COMPSCI 311: Introduction to Algorithms
Lecture 21: Reductions and NP-Complete Problems
Marius Minea
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>Try all subsets and check (but not poly-time)</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 Problems So Far

Theorem: IndependentSet, VertexCover, SetCover, SAT, 3-SAT are all NP-Complete.

Clicker Question 1
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 ≥ 2
D: k-SAT is NP-complete for all k ≥ 2

Finding NP-Complete Problems

Want to prove problem X is NP-complete
▶ Check X ∈ NP.
▶ Choose known NP-complete problem Y.
▶ Prove Y ≤_P X.
▶ Usually suffices to do single transformation s_Y → s_X s.t.
  ▶ s_X is YES instance iff s_Y is YES instance

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, ..., i_n with i_1 = 1
  ▶ Distance is ∑_{j=1}^{n-1} d(i_j, i_{j+1}) + d(i_n, 1)

▶ Applications: traveling salesperson, 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?

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

**Ham-Cycle**

**Theorem.** **Ham-Cycle** is NP-Complete.

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

**Claim.** 3-SAT \( \leq_P \) **Ham-Cycle**.

Reduction has two main parts.

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

**Reduction: Skeleton Construction**

- \( n \) rows (bidirected paths) \( P_1, \ldots, P_n \) (one per variable)
- Row has \( 3m + 3 \) 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: Clause Gadgets**

For each clause \( C_i \) construct gadget to restrict possible truth assignments

- New node \( c_i \)
  - If \( x_i \in C_i \)
    - Add edges \((v_{i,3m+1}, c_i)\) and \((c_i, v_{i,3m+1})\)
    - \( c_i \) can be visited during L to R traversal of \( P_i \)
  - If \( \neg x_i \in C_i \)
    - Add edges \((v_{i,3m+1}, c_i)\) and \((c_i, v_{i,3m})\)
    - \( c_i \) can be visited during R to L traversal of \( P_i \)
Proof of Correctness

Given a satisfying assignment, construct Hamiltonian Cycle

- If \( x_i = 1 \) traverse \( P_i \) from \( L \rightarrow R \), else \( R \rightarrow L \).
- Each \( C_i \) is satisfied, so one path \( P_i \) is traversed in the correct direction to “splice” \( C_i \) 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 \( L \rightarrow R \) or \( R \rightarrow L \).
- Set \( x_i = 1 \) if \( P_i \) traversed \( L \rightarrow R \), else \( x_i = 0 \).
- Every node \( c_j \) visited \( \Rightarrow \) every clause \( C_j \) is satisfied.

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 Ham-Cycle

Reduction from Ham-Cycle to TSP

Given HamCycle 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. Ham-Path is NP-Complete.

Two proofs.
- Modify 3-SAT to Ham-Cycle reduction.
- Reduce from Ham-Cycle directly.

Clicker Question 2

Suppose now I want to reduce Ham-Path to Ham-Cycle. Which of the following statements is true?

A: Trivial: any cycle with all nodes is also a path with all nodes
B: Ham-Path \( \not\leq_P \) Ham-Cycle since not all paths are cycles
C: Ham-Path \( \not\leq_P \) Ham-Cycle since there are some graphs that have a Hamiltonian path but no Hamiltonian cycle
D: None of the above

NP-Complete Problems