CS154, Lecture 16: More NP-Complete Problems; PCPs
Is 3SAT solvable in $O(n)$ time on a multitape TM?

Are there logic circuits of size $6n$ for 3SAT?

If yes, then not only is $P=NP$, but there would be a “dream machine” that could crank out short proofs of theorems, quickly optimize all aspects of life...

recognizing quality work is all you need to produce
There are thousands of NP-complete problems

Your favorite topic certainly has an NP-complete problem somewhere in it

Even the other sciences are not safe: biology, chemistry, physics have NP-complete problems too!
Given a favorite problem $\Pi \in \text{NP}$, how can we prove it is NP-hard?

Generic Recipe:
1. Take a problem $\Sigma$ that you know to be NP-hard (3-SAT)
2. Prove that $\Sigma \leq_p \Pi$

Then for all $A \in \text{NP}$, $A \leq_p \Sigma$ and $\Sigma \leq_p \Pi$
We conclude that $A \leq_p \Pi$, and $\Pi$ is NP-hard
$\Pi$ is NP-Complete
The Clique Problem

Given a graph $G$ and positive $k$, does $G$ contain a complete subgraph on $k$ nodes?

CLIQUE = \{(G,k) \mid G\text{ is an undirected graph with a } k\text{-clique}\}

Theorem (Karp): CLIQUE is NP-complete
Proof Idea: $3\text{SAT} \leq_p \text{CLIQUE}$

Transform a 3-cnf formula $\phi$ into $(G,k)$ such that

$\phi \in 3\text{SAT} \iff (G,k) \in \text{CLIQUE}$

Want transformation that can be done in time that is polynomial in the length of $\phi$

How can we encode a logic problem as a graph problem?
We transform a 3-cnf formula $\phi$ into $(G,k)$ such that

$$\phi \in 3\text{SAT} \iff (G,k) \in \text{CLIQUE}$$

Let $C_1, C_2, ..., C_m$ be clauses of $\phi$. Assign $k := m$. Make a graph $G$ with $m$ groups of 3 nodes each.

Group $i$ corresponds to clause $C_i$ of $\phi$. Each node in group $i$ is labeled with a literal of $C_i$.

Put edges between all pairs of nodes in different groups, except pairs of nodes with labels $x_i$ and $\neg x_i$.

Put no edges between nodes in the same group.

When done putting in all the edges, erase the labels.
\[(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)\]

\[|V| = 9\]

\[k = 3\]
\((x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land (x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)\)
Claim: \( \phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE} \)

Claim: If \( \phi \in 3\text{SAT} \) then \((G,m) \in \text{CLIQUE})
Proof: Given a SAT assignment \( A \) of \( \phi \), for every clause \( C \) there is at least one literal in \( C \) that’s set true by \( A \)
For each clause \( C \), let \( v_C \) be a vertex from group \( C \) whose label is a literal that is set true by \( A \)

Claim: \( S = \{v_C : C \in \phi\} \) is an \( m \)-clique
Proof: Let \( v_C, v_C' \) be in \( S \). Suppose \((v_C,v_C') \notin E\).
Then \( v_C \) and \( v_C' \) must label inconsistent literals, call them \( x \) and \( \neg x \)
But assignment \( A \) cannot satisfy both \( x \) and \( \neg x \)
Therefore \((v_C,v_C') \in E\), for all \( v_C, v_C' \in S \).
Hence \( S \) is an \( m \)-clique, and \((G,m) \in \text{CLIQUE} \)
Claim: \( \phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE} \)

Claim: If \((G,m) \in \text{CLIQUE}\) then \(\phi \in 3\text{SAT}\)

Proof: Let \(S\) be an \(m\)-clique of \(G\). We construct a satisfying assignment \(A\) of \(\phi\).

Claim: \(S\) contains exactly one node from each group.

Now for each variable \(x\) of \(\phi\), make assignment \(A\):

Assign \(x\) to 1 \(\iff\) There is a vertex \(v \in S\) with label \(x\)

For all \(i = 1,\ldots,m\), at least one vertex from group \(i\) is in \(S\).
Therefore, for all \(i = 1,\ldots,m\). \(A\) satisfies at least one literal in the \(i\)th clause of \(\phi\). Therefore \(A\) is a satisfying assignment to \(\phi\)
Independent Set

**IS:** Given a graph $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| = k$ and no two vertices in $S$ have an edge? 

**IS** = \{ $(G,k)$ | $G$ is an undirected graph with an IS of size $k$ \}

**CLIQUE:** Given $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| = k$ and every pair of vertices in $S$ have an edge?

