Intractability and NP Completeness (II)

Lecture 16
Recall, the Class NP

NP: Decision problems $D$ for which there exist polynomial time algorithm $V$ & $c > 0$

- if $D(x) = 1$ (x is a yes-input of $D$),
  then $\exists \ y, \ |y| < |x|^c, \ V(x,y) = True$
- if $D(x) = 0$ (x is a no-input of $D$),
  then $\forall y, \ V(x,y) = False$

$V$: verification algorithm

$y$: certificate of $x \in D$
NP-Complete Problems:

The “hardest” problems in NP

How would you define NP-Complete?
Polynomial Time Reductions

• Let $A$ and $B$ be decision problems
• $A$ is polynomial time reducible to $B$ if there exists a polynomial time algorithm $R$ such that:
  - $R$ transforms input $x$ to problem $A$ into inputs $R(x)$ to problem $B$
  - $x$ is a yes-input for $A$ if and only if $R(x)$ is a yes-input for $B$

Notation: $A \leq B$
NP-completeness

D is NP-complete:
1. D is in NP
2. for all decision problems A in NP, A is polynomial time reducible to D (D is NP-hard)

Theorem [Cook-Levin]:
Circuit Satisfiability (cSAT) is NP-complete
Proving Cook-Levin Theorem: cSAT is NP-complete (I)

Claim: cSAT is in NP

Proof: certificate = assignment to inputs $x_1 \ldots x_n$ of $C$ that make $C(x_1 \ldots x_n) = 1$
Proving Cook-Levin Theorem

cSAT is NP-complete (II)

Claim: For any decision problem $D \in NP$, $D \leq cSAT$.

Idea: For every $D$. Show how to transform in polynomial time an input for problem $D$ into a circuit $C$ such that $D(x) = 1$ iff $C$ is satisfiable.
D ≤ cSAT ∀ D in NP

- D could be any decision problem in NP. What do we know about it?
- Only that: ∃ polynomial time algorithm V(x,y) such that ∀ yes-input x,
  ∃ short y such that V(x,y)=True

**Reduction Idea:** On input x, the reduction builds a Boolean circuit $C_x$ on variables $y=y_1...y_n$
  s.t. $C_x(y)$ is satisfiable if and only if $V(x,y)=true$

**Challenge:** Build polynomial size $C_x$, starting with polynomial time $V$ and $x$
View computation steps of $V$ (verifying $V(x,y) = \text{Yes}$) as a sequence of $Q(|x|) = \text{poly}(|x|)$ configurations. Each configuration consists of:

- the program for $V$,
- the program counter,
- the input $x$,
- the certificate $y$,
- and working storage.

Computing each configuration from the previous one starting from an initial configuration can be computed by a small combinatorial circuit of size $Q(|x|) \log Q(|x|)$ (essentially implementing a CPU).
Fix all inputs beside $y$

This circuit built of a sequence of sub-circuits taking one configuration to the next till the final output is of $V(x,y)$ is produced is the output of the reduction $C_x$. 0/1 output
Wrapping up Cook’s theorem

Reduction $R$ is essentially a CAD tool: On input $x$ takes $V(x, y)$ and outputs a description of a circuit $C_x(y)$ that computes the same function

- The size of the circuit is a polynomial in $|x|$
- $x$ is a yes-input of $D$ if and only if
  - $\exists y$ that makes $V(x, y) = \text{True}$ iff
  - $\exists y$ that makes $C_x(y) = 1$

QED
Conclusion: all of the above problems are NP-complete
Recall: To Show a Problem is NP Complete

Recipe: To show D is NP-Complete,
(1) show D is in NP
(2) show D is NP-hard: reduce a previously known NP-Complete problem to C
**SAT**

- **Input:** Boolean Formula \( \phi \) in CNF form
- **Question:** Does \( \phi \) have a satisfying assignment

\[
\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)
\]

- Is SAT in NP?
  - Yes, a satisfying assignment is a `verifiable short certificate`

- **K-SAT:** each clause has exactly \( k \) literals
- **2-SAT:** is in polynomial time – show it
3-SAT is NP-complete

To prove NP-completeness:

✓ 3SAT ∈ NP

• Reduce cSAT to 3SAT:
  Show how to transform a circuit C in polynomial time to a 3CNF formula φ
  s.t. C ∈ cSAT if and only if φ ∈ 3SAT
cSAT $\leq$ 3-SAT

