time.

Once the first NPC problem has been found, it is much easier to prove the second.

**Theorem:** Let $\pi^*$ be a decision problem in NPC. If $\pi^*$ can be reduced to another decision problem $\pi$ in polynomial time, then $\pi$ must be NP-hard.
Hence, if $\pi$ is also in NP, $\pi$ is NP-complete.
Recall

**The Clique Decision Problem:** Let $G = (V, E)$ be an undirected graph. Given an integer $k$, decide whether we can find a set $S$ of at least $k$ vertices in $V$ that are mutually connected (i.e., there is an edge between any two vertices in $S$). Those $k$ vertices and the edges among them form a $k$-clique.

**Example:** Consider

The answer is “yes” for $k \leq 3$, but “no” for $k \geq 4$. 
Recall:

**3-SAT**

**Variable**: a boolean unknown $x$ that can be assigned 0 or 1.

**Literal**: a variable $x$ or its negation $\bar{x}$.

**Clause**: the OR of up to 3 literals.

**Formula**: the AND of clauses

**The 3-SAT problem**: Is there an assignment to the variables under which the formula evaluates to 1? Such an assignment is called a **truth assignment**.

**Example**:

$$(x_1 \lor x_2 \lor x_3) \land (x_2 \lor x_3 \lor x_4) \land (\bar{x}_1 \lor \bar{x}_4)$$

The answer is “yes”. A certificate: $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$.

$$(x_1) \land (\bar{x}_1 \lor x_2) \land (\bar{x}_2)$$

The answer is “no”.
Recall

**Theorem:** 3-SAT is NP-complete.

Next we will prove the NP-completeness of the clique decision problem with a reduction **from** the 3-SAT problem. Specifically, we will prove:

**Theorem:** If we have an algorithm $A$ solving the clique decision problem in polynomial time, we can solve the 3-SAT problem using $A$ in polynomial time.

The next few slides serve as a proof of the theorem.
Given an input to 3-SAT — namely a formula $F$ with $k$ clauses — we will construct a graph $G(V, E)$ such that $F$ has a truth assignment if and only if $G$ has a $k$-clique.

We construct $G(V, E)$ as follows:

- For each clause, create a vertex in $V$ for every literal in the clause.
- For each pair of distinct vertices $u, v \in V$, create an edge $\{u, v\}$ in $E$ if
  - The literals corresponding to $u, v$ are not in the same clause.
  - The literals corresponding to $u, v$ are not negations of each other.
Example 1

Consider formula \( F = (x_1 \lor x_2 \lor x_3) \land (x_2 \lor x_3 \lor x_4) \land (\bar{x}_1 \lor \bar{x}_4) \)

**First step:** create vertices

\[
\begin{array}{ccc}
x_1 & x_2 & \bar{x}_3 \\
\circ & \circ & \circ \\
x_2 & \circ & \circ & \circ & \circ \\
x_3 & \circ & \circ & \circ & \circ \\
x_4 & \circ & \circ & \circ & \circ \\
\end{array}
\]

**Second step:** create edges

\[
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
\circ & \circ & \circ \\
x_2 & \circ & \circ & \circ & \circ & \circ & \circ \\
x_3 & \circ & \circ & \circ & \circ & \circ & \circ \\
x_4 & \circ & \circ & \circ & \circ & \circ & \circ \\
\end{array}
\]
Example 2

Consider formula \( F = (x_1) \land (\overline{x_1} \lor x_2) \land (\overline{x_2}) \)

**First step:** create vertices

\[ x_1 \]
\[ \overline{x_1} \]
\[ x_2 \]
\[ \overline{x_2} \]

**Second step:** create edges

\[ x_1 \rightarrow \overline{x_1} \]
\[ x_1 \rightarrow x_2 \]
\[ x_1 \rightarrow \overline{x_2} \]
\[ \overline{x_1} \rightarrow x_2 \]
\[ \overline{x_1} \rightarrow \overline{x_2} \]
\[ x_2 \rightarrow \overline{x_2} \]
Claim 1: If $F$ has a truth assignment, then $G$ has a $k$-clique.

Proof: Every clause has a literal set to 1 in the truth assignment. Pick such a literal from every clause. Clearly, no two literals can be negations of each other (because $x$ and $\bar{x}$ cannot both be 1).

Let $v_i$ be the vertex in $G$ corresponding to that literal in the $i$-th clause ($1 \leq i \leq k$). The claims follows from the fact that there is an edge between any two $v_i, v_j$ for $1 \leq i < j \leq k$. 

☐
Claim 2: If \( G \) has a \( k \)-clique, \( F \) has a truth assignment.

Proof: Let \( v_1, v_2, ..., v_k \) be the vertices of the \( k \)-clique in \( G \). Their corresponding literals must come from different clauses (because no edge exists between the vertices of two literals from the same clause). Furthermore, the literals corresponding to \( v_1, v_2, ..., v_k \) cannot be negations of each other (because no edge exists between the vertices of two literals that are negations of each other). We can therefore construct a truth assignment by setting those \( k \) literals to 1.
The construction of $G$ clearly can be done in polynomial time. We can therefore apply the algorithm $\mathcal{A}$ to determine whether $G$ has a $k$-clique, and thereby, decide whether a truth assignment exists for $F$.

This shows that the clique decision problem is NP-hard.

Combining this with the obvious fact that the problem is in NP, we conclude that the problem is NP-complete.