COMPSCI 311: Introduction to Algorithms
Lecture 23: Reductions and NP-Complete Problems
Dan Sheldon
University of Massachusetts Amherst

Review
- P – class of problems with polytime algorithm.
- NP – class of problems with polytime certifier.

Example

<table>
<thead>
<tr>
<th>Problem (X)</th>
<th>Independent-Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance (s)</td>
<td>Graph G and number k</td>
</tr>
<tr>
<td>Algorithm (A)</td>
<td>No poly-time algorithm known</td>
</tr>
<tr>
<td>Hint (t)</td>
<td>Which nodes are in the answer?</td>
</tr>
<tr>
<td>Certifier (C)</td>
<td>Are those nodes independent and size k?</td>
</tr>
</tbody>
</table>

NP-Complete
- NP-complete = a problem \( Y \in \text{NP} \) with the property that \( X \leq_p Y \) for every problem \( X \in \text{NP} \)!

Circuit-SAT

<table>
<thead>
<tr>
<th>Circuit-SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem: Given a circuit built of AND, OR, and NOT gates with some hard-coded inputs, is there a way to set remaining inputs so the output is 1?</td>
</tr>
<tr>
<td>Proof Idea: encode arbitrary certifier ( C(s, t) ) as a circuit</td>
</tr>
<tr>
<td>Cook-Levin Theorem: In 1971, Cook and Levin independently showed that particular problems were NP-Complete.</td>
</tr>
<tr>
<td>We’ll look at Circuit-SAT as canonical NP-Complete problem.</td>
</tr>
</tbody>
</table>

Cook-Levin Theorem: Circuit-SAT is NP-Complete.
Proof Idea: encode arbitrary certifier \( C(s, t) \) as a circuit

- If \( X \in \text{NP} \), then \( X \) has a poly-time certifier \( C(s, t) \):
  - \( s \) is \text{Yes} instance \( \iff \exists t \text{ such that } C(s, t) \text{ outputs } \text{Yes} \)
  - Construct a circuit where \( s \) is hard-coded, and circuit is satisfiable if \( \exists t \) that causes \( C(s, t) \) to output \text{Yes} |
- \( s \) is \text{Yes} instance \( \iff \) circuit is satisfiable |
- Algorithm for Circuit-SAT implies an algorithm for \( X \)
A Circuit-SAT reduction

See Independent Set example in other slides

A Circuit-SAT reduction

Vertex Cover – Does $G$ have VC of size at most $k$? (Counting gadget is an example for $v_3, v_4$ only)

Proving New Problems NP-Complete

Suppose $X$ is in NP.

Fact: If $Y$ is NP-complete and $Y \leq_P X$, then $X$ is NP-complete.

Want to prove problem $X$ is NP-complete

- Check $X \in$ NP.
- Choose known NP-complete problem $Y$.
- Prove $Y \leq_P X$.

Clicker

It’s easy to show that 3-SAT $\leq_P$ Circuit-SAT. What can we conclude from this?

A. 3-SAT is NP-complete.
B. 3-SAT is in NP.
C. If there is no polynomial time algorithm for 3-SAT, then there is no polynomial time algorithm for Circuit-SAT.

Proving New Problems NP-Complete

Theorem: 3-SAT is NP-Complete.

- In NP? Yes, check satisfying assignment in poly-time.
- Can show that Circuit-SAT $\leq_P$ 3-SAT

From Circuit-SAT to 3-SAT

Fact: If $Y$ is NP-complete, $X$ is in NP, and $Y \leq_P X$, then $X$ is NP-complete.

Theorem: 3-SAT is NP-Complete.

