NP-Completeness

Huan Long

Shanghai Jiao Tong University

Acknowledgements

Part of the slides comes from a similar course given by Prof. Yijia Chen.

`http://basics.sjtu.edu.cn/~chen/`

Textbook

Introduction to the theory of computation
Michael Sipser, MIT
Third edition, 2012

Outline

The NP-Completeness

Additional NP-complete problems

The NP-Completeness

In 1970s, Stephen Cook and Leonid Levin discovered certain problems in NP whose individual complexity is related to that of the entire class.

If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable.

These problems are called NP-complete

Satisfiability Problem

- Boolean variables are assigned to TRUE(1) or FALSE(0).
- Boolean operations are AND, OR, and NOT.
- A Boolean formula is an expression involving Boolean variables and operations.
- A Boolean formula is satisfiable if some assignment makes the formula evaluate to 1.
- The satisfiability problem is to test whether a Boolean formula is satisfiable, i.e.,

$$\text{SAT} = \{\langle\varphi\rangle \mid \varphi \text{ is a satisfiable Boolean formula}\}.$$

Theorem

SAT$\in$P if and only if P=NP.

Definition

A function $f : \Sigma^* \to \Sigma^*$ is a <u>polynomial time computable function</u> if some polynomial time Turing machine exists that halts with just $f(w)$ on its tape, when started on any input w.

Definition

Let $A, B \subseteq \Sigma^*$. Then A is <u>polynomial time mapping reducible</u>, or simply <u>polynomial time reducible</u>, to B, written $A \leq_P B$, if a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ exists, where for every w

$$w \in A \Leftrightarrow f(w) \in B.$$

The function f is called the <u>polynomial time reduction</u> of A to B.

Theorem

If $A \leq_P B$ *and* $B \in$*P, then* $A \in$*P.*

3SAT

- A literal is a Boolean variable or a negated Boolean variable.
- A clause is several literals connected with $\vee$s.
- A Boolean formula is in conjunctive normal from, called a cnf-formula, if it comprises several clauses connected with $\wedge$s.
- A Boolean formula is a 3cnf-formula if all the clauses have three literals.
- Let

$$\text{3SAT} = \{\langle\varphi\rangle \mid \varphi \text{ is a satisfiable 3cnf-formula}\}.$$

Theorem

3SAT is polynomial time reducible to CLIQUE.

Proof (1)

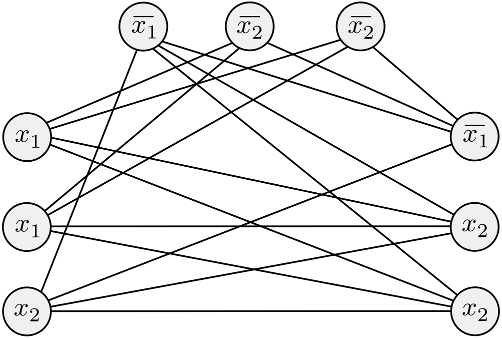
Let φ be a formula with k clauses such as

$$\varphi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots (a_k \vee b_k \vee c_k).$$

The reduction generates a string $\langle G, k \rangle$.

1. The nodes in G are organized into k groups of three nodes $t_1, \ldots, t_k$. Each triple corresponds to one of the clauses, and each node in a triple corresponds to a literal in the associated clauses.
2. The edge of G connect all but two types of pairs of nodes in G.
   - No edge is present between nodes in the same triple.
   - No edge is present between two nodes with contradictory labels, e.g., x_2 and $\overline{x_2}$.

Proof (2)



$$\varphi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge \cdots \wedge (\overline{x_1} \vee x_2 \vee x_2).$$

Definition

A language B is NP-complete if it satisfies two conditions:

1. B is in NP, and
2. every A in NP is polynomial time reducible to B.

Theorem

If B is NP-complete and $B \in$P, then P=NP.

Theorem

If B is NP-complete and $B \leq_P C$ for some C in NP, then C is NP-complete.

Theorem

SAT is NP-complete.

Proof (1)

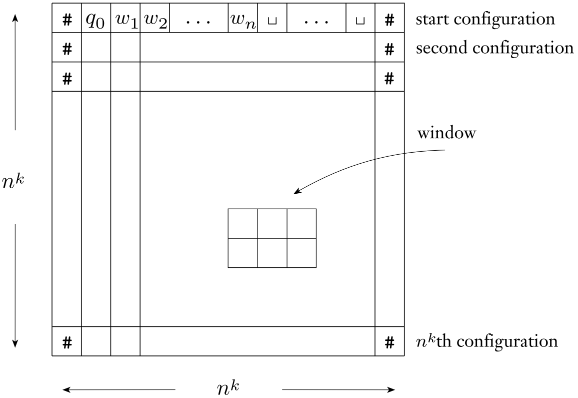
SAT is in NP, since a nondeterministic polynomial time Turing machine can

1. guess an assignment to a given formula φ,
2. accept if the assignment satisfies φ.

