Formal Languages, Automata and Computation

Proving Problems NP-Complete

SUMMARY

- Complexity Classes: P and NP
- Polynomial time reducibility
- Satisfiability Problem (*SAT*)
    - CNF, 3CNF Forms
    - 3*SAT* Problem
- NP-Completeness
- NP-Completeness of the *SAT* problem
    - Reduction from accepting computation histories of nondeterministic TMs to a *SAT* formula such that
        - A polynomial time NTM accepts w iff the corresponding *SAT* formula has a satisfying assignment.
- 3*SAT* is NP-Complete.

Showing Problems NP-complete

- Remember that in order to show a language X to be NP-complete we need to show
  1. X is in NP, and
  2. Every Y in NP is polynomial time reducible to X,
- Part 1 is (usually) easy. You argue that there is polynomial time verifier for X, which, given a solution (certificate), will verify in polynomial time, that, it is a solution.
- For part 2, pick a known NP-complete problem Z
  1. We already know that all problems Y in NP reduce to Z in polynomial time.
  2. We produce a polynomial time algorithm that reduces all instances of Z to some instance of X.
  3. So $Y \leq_P Z$ and $Z \leq_P X$ then $Y \leq_P X$.

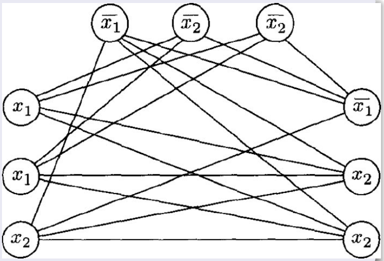
Showing Problems NP-Complete

Theorem

CLIQUE is NP-complete.

Proof

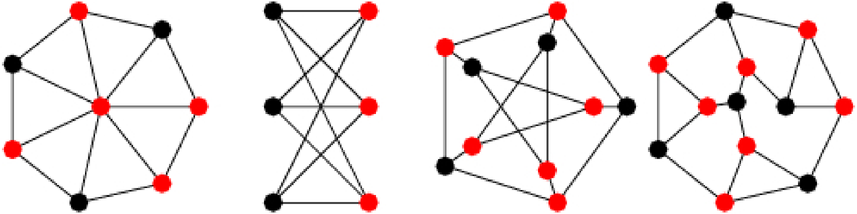
1. We know $3SAT$ is NP-complete.
2. We know that $3SAT \leq_P CLIQUE$.
3. Hence *CLIQUE* is NP-complete.



The Vertex Cover Problem

Definition – Vertex Cover

Given an undirected graph G, a vertex cover of G is a subset of the nodes where every edge of G touches one of those nodes.



- $\mathit{VERTEX\text{-}COVER} = \{\langle G, k\rangle \mid G$ is an undirected graph that has a k-node vertex cover$\}$.

The Vertex Cover Problem

Theorem

VERTEX-COVER is NP-complete.

Proof Idea

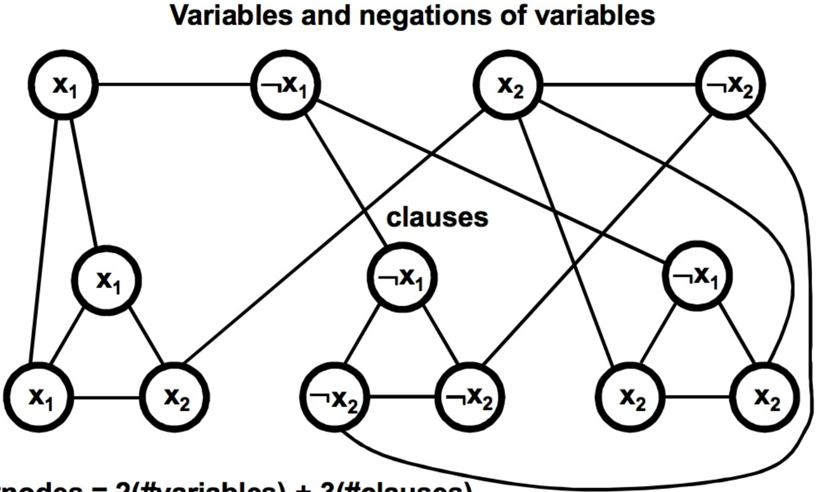
- Show *VERTEX-COVER* is in NP.
  - Easy, the certificate is the vertex cover of size k.
- We reduce an instance of $3SAT$, ϕ, to a graph G and an integer k so that ϕ is satisfiable whenever G has a vertex cover of size k.
- We employ a concept called gadgets, groups of nodes with specific functions, in the graph.
  - Variable gadgets – representing literals
  - Clause gadgets – representing clauses