1. In NP? Yes, check satisfying assignment in poly-time.
2. Prove by reduction from Circuit-SAT.

Example.
**Reduction: Circuit-Sat ≤_P 3-Sat**

- One variable \(x_v\) per circuit node \(v\) plus clauses to enforce circuit computations.
- Express Negation, OR, and AND nodes using several implications of the form \(A \implies B\) (which is equivalent to the clause \(\neg A \lor B\)).
- Negation node: \(x_v = \neg x_u\)
  - \(x_u \implies \neg x_v\)
  - \(\neg x_u \implies x_v\)
- OR node: \(x_v = x_u \lor x_w\)
  - \(x_u \implies x_v\)
  - \(x_w \implies x_v\)
  - \(\neg x_v \implies \neg x_u \lor \neg x_w\)
- AND node: \(x_v = x_u \land x_w\)
  - \(x_u \implies x_v\)
  - \(x_w \implies x_v\)
  - \(\neg x_v \implies \neg x_u \lor \neg x_w\)

**Clicker**

Which of the following statements is NOT true?

A. SAT ≤_P 3-SAT
B. 3-SAT ≤_P SAT
C. \(k\)-SAT ≤_P SAT for all \(k \geq 2\)
D. \(k\)-SAT is NP-complete for all \(k \geq 2\)

**NP-Complete Problems So Far**

**Theorem:** INDEPENDENTSET, VERTEXCOVER, SETCOVER, SAT, 3-SAT are all NP-Complete.

**Traveling Salesman Problem**

- TSP. Given \(n\) cities and distance function \(d(i, j)\), is there a tour that visits all cities with total distance less than \(D\)?
  - Tour: ordering of cities \(i_1, i_2, \ldots, i_n\) with \(i_1 = 1\)
  - Distance is \(\sum_{j=1}^{n-1} d(i_j, i_{j+1}) + d(i_n, 1)\)
- Applications: traveling salesman, moving robotic arms
- Let’s prove a simpler problem is NP-complete, and then use it to show TSP is NP-complete.
Hamiltonian Cycle Problem

- **HAMCYCLE** – Hamiltonian Cycle. Given directed graph \( G = (V, E) \), is there a cycle that visits each vertex exactly once?

\[
\begin{array}{c}
\text{v1} \\
\text{v2} \\
\text{v3} \\
\text{v4} \\
\text{v5} \\
\text{v6}
\end{array}
\]

- \( v_1, v_3, v_2, v_5, v_4, v_6 \) is a Hamiltonian Cycle

**Theorem.** **HAMCYCLE** is NP-Complete.

- It is in NP.
- Need to reduce from some NP-Complete problem. Which one?

**Claim.** **3-SAT \( \leq_p \) HAMCYCLE.**

Reduction has two main parts.

- Make a graph with \( 2^n \) Hamiltonian cycles, one per assignment.
- Augment graph with clauses to invalidate assignments.

**Reduction: Details**

- **n rows (bidirected paths) \( P_1, \ldots, P_n \) (one per variable)**
- **Row has 3\( m + 3 \) vertices, connected to neighbors in forward/backward direction**
- First and last vertex of row \( i \) connected to first and last of \( i+1 \).
- Source \( s \) connected to first and last of row 1.
- First and last of row \( n \) connected to \( t \).
- **Edge \((t, s)\)**
- **Skeleton has \( 2^n \) possible Hamiltonian Cycles, corresponding to truth assignments to \( x_1, \ldots, x_n \)**
  - Traverse \( P_i \) L to R \( \iff \) \( x_i = 1 \)
  - Traverse \( P_i \) R to L \( \iff \) \( x_i = 0 \)

**Reduction: Graph skeleton**

- \( x_i = 1 \iff \) traverse \( P_i \) from L \( \rightarrow \) R

**Reduction: Clause Gadgets**

- **Node \( c_j \) for clause \( C_j \) must be visited in middle of some \( P_i \)**
  - \( x_i \in C_j \implies \) can visit \( c_j \) during \( L \rightarrow R \) traversal of \( P_i \)
  - \( x_i = 1 \) satisfies \( C_j \)
  - \( x_i \in C_j \implies \) can visit \( c_j \) during \( R \rightarrow L \) traversal of \( P_i \)
  - \( x_i = 0 \) satisfies \( C_j \)

