

# Lecture 12: NP-complete graph problems

David Dill

Department of Computer Science

# Outline

---

- NP-complete problems
  - Independent Set (IS)
  - Node Cover (NC)
  - Hamiltonian circuit
    - \* Directed Hamiltonian Circuit
    - \* Undirected Hamiltonian Circuit
    - \* Traveling Salesman Problem

## Independent Set (IS)

**Def** In an undirected graph,  $G$ , a set of nodes  $I$  is independent if no two nodes in  $I$  are connected by an edge.

Input: A graph  $G$  and a bound  $1 \leq k \leq |V|$ .

Output: “Yes” iff there is an independent set of size  $k$ .

**Theorem** IS is NP-complete.

*IS is in NP.* Given a graph and  $I \subset V$ , examine the edges one at a time to make sure that no edge connects two nodes in  $I$ . With any reasonable representation of the graph and  $I$ , this would be polynomial time (probably linear in the number of edges on a conventional computer).

## Reducing from CSAT

To reduce CSAT (or 3SAT) to another problem, the reduction has to capture two crucial ideas:

- If a literal is selected, its negation is never selected.
- At least one literal is selected per clause.

These conditions can be satisfied iff there is a satisfying truth assignment for the original CSAT problem.

$\Rightarrow$ : Start with a truth assignment. There must be at least one true literal per clause.

$\Leftarrow$ : If we satisfy these conditions, we can build a truth assignment by choosing  $x_i = T$  if  $x_i$  literal is selected,  $x_i = F$  if  $\neg x_i$  is selected, and  $x_i =$  whatever if neither is selected.

## NP-hardness of IS

*IS is NP-hard.* By reduction from 3SAT.

Construction: Let  $\phi$  be a 3CNF formula with  $m$  clauses.

Create a node for each occurrence of a literal in each clause. Each node is labelled with the literal, the clause number, and the position of the literal within the clause (e.g.  $x_3[1, 3]$  represents the occurrence of  $x_3$ , in the third position in the first clause).

Main idea:  $I$  represents one literal per clause that has to be true. (There may be other true literals as well.)

Idea 1: Put *negation edges* between literals and their negations. That prevents adding  $x$  and  $\neg x$  to the IS.

Idea 2: Put *clause edges* between literals in the same clause. We're going to require  $I$  to have at least  $m$  nodes, so at least one literal has to be true for each clause.

## Proof of construction

*Claim:* The 3SAT formula is satisfiable iff there is an independent set of size  $m$ .

### **Proof**

Satisfiable  $\rightarrow$  IS of size  $m$ . If  $\phi$  is satisfiable, there is at least one truth assignment,  $A$ , that makes  $\phi$  true.  $A$  must have at least one true literal per clause. Choose a *designated literal* for each clause arbitrarily from among the true ones. The designated literals form the IS. It truly is independent because the clause edges will never connect two designated literals, since there is only one per clause; the complement edges will never connect two designated nodes, because the truth assignment will never make a literal and its complement both true.

IS of size  $m \rightarrow$  Satisfying assignment. Given an IS, we can extend it to a satisfying truth assignment. Whenever  $x_\ell[i, j]$  is in the IS, set  $x_\ell = T$  in the truth assignment. Whenever  $\neg x_\ell[i, j]$  appears in the IS, set  $x_\ell = F$  in the truth assignment. If neither appears, choose  $x_\ell = F$ . The truth assignment will be consistent because  $x_\ell$  and  $\neg x_\ell$  will never both be designated, because of the negation edges. The IS must have exactly one literal from each clause, because the clause edges require there to be *at most* one member of the IS per clause, and the size  $m$  requirement requires there to be *at least* one literal per clause.

## Node Cover (NC)

Given a graph  $G$ , a node cover is a set of nodes such that every edge in  $G$  connects to some node in  $C$ .

(Note: smaller covers are harder.)

**NC problem:** Given a graph  $G$  and a bound  $k$ , is there a node cover in  $G$  of  $k$  nodes?

**Theorem** NC is NP-complete

It is easy to check a candidate cover in PTIME: Just examine each edge to make sure one of the endpoints is in the cover.

Reduction: To solve an IS with graph  $G$  and number  $k$ , solve NC with graph  $G$  and number  $m - k$ . Then complement the solution, which is the desired independent set.

Fact:  $C$  is a node cover of  $G$  iff  $\overline{C}$  is an independent set of  $G$ .

Intuition: The IS had *at most one* node per edge. That leaves *at least one* node per edge that was not in the IS.

If the set is represented as a list, this is polynomial-time.

# Hamiltonian Circuit

A *circuit* is a path through a graph that ends where it starts.

A *Hamiltonian circuit* is a circuit that enters and exits each node exactly once.

Checking the existence of a Hamiltonian circuit is NP-complete.

Finding reductions for NP-complete problems can be very hard. HC is an example!

## Directed Hamiltonian Circuit

*Directed Hamiltonian Circuit* (DHC) is the problem of finding a Hamiltonian Circuit on a directed graph.

First, DHC is obviously in NP.

The reduction is from 3SAT to DHC.