THE VERTEX COVER PROBLEM

- Let ϕ be a 3-cnf formula with *m* variables and *l* clauses.
- We construct in polynomial-time, an instance of $\langle G, k \rangle$ where $k = m + 2l$.
  - For each variable x in ϕ, we add two nodes to G labeled x and $\overline{x}$, connected by an edge (variable gadget).
  - For every clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ in ϕ, we add 3 nodes labeled ℓ_1, ℓ_2 and ℓ_3, with edges between every pair so that they form a triangle (clause gadget)
  - We add an edge between any two identically labelled nodes, one from a variable gadget and one from a clause gadget.

The Vertex Cover Problem

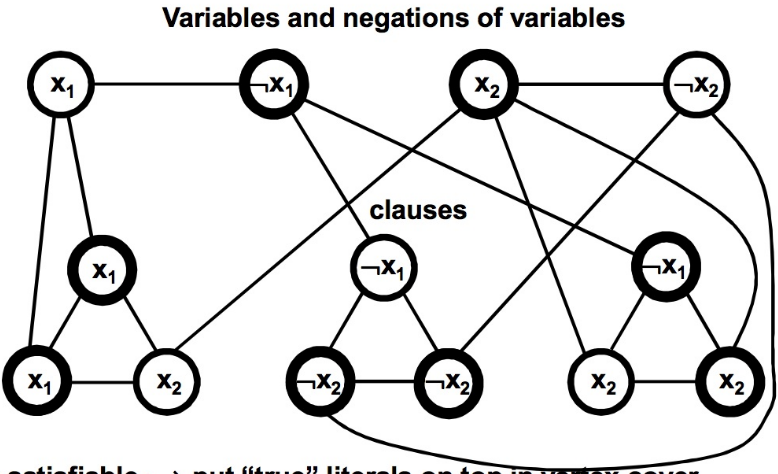
$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$



#nodes = 2(#variables) + 3(#clauses)

The Vertex Cover Problem

$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$

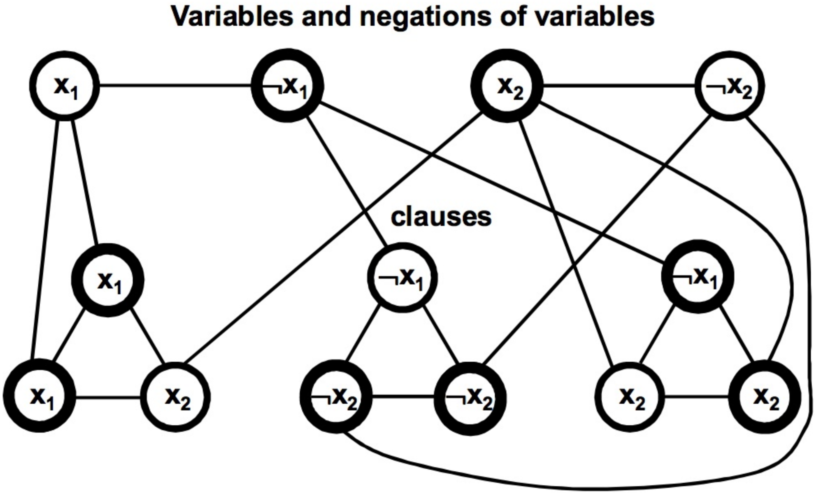


ϕ satisfiable $\Longrightarrow$ put "true" literals on top in vertex cover

For each clause, pick a true literal and put other 2 in vertex cover

THE VERTEX COVER PROBLEM

$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$



$k = 2(\#\text{clauses}) + (\#\text{variables})$

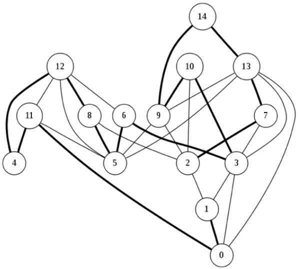
The Hamiltonian Path Problem

Definition - Hamiltonian Path

(Recall that) A Hamiltonian path in a directed graph G is a directed path that goes through each node exactly once.