- There is a Hamiltonian cycle
  - \( \iff \) can visit all clause nodes
  - \( \iff \) there is a truth assignment that satisfies all clauses

**Reduction: High-Level**

- Correspondence between Hamiltonian cycles and truth assignments
  - \( x_i = 1 \): traverse path \( P_i \) from \( L \rightarrow R \)
  - \( x_i = 0 \): traverse path \( P_i \) from \( R \rightarrow L \)

- Node \( c_j \) for clause \( C_j \) must be visited in middle of some \( P_i \)
  - \( x_i \in C_j \implies \) can visit \( c_j \) during \( L \rightarrow R \) traversal of \( P_i \)
  - \( x_i = 1 \) satisfies \( C_j \)
  - \( x_i \in C_j \implies \) can visit \( c_j \) during \( R \rightarrow L \) traversal of \( P_i \)
  - \( x_i = 0 \) satisfies \( C_j \)

- There is a Hamiltonian cycle
  - \( \iff \) can visit all clause nodes
  - \( \iff \) there is a truth assignment that satisfies all clauses
**Proof of Correctness**

Given a satisfying assignment, construct Hamiltonian Cycle

- If $x_i = 1$ traverse $P_i$ from $L 	o R$, else $R 	o L$.
- Each $C_j$ is satisfied, so one path $P_i$ is traversed in the correct direction to "splice" $c_j$ into our cycle
- The result is a Hamiltonian Cycle

Given Hamiltonian cycle, construct satisfying assignment:
- If cycle visits $c_j$ from row $i$, it will also leave to row $i$ because of "buffer" nodes
- Therefore, ignoring clause nodes, cycle traverses each row completely from $L 	o R$ or $R 	o L$
- Set $x_i = 1$ if $P_i$ traversed $L 	o R$, else $x_i = 0$
- Every node $c_j$ visited $\Rightarrow$ every clause $C_j$ is satisfied

---

**Reduction: Clause Gadgets**

For each clause $C_j$ construct gadget to restrict possible truth assignments
- New node $c_j$
- If $x_i \in C_j$
  - Add edges $(v_i, 3j, c_j)$ and $(c_j, v_i, 3j)$
  - $c_j$ can be visited during L to R traversal of $P_i$
- If $\neg x_i \in C_j$
  - Add edges $(v_i, 3j+1, c_j)$ and $(c_j, v_i, 3j+1)$
  - $c_j$ can be visited during R to L traversal of $P_i$

**Traveling Salesman**

TSP. Given $n$ cities and distance function $d(i, j)$, is there a tour that visits all cities with total distance less than $D$?

**Theorem.** TSP is NP-Complete
- Clearly in NP.
- Reduction? From [H Amp-Cycle](#)

---

**Clicker**

We want to show that $\text{HAM-CYCLE} \leq_p \text{TSP}$. How can we do so?

Given a $\text{HAM-CYCLE}$ instance $G = (V, E)$ make TSP instance with one city per vertex and...

- A. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else $2$. Tour distance: $\leq n$?
- B. $d(v_i, v_j) = 2$ if $(v_i, v_j) \in E$, else $1$. Tour distance: $\leq n$?
- C. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else $2$. Tour distance: $\leq m$?

---

**Reduction from H Amp-Cycle to TSP**

Given $\text{HAM-CYCLE}$ instance $G = (V, E)$ make TSP instance
- One city per vertex
- $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else $2$

**Claim:** there is a tour of distance $\leq n$ if and only if $G$ has a Hamiltonian cycle
- A Hamiltonian cycle clearly gives a tour of length $n$
- A tour of length $n$ must travel $n$ hops of length 1, which corresponds to a Hamiltonian cycle

---

**HAM-Path**

Similar to Hamiltonian Cycle, visit every vertex exactly once.

**Theorem.** $\text{HAM-Path}$ is NP-Complete.

Two proofs.
- Modify 3-SAT to $\text{HAM-CYCLE}$ reduction.
- Show that $\text{HAM-CYCLE}$ reduces to $\text{HAM-Path}$