Like many reductions, this depends on “gadgets” – subproblems with useful properties.

The first gadget is for the truth assignment. It is a big loop of gadgets for the individual variables. Suppose we have variables  $x_1 \dots x_n$  and clauses  $c_1 \dots c_m$ .

⟨⟨ Loop diagram 10.9(b) ⟩⟩

Each variable  $x_i$  has a “ladder” of nodes representing literals.

⟨⟨ Ladder diagram 10.9(a) ⟩⟩

Each ladder has  $m + 1$  levels,  $0 \dots m$ .

There are only two paths from the entry node to the exit node, depending on whether we go to  $b_{i0}$  or  $c_{i0}$  from the entry node. Crossing levels from  $c$  to  $b$  means  $x_i$  is true, and from  $b$  to  $c$  means it's false.

## DHC, continued

There is another gadget for each 3-clause:

⟨⟨ Picture of the  $I$  gadget ⟩⟩.

This gadget has the property that, if you enter at one of the top nodes, you must exit via the node beneath it, or else nodes will be isolated.

There are three entry points, corresponding to the three literals in the clause. The gadget can be entered and exited from one to three times.

## DHC, continued

The clause gadget is connected to the literals gadget by putting it in parallel with one of the level-crossing edges.

Thus, a path can cross levels by entering via the  $i$ th entry node for the  $i$ th literal in the clause, and returning via the corresponding exit node (it's sort of like a subroutine call).

If the clause literal is positive, we parallel the  $c_{i,j-1} \rightarrow b_{ij}$  edge. Otherwise, we parallel the  $b_{i,j-1} \rightarrow c_{ij}$  edge.

## DHC – How it Works

We need to prove that there is a Directed Hamiltonian Cycle iff there is a satisfying truth assignment to the original problem.

3SAT solution  $\Rightarrow$  DHC solution:

First, build a tour of all the literal gadgets. If  $x_i = T$ , visit  $b_{i0}$  first, else visit  $c_{i0}$  first.

Then, add “detours” to the clause gadgets. If clause  $c_j$  contains  $x_i$ , the gadget for clause  $c_i$  will parallel the  $c_{i,j-1} \rightarrow b_{ij}$  edge. The first tour will traverse this edge if  $x_i = T$ , so we can take a “detour” to visit all nodes in the clause gadget and return where we left off (similarly when  $x_i = F$ ).

The resulting tour will be a Hamiltonian circuit.

DHC solution  $\Rightarrow$  3SAT solution:

Start with a Hamiltonian circuit of the graph. Look at the detours to the clause gadgets, and assign the truth values to the variables based on whether  $b_{i0}$  or  $c_{i0}$  is visited first in each case.

These must make all the clauses true, because it has to visit the clause gadgets. To do that, it has to take a detour from the appropriate edge, which it can only do if there is a literal that makes the clause true.

# Undirected Hamiltonian Circuit

The usual version of HC is on undirected graphs.

To prove HC NP-complete, the book has a simple reduction from DHC (just proven NP-complete).

Start with directed graph  $G_D$  and construct undirected graph  $G_U$ .

The reduction splits each node  $v$  in  $G_D$  to three nodes  $v_0, v_1, v_2$ .  $v_1$  has just two edges, so one will have to be the entry edge and the other will have to be the exit.

DHC  $\Rightarrow$  HC:

An edge in directed graph from  $v$  to  $w$  is translated to a directed edge from  $v_2$  to  $w_0$ .

A circuit  $v_0, v_1, \dots, v_n$  in  $G_D$  can obviously be made into a circuit  $v_0^{(0)}, v_0^{(1)}, v_0^{(2)}, v_1^{(0)}, v_1^{(1)}, v_1^{(2)}, \dots, v_n^{(0)}, v_n^{(1)}, v_n^{(2)}$  in  $G_U$ .

## Rest of HC Proof

HC  $\Rightarrow$  DHC:

Any Hamiltonian circuit on  $G_U$  must visit  $v^{(0)}, v^{(1)}, v^{(2)}$  in sequence in forward or reverse order, because otherwise  $v_i^{(1)}$  would be visited 0 times or more than once.

Suppose  $v^{(0)}, v^{(1)}, v^{(2)}$  are visited in forward order for some  $i$ . The next node in the sequence must be  $w^{(0)}$  for some  $w$  because all edges exiting  $v^{(2)}$  go to a node of this form. But then the circuit will have to visit  $w^{(1)}$  and  $w^{(2)}$  in that order.

But then we can easily construct a circuit in  $V_D$  from this circuit, since the directed edges will be consistently followed.

# Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a famous NP-complete problem.

The problem is: Given an undirected graph  $G$  where each edge has a non-negative real-number weight, is there a circuit of weight  $w$  or less?

Given a solution, we can easily check that it visits each city once and add up the weights to make sure they are  $\leq k$ , so it is in NP.

To prove NP-Hard, how do we reduce TSP to HC? [We don't; that's the wrong direction.]

Reducing HC to TSP is easy: Given HC problem  $G$ , set edge weights to 1, and make the bound  $w =$  number of nodes.