Proof (2)

Let N be an NTM that decides a language A in time n^k for some $k \in \mathbb{N}$. We show $A \leq_p$ SAT.

A tableau for N on w is an $n^k \times n^k$ table whose rows are the configurations of the branch of the computation of N on input w.



Proof (3)

- We assume that each configuration starts and ends with a $\#$ symbol. Therefore, the first and last columns of a tableau are all $\#$s.
- The first row of the tableau is the starting configuration of N on w, and each row follows the previous one according to N's transition function.
- A tableau is accepting if any row of the tableau is an accepting configuration.
- Every accepting tableau for N on w corresponds to an accepting computation branch of N on w. Thus the problem of determining whether N accepts w is equivalent to the problem of determining whether an accepting tableau for N on w exits.

Proof (4)

On input w, the reduction produces a formula φ.

1. Let Q and Γ be the state set and tape alphabet of N. We set

$$C = Q \cup \Gamma \cup \{\#\}.$$

2. For each $i, j \in [n^k]$ and for each $s \in C$, we have a variable $x_{i,j,s}$.
3. Each of the $\left(n^k\right)^2$ entries of a tableau is called a cell.
4. If $x_{i,j,s}$ takes on the value 1, it means that the cell in row i and column j contains an s.

We represent the contents of the cells with the variable of φ.

Proof (5)

We design φ so that a satisfying assignment to the variables does correspond to an accepting tableau for N for w:

$$\varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{move}} \wedge \varphi_{\text{accept}}.$$

Proof (6)

$$\varphi_{\text{cell}} = \bigwedge_{i,j \in [n^k]} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right].$$

Proof (7)

$$\begin{aligned}\varphi_{\text{start}} \quad = \quad & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_1} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}.\end{aligned}$$

Proof (8)

$$\varphi_{\text{accept}} = \bigvee_{i,j \in [n^k]} x_{i,j,q_{\text{accept}}}.$$

Proof (9)

Finally, formula φ_{move} guarantees that each row of the tableau corresponds to a configuration that legally follows the preceding row's configuration according to N's rules.
It does so by ensuring that each 2 window of cells is legal.

We say that a 2×3 windows is legal if that window does not violate the actions specified by N's transition function.

Proof (10)

Assume that:

- When in state q_1 with the head reading an a, N writes a b, stays in state q_1, and moves right.
- When in state q_1 with the head reading a b, N nondeterministically either
  1. writes a c, enters q_2, and moves to the left, or
  2. writes an a, enters q_2, and moves to the right.

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

(c)

a	a	q_1
a	a	b

(d)

#	b	a
#	b	a

(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b

Legal moves

Proof (11)

Assume that

- When in state q_1 with the head reading an a, N writes a b, stays in state q_1, and moves right.
- When in state q_1 with the head reading a b, N nondeterministically either
  1. writes a c, enters q_2, and moves to the left, or
  2. writes an a, enters q_2, and moves to the right.

(a)

a	b	a
a	a	a

(b)

a	q_1	b
q_2	a	a

(c)

b	q_1	b
q_2	b	q_2

Illegal moves

Proof (12)

If the top row of the tableau is the start configuration and every window in the tableau is legal, each row of the tableau is a configuration that legally follows the preceding one.

$$\varphi_{\text{move}} = \bigwedge_{1 \le i,j < n^k} \text{the } (i,j)\text{-window is legal.}$$

We replace "the (i,j)-window is legal " by

$$\bigvee_{\substack{a_1,\ldots a_6 \\ \text{is a legal window}}} \Big(x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3}$$

$$\wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6} \Big).$$

Corollary

3SAT is NP-complete.

Proof (1)

3SAT∈NP is clear.
To show hat every problem in NP can be reduced to 3SAT, we modify the previous reduction to SAT,recall

$$\varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{move}} \wedge \varphi_{\text{accept}},$$

where

$$\begin{aligned}
\varphi_{\text{cell}} &= \bigwedge_{i,j\in[n^k]} \left[\left(\bigvee_{s\in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t\in C\\ s\neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right] \\
\varphi_{\text{start}} &= x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_1} \wedge \ldots \wedge x_{1,n+2,w_n} \\
&\quad \wedge x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \\
\varphi_{\text{move}} &= \bigwedge_{1\leq i,j<n^k} \quad \bigvee_{\substack{a_1,\ldots a_6 \\ \text{is a legal window}}} (x_{i,j-1,a_1} \ldots \wedge x_{i+1,j+1,a_6}) \\
\varphi_{\text{accept}} &= \varphi_{\text{accept}} = \bigvee_{i,j\in[n^k]} x_{i,j,q_{\text{accept}}}
\end{aligned}$$