- **On input C** with m variables & gates **construct** $\phi$
- **Variables of $\phi$:**
  - $g_1, g_2, ... g_n$ correspond to C’s variables $x_1, ..., x_n$,
  - $g_j$, for $j > n$ corresponds to the output of gate $j$
  - $g_{m+1}$ corresponds to the value of the circuits output gate.
- **Clauses of $\phi$:** for each gate $j > n$, create sub-formulas $\phi_j$ corresponding to the gate type

For $\phi_j = (g_i \vee g_j) \land (\overline{g}_i \vee \overline{g}_j) 

\text{FACT: } \phi_j \text{ is TRUE iff } g_i = \overline{g}_j$
\( \phi_j \) for AND gate \( j \)

- For a gate \( j \) which is an AND gate applied to the output of gates and add the sub-formula

\[
\phi_j = (\bar{g}_j \lor g_i) \land (\bar{g}_j \lor g_e) \land (g_j \lor g_i \lor g_e)
\]

FACT: \( \phi_j \) is true iff \( (g_i \text{AND } g_e) = \text{true} \)

- Similarly for OR gate: do it!
The final formula is of poly(m) size
\[ \phi = (\phi_{n+1} \land \phi_1 \ldots \land \phi_m) \land g_{m+1} \]

\( \phi \) is satisfiable \( \Rightarrow \)
- all \( \phi_i \) is satisfiable \( \Rightarrow \) the output of gate \( i \) is computed properly on its inputs
- \( g_{m+1} \) is true \( \Rightarrow \) circuits output gate is 1

\( \checkmark \) \( C(x_1\ldots x_n) \) is 1 on some \( x_1\ldots x_n \) \( \Rightarrow \)
- For \( i=1\ldots n \), let the formula variables \( g_i = x_i \)
- For \( j>n \), let variable \( g_j \) = the output of gate \( j \) of \( C \) on inputs \( x_1\ldots x_n \)
- Then \( \phi_j \) is true \( \forall j \) & \( g_{m+1} \) is 1 \( \Rightarrow \) \( \phi \) is satisfiable
Clique

Input: An undirected graph $G=(V,E)$ be and $K > 0$

Question: Is there a subset $C$ of $V$, $|C| \geq K$ such that every pair of vertices in $C$ has an edge between them? That is has a clique of size $K$

Example: Clique of size 3 (easy to detect)

Theorem: Clique is NP-Complete
Proof: First, obviously Clique is in NP
3SAT \leq \text{ Clique Reduction}

**Goal:** Given 3CNF $\phi = C_1, \ldots, C_m$ over $x_1, \ldots, x_n$, construct $G=(V,E)$ and $K$ s.t $\phi$ satisfiable iff $G$ has a clique of size $\geq K$.

Notation: a literal $t$ is either $x_i$ or $\neg x_i$

**Reduction:**
- **Vertices:** Add vertex for each literal $t$ occurring in each clause
- **Create an edge** $v_t - v_{t'}$ unless
  - $t$ and $t'$ are in the same clause, or
  - $t$ is the negation of $t'$
- **Set** $K=m$
3SAT ≤ Clique Reduction Example

Say, formula has clauses:
\[ x_1 \lor x_2 \lor x_3, \neg x_2 \lor \neg x_3, \neg x_1 \lor x_2 \]

Graph: G

**Claim:** \( \phi \) satisfiable iff \( G \) has clique of size \( \geq m \)
Proof

“φ satisfiable ⇒” part:
- Take any assignment that satisfies φ.
  E.g., \(x_1 = F, x_3 = F, x_2 = T\)
- Let the set \(C\) contain for each clause, one literal which made the clause true
  \(⇒\) \(C\) is a clique of size \(m\)

- For each literal \(t\) occurring in a clause, create a vertex \(v_t\)
- Create an edge \(v_t - v'_t\) unless
  - \(t\) and \(t'\) are in the same clause, or
  - \(t\) is the negation of \(t'\)
Proof

"⇐ G has clique of size ≥ m" part:
- Take any clique C of size ≥ m (i.e. = m)
- Must contain one literal from each clause
- Literals contained are not the negation of each other
- Set all literals in the clique to evaluate to true in \( \phi \), E.g., \( x_3 = T, x_2 = F, x_1 = F \)
- This is a legal assignment which satisfies \( \phi \)
To summarize

