**NP-Completeness**

An abstract problem is a binary relation on a set of instances and a set of solutions

<table>
<thead>
<tr>
<th>Problem</th>
<th>Instance</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SORTING</strong></td>
<td>Array of keys</td>
<td>( \pi ), permutation of indices that gives sorted order</td>
</tr>
<tr>
<td><strong>SHORTEST-PATH</strong></td>
<td>A graph ( G=(V,E) ) and 2 vertices</td>
<td>The length of the shortest path between the 2 vertices</td>
</tr>
<tr>
<td><strong>CLIQUE</strong></td>
<td>A graph ( G=(V,E) )</td>
<td>The size of the largest clique in ( G )</td>
</tr>
<tr>
<td><strong>SATISFIABILITY</strong></td>
<td>A boolean formula ( \varphi() )</td>
<td>A truth assignment for ( \varphi() ) which satisfies it (makes it evaluate to True) or nil</td>
</tr>
</tbody>
</table>

**NP-Completeness (cont)**

An algorithm for an abstract problem is a mapping which is a subset of the problem (relation)

- **SORTING**
  \[ A_{\text{sorting}} : \{\text{Array of keys}\} \rightarrow \{\pi, \text{permutation of indices}\} \]

An abstract decision problem is an abstract problem whose solution set is \{Yes, No\} (or \{1, 0\})

A decision problem is specified by giving the set of instances and a Yes/No question.

**NP-Completeness (cont)**

An abstract problem’s algorithm can be used to solve the related decision problem directly.

- **CLIQUE**

\[ G=(V,E), k \text{Algorithm for CLIQUE decision problem} \]

\[ G=(V,E), k \text{Algorithm for CLIQUE optimization problem} \]

Yes/No
NP-Completeness (cont)

A decision problem’s algorithm can be used to solve the related abstract optimization problem using a search method on the solution space.

\[\begin{align*}
\text{min} & \leftarrow 1 \\
\text{max} & \leftarrow |V| + 1 \\
\text{while} \ (\text{min} < \text{max} - 1) \\
& \quad \text{mid} \leftarrow (\text{max} - \text{min}) / 2 \\
& \quad \text{If} \ (G \text{ has a clique of size} \geq \text{mid}) \\
& \quad \quad \text{then} \ \text{min} \leftarrow \text{mid} \\
& \quad \quad \text{else} \ \text{max} \leftarrow \text{mid} \\
\text{endwhile} \\
\text{return} \ (\text{min})
\end{align*}\]

NP-Completeness (cont)

Encodings

Complexity of a problem can vary depending on the encoding since complexity is measured as a function of the input size.

Input encoding can be artificially padded or contain the solution to make the complexity appear smaller.

A concrete problem has as its instance set the set of binary strings.

An algorithm solves a concrete problem in \(O(T(n))\) time if it always produces the solution within time \(O(T(n))\) where \(n\) is the length of the binary input string.

NP-Completeness (cont)

Encodings

A concrete problem is polynomially solvable if there is an algorithm which solves it in time \(O(n^k)\) for some constant \(k\).

\[\mathbb{P} = \{\text{polynomially solvable concrete decision problems}\}\]

\[e(\text{SORTING}), e(\text{SHORTEST-PATH}) \in \mathbb{P}\]

\[e(\text{CLIQUE}), e(\text{SATISFIABILITY}) \in \mathbb{P}\]

Can different encodings of an abstract decision problem affect membership in \(\mathbb{P}\)?

NP-Completeness (cont)

An encoding of an abstract decision problem.

\(e : I \rightarrow \{0,1\}^+\) an encoding of an abstract decision problem.

Two encodings \(e_1\) and \(e_2\) of an abstract decision problem are polynomially related if there exist functions \(f_{12}\) and \(f_{21} : \{0,1\}^+ \rightarrow \{0,1\}^+\) such that

\[\forall \text{instance } i, \ f_{12}(e_1(i)) = e_2(i) \text{ and } f_{21}(e_2(i)) = e_1(i)\]

and \(f_{12}\) and \(f_{21}\) are polynomial time computable.

Prevents padding of input, unary encodings and inclusion of solutions (unless solution can be found in polynomial time.)
NP-Completeness (cont)

**Lemma** If $Q$ is an abstract decision problem with two encodings $e_1$ and $e_2$ that are polynomially related, then

$$e_1(Q) \in P \iff e_2(Q) \in P$$

**Proof:**

If $e_1(Q) \in P$, then $e_2(Q) \in P$ for all $Q$ such that $L_1 \leq_p L_2$ and $e_1$ is polynomial time reducible to $e_2$.

$e_1(Q) \in P \Rightarrow e_2(Q) \in P$

For Algorithm $L_1$:

1. $L_1$ is polynomial time reducible to $L_2$
2. Construct $f : \{0,1\}^* \rightarrow \{0,1\}^*$ mapping instances of $L_1$ to $L_2$
3. Show that $x \in L_1 \iff f(x) \in L_2$

In terms of abstract decision problems, this means an instance of $Q_1$ can be converted to an instance of $Q_2$ in polynomial time so that the answer (Yes or No) is the same for both instances.
**NP-Completeness (cont)**

**3SAT (or 3CNF-SAT)**

*Instance:* A boolean formula $\phi$ in conjunctive normal form in which each clause has exactly 3 distinct literals.