Definition Hamiltonian Path Problem

$HAMPATH = \{\langle G, s, t\rangle \mid G$ is a directed graph with a Hamiltonian path from s to $t\}$.



The Hamiltonian Path Problem

Theorem

HAMPATH is NP-complete.

Proof Idea

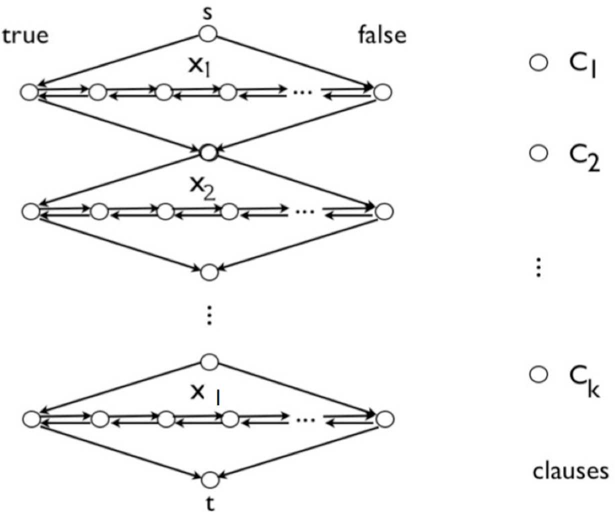
- We show $3SAT \leq_P HAMPATH$.
- We again use gadgets to represent the variables and clauses.
- For a given 3-cnf formula with k clauses

$$\phi = \underbrace{(a_1 \vee b_1 \vee c_1)}_{C_1} \wedge \underbrace{(a_2 \vee b_2 \vee c_2)}_{C_2} \wedge \cdots \wedge \underbrace{(a_k \vee b_k \vee c_k)}_{C_k}$$

where each a_i, b_i or c_i is a literal x or $\overline{x}$. We have l variables $x_1, x_2, \ldots x_l$.

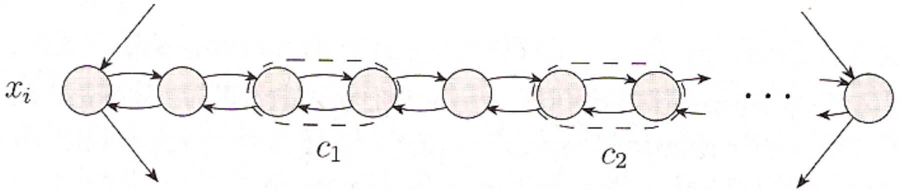
The Hamiltonian Path Problem

- 1-node gadgets for clauses
- Diamond-shaped gadgets for variables



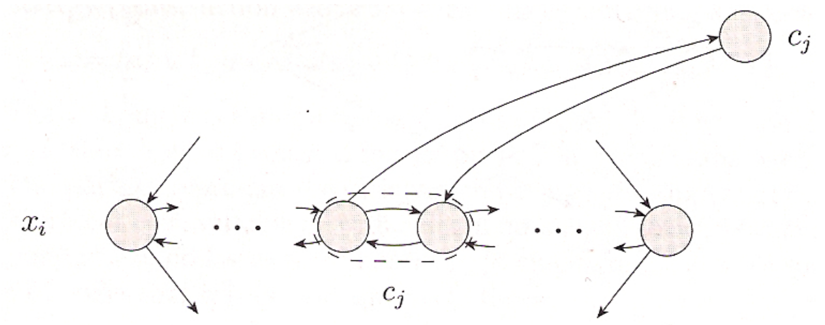
The Hamiltonian Path Problem

- The middle spine in each diamond has $3k + 3$ nodes.
  - 3 nodes per clause + 1 to isolate them from the two literal nodes and 2 nodes on each side for the literals x_i, $\overline{x_i}$.