Proof (2)

The formula is almost in conjunctive normal form, except

$$\varphi_{\text{move}} = \bigwedge_{1 \le i,j < n^k} \bigvee_{\substack{a_1,\ldots a_6 \\ \text{is a legal window}}} (x_{i,j-1,a_1} \ldots \wedge x_{i+1,j+1,a_6})$$

Recall the distributive laws:

$$(a_{1,1} \vee \ldots a_{1,n_1}) \wedge (a_{2,1} \vee \ldots \vee a_{2,n_2}) = \bigvee_{i \in [n_1], j \in [n_2]} a_{1,i} \wedge a_{2,j}.$$

Therefore $\bigvee_{\substack{a_1,\ldots a_6 \\ \text{is a legal window}}} (x_{i,j-1,a_1} \ldots \wedge x_{i+1,j+1,a_6})$ is equivalent to an cnf-formula of size at most

$$|C|^6 = \mathcal{O}(1),$$

Where recall $C = Q \cup \Gamma \cup \{\#\}$.

Proof (3)

Now we need to convert the formula in cnf to one with three literals per clause:

1. In each clause that currently has one or two literals, we replicate one of the literals until the total number is three.
2. If a clause contains $\ell > 3$ clauses

$$(a_1 \vee a_2 \vee \ldots \vee a_\ell),$$

   We replace it with the $\ell - 2$ clauses

$$(a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee z_2) \wedge (\overline{z_2} \vee a_3 \vee z_3) \wedge \ldots \wedge (\overline{z_{\ell-3}} \vee a_{\ell-1} \vee a_\ell).$$

Additional NP-complete problems

Corollary

CLIQUE is NP-complete.

Vertex cover

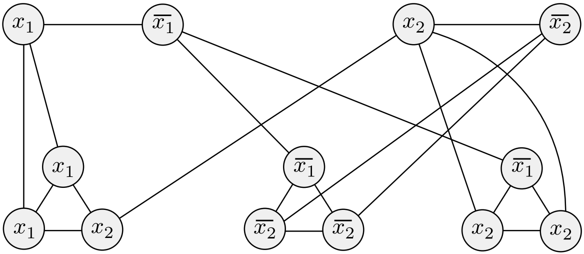
If G is an undirected graph, a vertex cover of G is a subset of the nodes where every edge of G touches one of those nodes.

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

Theorem

VERTEX-COVER is NP-complete.

$$\varphi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2).$$



The Hamiltonian path problem

The Hamiltonian path problem, i.e., *HAMPATH*, asks whether the input graph contains a path from s to t that goes through every node exactly once.

Theorem

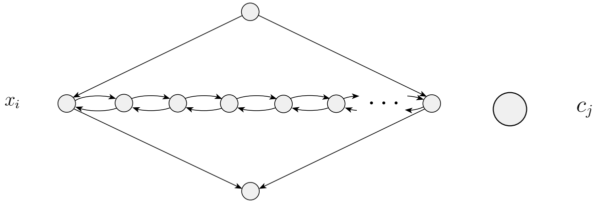
HAMPATH is NP-complete.

Proof (1)

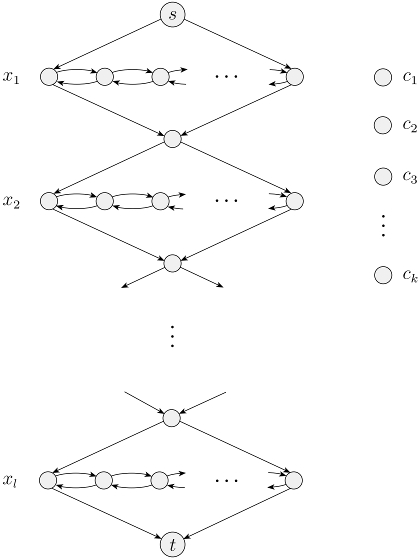
We show 3SAT$\leq_P$ HAMPATH.
Let

$$\varphi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_k \vee b_k \vee c_k).$$

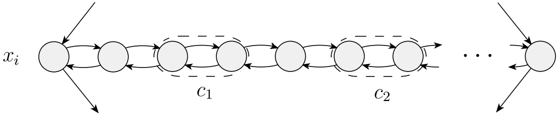
We represent each variable x_i with a diamond-shaped structure, and each clause as a single node.