**CLIQUE \leq_p IS:**
Given $G = (V, E)$, output $G' = (V, E')$ where $E' = \{ (u,v) | (u,v) \notin E \}$.

$$(G, k) \in \text{CLIQUE} \text{ iff } (G', k) \in \text{IS}$$
The Vertex Cover Problem

vertex cover = set of nodes $C$ that cover all edges:
For all edges, at least one endpoint is in $C$
VERTEX-COVER = \{ (G,k) | G is a graph with a vertex cover of size at most k \}

Theorem: VERTEX-COVER is NP-Complete
(1) VERTEX-COVER ∈ NP
(2) IS ≤_p VERTEX-COVER
IS $\leq_p$ VERTEX-COVER

Want to transform a graph $G$ and integer $k$ into $G'$ and $k'$ such that

$$(G,k) \in IS \iff (G',k') \in VERTEX$-COVER$$
IS \leq_p \textsc{VERTEX-COVER}

Claim: For every graph $G = (V,E)$, and subset $S \subseteq V$, $S$ is an independent set if and only if $(V - S)$ is a vertex cover.

Proof: $S$ is an independent set

$\iff (\forall u, v \in V)[(u \in S \text{ and } v \in S) \Rightarrow (u,v) \notin E]$

$\iff (\forall u, v \in V)[(u,v) \in E \Rightarrow (u \notin S \text{ or } v \notin S)]$

$\iff (V - S)$ is a vertex cover

Therefore $(G,k) \in \text{IS} \iff (G,|V| - k) \in \text{VERTEX-COVER}$

Our polynomial time reduction: $f(G,k) := (G, |V| - k)$
The Subset Sum Problem

Given: Set $S = \{a_1, \ldots, a_n\}$ of positive integers and a positive integer $t$

Is there an $A \subseteq \{1, \ldots, n\}$ such that $t = \sum_{i \in A} a_i$?

$\text{SUBSET-SUM} = \{(S, t) | \exists A \subseteq S \text{ s.t. } t = \sum_{i \in A} a_i\}$

A simple number-theoretic problem

Theorem: $\text{SUBSET-SUM}$ is NP-complete

Note: There is an $O(n \cdot t)$ time algorithm for Subset Sum. Does this prove $P=NP$?
Want to reduce a graph to a set of numbers

Given \((G, k)\), let \(E = \{e_0, ..., e_{m-1}\}\) and \(V = \{1, ..., n\}\)

Our subset sum instance \((S, t)\) will have \(|S| = n+m\)

“Edge numbers”: For every \(e_j \in E\), put \(b_j = 4^j\) in \(S\)

“Node numbers”: For every \(i \in V\), put \(a_i = 4^m + \sum_{j: i \in e_j} 4^j\) in \(S\)

Set the target number: \(t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)\)
For every $e_j \in E$, put $b_j = 4^j$ in $S$
For every $i \in V$, put $a_i = 4^m + \sum_{j : i \in e_j} 4^j$ in $S$
Set the target number: $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

Claim: If $(G, k) \in VC$ then $(S, t) \in \text{SUBSET-SUM}$
Suppose $C \subseteq V$ is a VC with $k$ vertices.
Let $S' = \{a_i : i \in C\} \cup \{b_j : |e_j \setminus C| = 1\}$
$S' = (\text{node numbers corresponding to nodes in } C) \text{ plus}$
$\quad (\text{edge numbers corresponding to edges covered only once by } C)$

Claim: The sum of all numbers in $S'$ equals $t$

Think of the numbers as being in “base 4”... as vectors with $m+1$ components
For every $e_j \in E$, put $b_j = 4^j$ in $S$

For every $i \in V$, put $a_i = 4^m + \sum_{j : i \in e_j} 4^j$ in $S$

Set the target number: $t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$

Claim: If $(S,t) \in \text{SUBSET-SUM}$ then $(G,k) \in \text{VC}$

Suppose $C \subseteq V$ and $F \subseteq E$ satisfy

$$\sum_{i \in C} a_i + \sum_{e_j \in F} b_j = t = k \cdot 4^m + \sum_{j=0}^{m-1} (2 \cdot 4^j)$$

Claim: $C$ is a vertex cover of size $k$.

Proof: Subtract out the $b_j$ numbers from the above sum. What remains is a sum of the form:

$$\sum_{i \in C} a_i = k \cdot 4^m + \sum_{j=0}^{m-1} (c_j \cdot 4^j)$$

where each $c_j > 0$. But $c_j =$ number of nodes in $C$ covering $e_j$

This implies $C$ is a vertex cover!
Finding Paths - Two Problems

Let $G$ denote a graph, and $s$ and $t$ denote nodes.