✓ We constructed a reduction that maps:
  – YES-inputs of 3CNF to YES-inputs of Clique
  – NO-inputs of 3CNF to NO-inputs of Clique

✓ The reduction works in polynomial time

• Therefore, established SAT ≤ Clique
  and Clique NP-hard

• Since Clique ∈ NP and Clique is NP-hard ⇒
  Clique is NP-complete
Independent Set in NP-complete

Input: An undirected graph G=(V,E) and integer K>0.
Question: is there a subset S of V of size K which is an independent set (IS) (that is there are no edges between vertices in S)

Claim: Independent Set is NP-complete
Proof: Obviously, it is in NP
Complement of a Graph

- Def: $G'=(V,E')$ is the *complement* of $G=(V,E)$ if $(u,v)$ is in $E'$ iff $(u,v)$ is not in $E$.
Clique ≤ Independent Set

- **Claim**: S is a clique in G iff S is an independent set in the complement of G

- **To reduce** Clique to IS, compute the complement of the graph. The complement has an independent set of size K iff the original graph had a clique of size K.

- So, (G,K) is reduced to (G’,K) where G’ is the complement of G
Clique $\leq$ Independent Set Example

- This is the *dual problem!*
Vertex Cover is NP-complete

- **Input:** An undirected graph \( G = (V,E) \) and an integer \( K > 0 \)
- **Question:** Is there a subset of \( V \) which is a vertex cover of size at most \( K \)
  (that is a subset \( S \) of \( V \) such that every edge in \( E \) has at least one endpoint in \( S \))

Claim: Vertex Cover is NP-complete
Proof: Obviously its in NP
**Independent Set and Vertex Cover**

**Lemma:** S is independent set in graph G iff V-S is a vertex cover in G

⇒ If S in independent set in G, there are no edges between vertices in S, so all edges must have an end point in a vertex outside of S, and V-S must cover all edges

⇐ If S’ is a vertex cover in G, then all edges must have at least one end point in S’, so any pair of vertices out of S’ cannot have an edge between them otherwise it wouldn’t be covered. Thus, V-S’ is independent set
Find a maximum independent set $S$

Show that $V-S$ is a vertex cover
Summary: Showing NP-completeness techniques

- **cSAT**: describe general verification as an instance of cSAT.
- **3SAT and Clique**: introduce gadgets
- **IS, VC**: simple reductions between problems
- Special case to general case: Set Cover (in book)

Many problems that appear different on the surface can be efficiently reduced to each other, revealing a deeper similarity.
The Importance of NP-completeness

• If you recognize your problem is NP-complete, don’t waste trying to solve it in polynomial time
  - Use slow algorithm
  - Settle for an approximation of the solution (this applies to the search and optimization versions of the NP complete decision problem)
  - Change your problem formulation so its in P rather than being NP-complete. Sometime special cases are surprisingly easy.
Remarks on NP-complete problems

- **Special cases** of a problem may be in P even if the general case is not known to be
  - 2-SAT

- **Representation is important:** Sometimes if inputs to a problem (or even just part of the input) is specified not in binary but say in unary, then it becomes much easier
  - Clique when K is in unary. Can solve in polynomial in $n^K$ time.

- **Fixed vs. Growing:** Sometimes a portion of the input is really the same for all inputs (i.e. fixed), and that can make the problem tractable.
  - Integer programming with a fixed number of variables – is in P, using $L^3$
  - Clique with $k=3$ (i.e. deciding if there exists a triangle in the graph)
Not All Intractable Problems in NP are NP-complete

• **Graph Isomorphism:** Given two graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$, is there a mapping $f$ from $V_1$ to $V_2$ such that $(u,v) \in E_1$ iff $(f(u),f(v)) \in E_2$

• Important Problem !!!
• Obviously in NP: guess $f$ and verify it
• Best algorithm: $2^{\sqrt{V}}$
• Unlikely to be NP-complete
Some natural problems are not even in NP

- Can you verify quickly that a 3CNF formula is not satisfiable? What would be a certificate?

- Can you verify quickly that a graph $G$ does not have large clique? What would be a certificate?

- Can you verify quickly that a pair of graphs are not isomorphic? What would be a certificate?

- Best we know: exponential size `certificates'