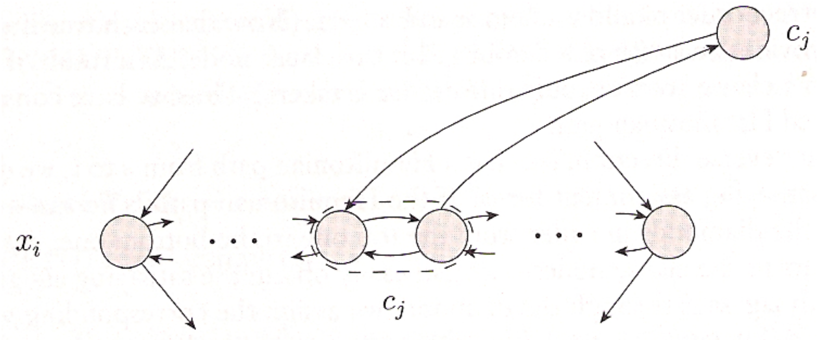
The Hamiltonian Path Problem

- If x_i appears in clause c_j, we add two edges from j^{th} group in the spine to the j^{th} clause node in the i^{th} diamond.



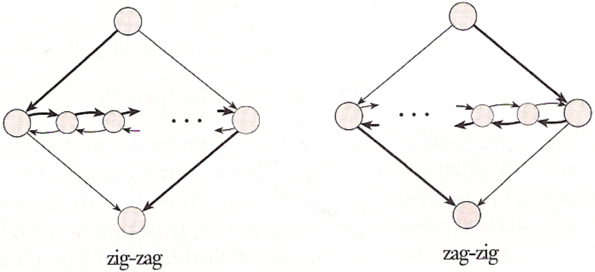
The Hamiltonian Path Problem

- If $\overline{x_i}$ appears in clause c_j, we add two edges from j^{th} group in the spine to the j^{th} clause node in the i^{th} diamond, but in the reverse direction.



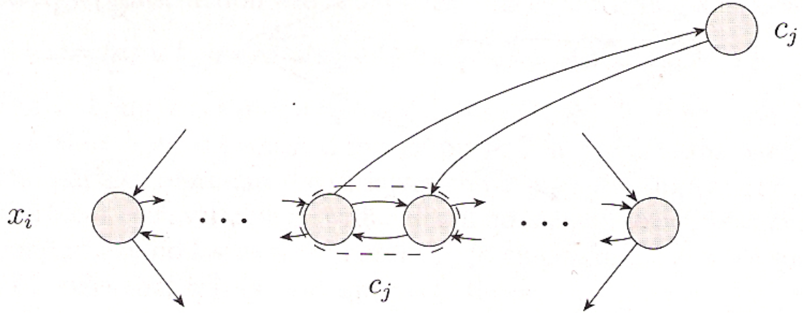
THE HAMILTONIAN PATH PROBLEM

- Suppose ϕ is satisfiable.
- Ignoring the clause nodes, we note that the Hamiltonian path
  - starts at s
  - goes through each diamond
  - ends up at t.
- In diamond i, it either goes left-to-right or right-to-left depending on the truth value of variable x_i.



The Hamiltonian Path Problem

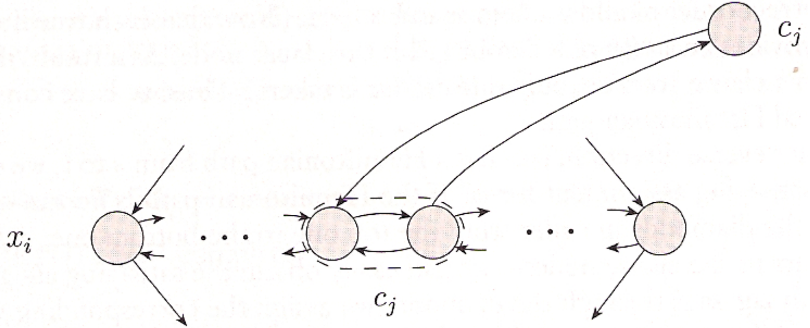
- The clause nodes can be incorporated into the path using the detours we provided.
- So if x_i is true and is in clause c_j, we can take a detour to node for c_j and back to the spine in the right direction.