*Question:* Can this formula $\phi$ be satisfied? (Is there an assignment of values to its variables that makes it evaluate to 1?)

Example of construction:

$$
\phi = (x_1 \lor x_2 \lor \overline{x}_3)(\overline{x}_1 \lor x_1 \lor x_3)(x_2 \lor \overline{x}_2 \lor x_4)
$$

Let $\phi_1$ be satisfiable

$$
\phi_2 = (x_1 \lor x_2 \lor x_3)(\overline{x}_1 \lor x_1 \lor x_3)(\overline{x}_2 \lor x_2 \lor \overline{x}_3)
$$

$\phi_2$ is not satisfiable

---

**3SAT $\leq_p$ CLIQUE**

Step 1: Construct $f : \{\phi \mid 3\text{SAT formula} \} \rightarrow \{(G,k)\}$

Given $\phi = C_1 C_2 \cdots C_m$ where $C_r = (l_1^r \lor l_2^r \lor l_3^r)$

Let $G = (V,E)$ where $V = \bigcup_{r=1}^{m} \{v_1^r, v_2^r, v_3^r\}$

$$
E = \{(v_i^r, v_j^s) \mid r \neq s \text{ and } l_i^r \neq \overline{l_j^s}\}
$$

Let $k = m$

---

**3SAT $\leq_p$ CLIQUE**

Step 2: Show that $x \in L_{3\text{SAT}} \iff f(x) \in L_{\text{CLIQUE}}$

$\phi$ is satisfiable $\iff f(\phi) = (G,k)$ where $G$ has a clique of size $\geq k$

**Proof:** Suppose $\phi$ is satisfiable.

Then $\exists \overline{x}$ such that $\phi(\overline{x}) = 1$.

Under $\overline{x}$ each clause of $\phi$ has at least one literal whose value is 1.

From each clause $C_r = (l_1^r \lor l_2^r \lor l_3^r)$ of $\phi$ pick a literal $l_i^r$ whose value is 1 under $\overline{x}$.
### NP-Completeness (cont)

**3SAT \( \leq_p \text{ CLIQUE} \)  \hspace{1cm} \text{Proof : (continued)}**

Let \( V' = \{ v_1', v_2', \ldots, v_m' \} \) be the \( m \) vertices corresponding to the \( m \) literals \( l_i' \), just picked.

No pair of vertices of \( V' \) correspond to literals \( x_i \) and \( \overline{x}_j \) since all of the literals corresponding to \( V' \) are 1 under \( \overline{x} \).

Each pair of vertices of \( V' \) share an edge since they belong to different clauses and are not the negation of each other.

\( V' \) is a clique of size \( m \).

\( f(\phi) = (G, k) \) where \( G \) has a clique of size \( \geq k = m \)

---

### NP-Completeness (cont)

**3SAT \( \leq_p \text{ CLIQUE} \)  \hspace{1cm} \text{Proof : (continued)}**

Now suppose \( f(\phi) = (G, k) \) and \( G \) has a clique of size \( \geq k = m \)

Let \( V' \) be a clique of size \( \geq m \) in \( G \).

Since vertices from the same clause do not share edges \( V' \) must have exactly \( m \) vertices one from each clause.

The \( m \) literals corresponding to \( V' \) do not contain both the negated and unnegated form of a variable since their corresponding vertices in \( V' \) would not share an edge.

---

### NP-Completeness (cont)

**3SAT \( \leq_p \text{ CLIQUE} \)  \hspace{1cm} \text{Proof : (continued)}**

Construct a truth assignment \( \overline{x} \) for \( \phi \) by setting variable \( x_i = 1 \) if \( x_i \) appears unnegated in the literals corresponding to \( V' \) and \( x_i = 0 \) otherwise.

By construction all literals corresponding to \( V' \) are 1 under the truth assignment \( \overline{x} \).

Since each clause of \( \phi \) contains a literal corresponding to a vertex in \( V' \), \( \phi(\overline{x}) = 1 \).

\( \phi \) is satisfiable. \hspace{1cm} \text{QED}

---

### NP-Completeness (cont)

**3SAT \( \leq_p \text{ CLIQUE} \)  \hspace{1cm} \text{Step 3: Show that} f is computable in polynomial time.**

If \( \phi \) has \( m \) clauses and \( n \) variables, then \( f(\phi) = (G, m) \) where \( G \) has \( 3m \) vertices and at most \( 9m(m-1)/2 \) edges.

Whether two vertices of \( G \) share an edge or not can be determined by inspection of \( \phi \).

\( f \) is computable in polynomial time.

**3SAT \( \leq_p \text{ CLIQUE} \) has been shown.**