Proof (2)

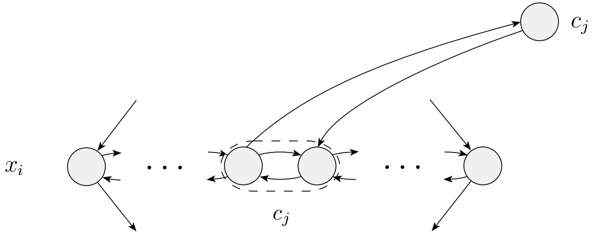


Proof (3)



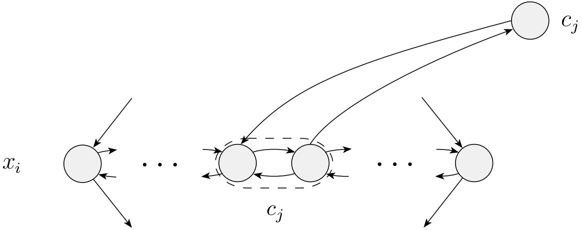
The horizontal nodes in a diamond structure

Proof (4)



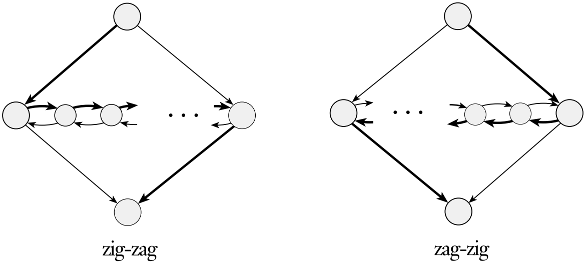
The additional edges when clause c_j contains x_i

Proof (5)



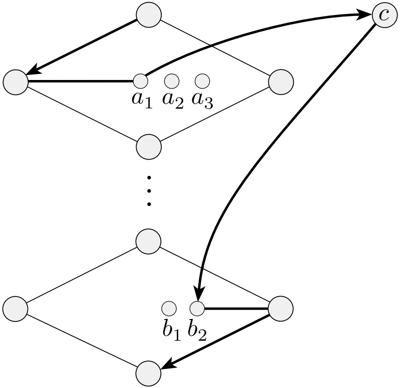
The additional edges when clause c_j contains $\overline{x_i}$

Proof (6)



- If x_i is assigned TRUE, the path zig-zags through the corresponding diamond.
- If x_i is assigned FALSE, the path zag-zigs.

Proof (7)



The above situation cannot occur.

The undirected Hamiltonian path problem

UHAMPATH, asks whether the undirected graph contains a path from s to t that goes through every node exactly once.

Theorem

UHAMPATH is NP-complete.

Proof

We show HAMPATH $\leq_P$ UHAMPATH. Let G be a directed graph with nodes s and t.

1. Let u be a node in G with $s \neq u \neq t$. We replace it by a path of length 3
$$u^{\text{in}} - u^{\text{mid}} - u^{\text{out}}.$$
2. s and t in G are replaced by $s^{\text{out}} = s'$ and $t^{\text{in}} = t'$.
3. If there is an edge from u to v in G, then in G' add an edge
$$u^{\text{out}} - v^{\text{in}}.$$

It is easy to conclude

$$\langle G, s, t \rangle \in \text{HAMPATH} \;\Leftrightarrow\; \langle G', s', t' \rangle \in \text{UHAMPATH}.$$

The subset-sum problem

Recall

$$\text{SUBSET-SUM} = \left\{ \langle S, t \rangle \mid \begin{array}{l} S = \{x_1, \ldots, x_k\}, \text{ and for some} \\ \{y_1, \ldots, y_\ell\} \subset S, \text{ we have } \sum_{i \in [\ell]} y_i = t \end{array} \right\}.$$

Theorem

SUBSET-SUM is NP-complete.

Proof (1)

We show 3SAT$\leq_P$SUBSET-SUM. Let φ be a Boolean formula with variables $x_1, \ldots, x_\ell$ and clauses $c_1, \ldots, c_k$.

1. S consists of the numbers $y_1, z_1, \ldots, y_\ell, z_\ell$ and $g_1, h_1, \ldots, g_k, h_k$.
2. For each x_i, we have two numbers y_i and z_i, where y_i for the positive and z_i for the negative literals.
3. The decimal representation of these numbers is in two parts.
   - The left-hand part comprises a 1 followed by $\ell - i$ 0s.
   - The right-hand part contains one digit for each clause, where the digit of y_i in column c_j is 1 if clause c_j contains literal x_i, and the digit of z_i in column c_j is 1 if clause c_j contains literal $\overline{x_i}$.
4. The target $t = \underbrace{1\ldots1}_{\ell \text{ times}}\underbrace{3\ldots3}_{k \text{ times}}$.

Proof (2)

	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

$$\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \ldots) \wedge \cdots \wedge (\overline{x_3} \vee \ldots \vee \ldots).$$