NP Completeness

02 - Additional Problems

February 5, 2003

1 Previous Lecture

1. Efficient algorithms: polynomial running time.
2. P - decision problems that can be solved in polynomial time.
3. \( NP \) - decision problems that can be verified in polynomial time.
4. Question: Is \( P = NP \)?
5. \( NP \)-Hard - problems that if they can be solved in polynomial time, then \( P = NP \).
6. Cook’s Theorem: Circuit Satisfiability is \( NP \)-Hard and thus \( NP \)-Complete.
7. \( CSAT = \) Circuit Satisfiability.
8. \( SAT = \) Formula satisfiability (which is in \( NP \)).
9. By reduction if exists polytime algorithm for \( SAT \) then \( CSAT \) is polytime. Thus, \( SAT \) is \( NP \)-Complete.
10. \( 3SAT = \) Formula satisfiability where the formula is restricted to be a 3CNF: 
    \[ (a \lor b \lor c) \land (\overline{b} \lor c \lor d) \]
11. By reduction - if we can solve \( 3SAT \) in polytime, then \( CSAT \) can be solved in polytime.

2 Max-Clique

**Problem:** MAXCLIQUE

*Instance:* A graph \( G \)
*Question:* What is the largest number of nodes in \( G \) forming a complete subgraph?
Q: Describe an algorithm for solving MaxClique?

- Enumerate all subsets $S \subseteq V(G)$
  - Check if $S$ is a clique in $G$
- Return the largest $S$ s.t. $G_S$ is a clique.

Running time: $O(2^n n^2)$

**Remark 2.1** When solving a problem, always try first to find a simple solution - try optimizing it later.

We will prove that MaxClique is NP-Hard.

Q: Why NP-Complete/Hard problems take exponential time?

A: Intuitively, we have to try all possibilities.

**How to prove that a problem X is NP-Hard?**

1. Chose a known NP-Complete problem: A.
2. Show how to solve any instance of A in polynomial time, assuming that you are given a polynomial time algorithm to solve X.

**Theorem 2.2** MaxClique is NP-Hard.

**Proof**: We show a reduction from 3SAT. [formula in 3SAT looks like: $(a \vee b \vee c) \wedge (b \vee c \vee d) \wedge (a \vee c \vee d) \wedge (a \vee b \vee d).$]

Let $F$ be the given 3SAT formula defined over $n$ variables with $m$ clauses.

We build a graph:

1. Every literal in the formula is a vertex.
   We label a vertex with the literal it corresponds to.
2. Every clause correspond to the three such vertices.
3. We connect two vertices in the graph, if they are:
   (a) In different clauses,
(b) **and** are NOT a negation of each other.

Let $G$ denote the resulting graph. See Figure 1 for an example.

We claim, that $F$ is satisfiable iff there exists a clique of size $m$ in $G$.

⇒ Let $x_1, \ldots, x_n$ be the variables in $F$. Let $v_1, \ldots, v_n$ be the satisfying assignment

For every clause $C$ in $F$, there must be at least one literal that evaluates to TRUE. Pick a "TRUE" vertex from each clause. Let $W$ be the resulting set of vertices. Clearly, $W$ form a clique in $G$. The set $W$ is of size $m$.

⇐ Let $U$ be the set of $m$ vertices which form a clique in $G$.

What if the largest clique in $G$ is of size $m - 1$?

Then, the original formula $F$ is not satisfiable!

1. $x_i \leftarrow true$ if there is a vertex in $U$ labeled with $x_i$.

2. $x_i \leftarrow false$ if there is a vertex in $U$ labeled with $\overline{x_i}$.

This is a valid assignment (why?). This is a satisfying assignment, as there is at least one vertex of $U$ in each clause, and as such, there is a literal evaluating to TRUE in each clause. Namely, $F$ evaluates to TRUE.

Thus, given a polytime algorithm for MaxClique, we can solve 3SAT in polytime. Thus, MaxClique in $NP$-Hard.

Observations:

1. Life sucks, and then you die.

2. MaxClique is an optimization problem, however we can restate it:

**Problem:** CLIQUE

| Instance: A graph $G$, integer $k$ |
| Question: Is there a clique in $G$ of size $k$? |

**Theorem 2.3** CLIQUE is $NP$-Complete.
Proof: It is \( NP \)-Hard, by the previous reduction. Thus, we only need to show that it is in \( NP \). Easy:

Given a graph \( G \) having \( n \) vertices, a parameter \( k \), and a set \( W \) of \( k \) vertices, verifying that every pair of vertices in \( W \) form an edge in \( G \), takes \( O(u + k^2) \), where \( u \) is the size of the input (i.e., number of edges + number of vertices).

Thus, \textsc{Clique} is \( NP \)-Complete.

Synonym to \textsc{Clique}?

coterie - a close circle of friends who share a common interest or background; clique.

3 \textbf{IndependentSet}

\textbf{Problem:} \textsc{IndependentSet}

\textbf{Instance:} A graph \( G \), integer \( k \)

\textbf{Question:} Is there an independent set in \( G \) of size \( k \)?

\textbf{Theorem 3.1} \textsc{IndependentSet} is \( NP \)-Complete.

\textit{Proof:} We do a reduction from \textsc{Clique}. Given \( G \) and \( k \), compute the complement graph \( \overline{G} \) where we connected two vertices \( u, v \) in \( \overline{G} \) iff they are independent in \( G \). Clearly, a clique in \( G \) corresponds to an independent set in \( \overline{G} \). Thus, \textsc{IndependentSet} is \( NP \)-hard, and since it is in \( NP \), it is \( NPC \).
4 Vertex Cover

Definition 4.1 For a graph $G$, a set of vertices $S \subseteq V(G)$ is a Vertex Cover if it touches every edge of $G$.

**Problem:** \textsc{VertexCover}

| \textit{Instance:} A graph $G$, integer $k$ |
| \textit{Question:} Is there a vertex cover in $G$ of size $k$? |

Observation 4.2 $S$ is a vertex cover in $G$ iff $V \setminus S$ is an independent set in $G$.

Theorem 4.3 \textsc{VertexCover} is \textsc{NP}-Complete.

*Proof:* \textsc{VertexCover} is \textsc{NP}. Reduction from Independent Set. Given a graph $G$ and parameter $k$, we ask whether the graph $G$ has a \textsc{VertexCover} of size $n - k$.

5 Graph Coloring

Definition 5.1 A coloring, is a mapping $C : V(G) \rightarrow \{1, 2, \ldots, c\}$ such that every edge the colors assigned to its endpoints are different.

Coloring is extremely useful for:

1. Resource allocation - used in compilers
2. Scheduling.

**Problem:** \textsc{3Colorable}

| \textit{Instance:} A graph $G$. |
| \textit{Question:} Is there a coloring of $G$ using three colors? |

Theorem 5.2 \textsc{3Colorable} is \textsc{NP}-Complete.

*Proof:* \textsc{3Colorable} is clearly in \textsc{NP}.

The reduction is from 3SAT. Let $F$ be the given 3SAT formula. We are going to transform $F$ into a graph using gadgets.

Color generating gadget

We have three special vertices: X, F, T.

Variable Gadget

We have three special vertices: X, F, T.
Note that X here is the SAME vertex as the X vertex in the above drawing.

**Clause Gadget**

\[
\begin{align*}
& a \\
& b \\
& \overline{\tau} \\
& \overline{\tau} \\
& \vdash
\end{align*}
\]

\[a \lor b \lor \overline{c}\]

For example, the formula:

\[
(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d}).
\]

Generates the graph:

![Diagram of a graph with labeled vertices and edges representing a clause gadget for SAT problem.]