- Note that each detour is optional but we have to incorporate c_j only once.

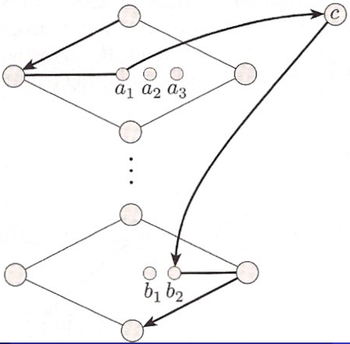
The Hamiltonian Path Problem

- The clause nodes can be incorporated into the path using the detours we provided.
- So if $\overline{x_i}$ is true and is in clause c_j, we can take a detour to node for c_j and back to the spine in the reverse direction.



THE HAMILTONIAN PATH PROBLEM

- How about the reverse direction? If G has a Hamiltonian path then ϕ has a satisfying assignment?
- If the path is normal, that is, it goes through from s zigzagging through the diamonds, then clearly there is a satisfying assignment.
- The following case can not happen!



The Undirected Hamiltonian Path

Definition Hamiltonian Path Problem

$UHAMPATH = \{\langle G, s, t \rangle \mid G$ is an undirected graph with a Hamiltonian path from s to $t\}$.

Theorem

UHAMPATH is NP-complete.

Proof Idea

- We reduce *HAMPATH* to *UHAMPATH*.
- All nodes except s and t in the directed graph G, map to 3 nodes in the undirected graph G'.
- G has a Hamiltonian path $\Leftrightarrow$ G' has an undirected Hamiltonian path.

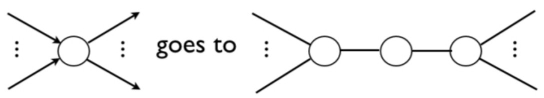
THE UNDIRECTED HAMILTONIAN PATH

THEOREM

UHAMPATH is NP-complete.

PROOF

- s in G maps to s^{out} in G'.
- t in G maps to t^{in} in G'.
- Any other node u_i maps to $u_i^{in}, u_i^{mid}, u_i^{out}$ in G'.
  - All arcs coming to u_i in G become edges incident on u_i^{in} in G'.
  - All arcs going out from u_i in G become edges incident on u_i^{out} in G'.



The Undirected Hamiltonian Path

- Note that if

$$s, u_1, u_2, \ldots, u_k, t$$

is a Hamiltonian path in G then

$$s^{out}, u_1^{in}, u_1^{mid}, u_1^{out}, u_2^{in}, u_2^{mid}, u_2^{out} \ldots, u_k^{out}, t^{tin}$$

is a Hamiltonian path in G'.

- Any Hamiltonian path between s^{out} and t^{in}, must go through the triple of nodes except for the start and end nodes.

The Subset Sum Problem

$$SUBSET\text{-}SUM = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_m\} \text{ and for some } \{y_1, \ldots, y_n\} \subseteq S, \sum y_i = t\}$$

Theorem

SUBSET-SUM is NP-complete.

Proof Idea

- We reduce $3SAT$ to an instance of the *SUBSET-SUM* problem with a set S and a bound t,
  - so that if a formula ϕ has a satisfying assignment,
  - then S has a subset T that adds to t
- We already know that *SUBSET-SUM* is in NP.

The Subset Sum Problem

- Let ϕ be a formula with variables $x_1, x_2, \ldots, x_l$ and clauses $c_1, \ldots, c_k$.
- We compute $m = 2 \times l + 2 \times k$ (large) numbers from ϕ and a bound t
- Such that when we choose the numbers corresponding to the literals in the satisfying assignment, they add to t.

The Subset Sum Problem

S for $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

The Subset Sum Problem

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

- We choose one of the numbers y_i if $x_i = 1$, or z_i if $x_i = 0$.
- The left part of t will add up the right number.
- The right side columns will at least be 1 each
- We take enough of the g and h's to make them add up to 3.