**SHORTEST PATH**
$= \{(G, s, t, k) \mid G \text{ has a simple path of length } < k \text{ from } s \text{ to } t \}$

**LONGEST PATH**
$= \{(G, s, t, k) \mid G \text{ has a simple path of length } > k \text{ from } s \text{ to } t \}$

Are either of these in P? Are both of them?
HAMPATH = \{(G,s,t) \mid G \text{ is an directed graph with a Hamiltonian path from } s \text{ to } t\}

Theorem: HAMPATH is NP-Complete

(1) HAMPATH ∈ NP

(2) 3SAT ≤_p HAMPATH

See Sipser for the proof
HAMPATH $\leq_p$ LONGEST-PATH

LONGEST-PATH
$= \{ (G, s, t, \mathbf{k}) \mid G \text{ has a simple path of length } > \mathbf{k} \text{ from } s \text{ to } t \}$

Can reduce HAMPATH to LONGEST-PATH by observing:

$(G, s, t) \in \text{HAMPATH} \iff (G, s, t, |V|) \in \text{LONGEST-PATH}$

Therefore LONGEST-PATH is NP-hard.
Coping with NP-Completeness [Advanced Topics]

There are thousands of NP-complete problems

Many are solved all the time !?!

Average Case vs. Worst Case; Heuristics vs. Algorithms [Beyond Worst-Case Analysis (CS264)]

Special cases/parameters that make a problem easy

Approximation Algorithms
Approximating Vertex Cover

**Vertex Cover** = set of nodes that cover all edges. **Minimization problem**: find the smallest VC

A very simple (greedy) approximation algorithm \( A \): finds a VC that is at most twice as large as the optimal (a 2-approximation).

**Algorithm**: Set \( C=\emptyset \) and while there exist uncovered edge \( e \), add both endpoints of such \( e \) to \( C \)

**Why does it work?**
Max-SAT

Max-SAT = given a cnf formula how many clauses can be satisfied? A maximization problem: satisfy the most clauses

Can always satisfy a constant fraction of all the clauses. Specifically: When all clauses have at least 3 literals, can satisfy at least $7/8$ of all clauses $\geq 7/8$ of clauses of clauses in optimal solution ($\Rightarrow$ a $7/8$-approximation).

Can we approximate MAX-SAT up to any constant $< 1$? Not if P≠NP

For other problems (clique) no constant-approximation is likely
The PCP Theorem

For some constant $\alpha > 0$ and for every language $L \in \text{NP}$, there exists a polynomial-time computable function $f$ that maps every input $x$ into a 3cnf formula $f(x)$ s.t.

- If $x \in L$ then $f(x) \in \text{SAT}$
- If $x \not\in L$ then no assignment satisfies more than $(1 - \alpha)$ fraction of $f(x)$ clauses.

$\Rightarrow$ sufficiently good approximation of MAX-SAT implies P=NP
A rich literature giving exceedingly sophisticated approximation algorithms and exceedingly sophisticated inapproximability results.

Know the best approximation factors for a wide range of problems (especially those where the algorithms are simple).

Inapproximability results via stronger PCPs and via approximation-preserving reductions.

3SAT \leq_p CLIQUE is (very) approximation-preserving; why?
IS \leq_p VERTEX-COVER is completely not; why?
PCPs = Probabilistically Checkable Proofs

Alternative statement of PCP Theorem (informal):

Every statement that has a polynomial-time verifiable proof has such a proof where the verifier only reads $O(1)$ bits of the proof such that

[perfect completeness]: if the statement is correct accept with Probability 1
[soundness]: if the statement is false reject with probability 0.99
PCPs = Probabilistically Checkable Proofs

Alternative statement of PCP Theorem (informal):

Every statement that has a polynomial-time verifiable proof has such a proof where the verifier only reads $O(1)$ bits of the proof such that

[perfect completeness]: if the statement is correct accept with Probability 1
[soundness]: if the statement is false reject with probability 0.99
Interactive Proofs

PCPs add randomness to proofs, what if we also add interaction?

\[ x \in L? \]

Interactive Proofs can be used to prove membership in powerful (PSPACE) languages. For example: \( V \) knows a winning Chess strategy: \( \text{IP}=\text{PSPACE} \)

Zero-Knowledge Proofs – reveal no information apart of \( x \in L \)