

## Introduction to NP-Completeness

Written and copyright © by Jie Wang <sup>1</sup>

We use time-bounded (deterministic and nondeterministic) Turing machines to study computational complexity of decision problems. In this model, a Turing machine has two different halting states, called the *accepting* state and the *rejecting* state, respectively. We say that a Turing machine  $M$  is bounded by time  $T$  if for any input  $x$ ,  $M$  on  $x$  always halts in  $T(|x|)$  moves (steps). If  $M$  is deterministic and  $M$  on input  $x$  halts at the accepting state, then we say that  $x$  is accepted by  $M$ . If  $M$  is nondeterministic and there exists at least one computation path of  $M$  on input  $x$  that halts on the accepting state, then we say that  $x$  is accepted by  $M$ . We are particularly interested in polynomial-time bounds for the following reasons.

1. All problems in practice that are solvable efficiently can be solved by deterministic Turing machines in polynomial time.
2. Polynomials are closed under addition, multiplication, and composition.

Let

$$\begin{aligned} P &= \{D \mid D \text{ is solvable by a DTM in polynomial time}\} \\ NP &= \{D \mid D \text{ is solvable by an NTM in polynomial time}\}. \end{aligned}$$

Problems in  $P$  are considered tractable. Clearly,  $P$  is included in  $NP$ . Many computational problems, while being in  $NP$ , are not known to be in  $P$ . It is the central open problem in computer science whether all problems in  $NP$  are tractable; namely, whether  $P = NP$ . While there is no known mathematical proof in either direction at this point, it is commonly believed that  $P \neq NP$ .

We use the notion of  $NP$ -completeness to measure the hardest problems in  $NP$ . (Remark:  $NP$ -completeness is a worst-case measure.) First, we define the notion of polynomial-time reductions.

**Definition 1** Let  $A$  and  $B$  be two decision problems. If there is a polynomial-time computable reduction  $f$  such that  $A \leq_m B$  via  $f$ , namely,  $\forall x : x \in A$  iff  $f(x) \in B$ , written as  $A \leq_m^p B$ , then we say that  $A$  is *polynomial-time reducible* to  $B$ .

The following proposition is straightforward (its proof is left to the reader).

**Proposition 2** 1. *Polynomial-time reductions are closed for  $P$ . Namely, if  $A \leq_m^p B$  and  $B \in P$ , then  $A \in P$ .*

---

<sup>1</sup>Written in Spring 2003. First update in Fall 2003. Second update in Fall 2006.

2. *Polynomial-time reductions are transitive. Namely, if  $A \leq_m^p B$  and  $B \leq_m^p C$ , then  $A \leq_m^p C$ .*

Proposition 2(1) indicates that, in the context of polynomial-time solvability, if  $A \leq_m^p B$ , then  $A$  is not harder than  $B$ . So the hardest problem in  $NP$  is one that can be reduced to from any problem in  $NP$ . We call such a problem an  $NP$ -complete problem.

**Definition 3** A problem  $D$  is  $NP$ -complete if (1)  $D \in NP$  and (2)  $\forall A \in NP : A \leq_m^p D$ .

Thousands of problems have been shown  $NP$ -complete. In the present note we will look at four classic  $NP$ -complete problems; they are bounded halting (BH), bounded tiling (BT), Boolean satisfiability (SAT), and 3-Satisfiability (3SAT).

## 1 Bounded Halting

Recall that the halting problem defined below is Turing-acceptable but not decidable.

$$H = \{\langle M, x \rangle \mid M \text{ is a TM and } M \text{ on } x \text{ halts}\}.$$

We can further show that every r.e. language is reducible to  $H$ . Let  $A$  be an r.e. set. Then there is a TM  $M_A$  such that  $M_A$  accepts  $A$ . Construct a reduction  $f$  by  $f(x) = \langle M_A, x \rangle$ . Clearly,  $f$  is a recursive function. We have  $\forall x : x \in A$  iff  $M_A$  halts on  $x$  iff  $f(x) \in H$ . This result indicates that  $H$  is complete for all r.e. sets; in other words,  $H$  is the hardest problem in r.e. sets.

### BOUNDED HALTING PROBLEM (BH)

*Instance.* A (nondeterministic) Turing machine  $M$ , a binary string  $x$ , and a unitary notion  $0^n$  representing positive integer  $n$ .

*Question.* Does  $M$  accept  $x$  in  $n$  steps?

For convenience we use  $BH$  to denote the positive instances of Bounded Halting; namely,

$$BH = \{\langle M, x, 0^n \rangle \mid M \text{ accepts } x \text{ in } n \text{ steps}\}.$$

**Theorem 4**  $BH$  is  $NP$ -complete.

**Proof.** It is straightforward to show that  $BH \in NP$ , for  $|\langle M, x, 0^n \rangle| = \Theta(|\langle M \rangle| + |x| + n)$  (the detailed proof is left to the reader). We will now polynomially reduce every problem in  $NP$  to  $BH$ . Let  $A \in NP$ . Then there exists a nondeterministic TM  $M_A$  such that  $M_A$  accepts  $A$  in  $p$  time, where  $p$  is a fixed polynomial. Construct a reduction  $f$  by

$$f(x) = \langle M_A, x, 0^{p(|x|)} \rangle.$$

Then  $\forall x : x \in A$  iff  $M_A$  accepts  $x$  in  $p(|x|)$  time iff  $\langle M_A, x, 0^{p(|x|)} \rangle \in BH$  iff  $f(x) \in BH$ . Since  $M_A$  is a fixed TM and  $p$  is a polynomial, producing  $\langle M_A, x, 0^{p(|x|)} \rangle$  from  $x$  can be carried out in time polynomial of  $|x|$ . Hence,  $f$  is polynomial-time computable. Thus,  $A \leq_m^p BH$ . This completes the proof. ■

## 2 Bounded Tiling

A tile is a square with a symbol on each side. We assume that tiles may not be rotated or turned over, and that there is an unlimited supply of each tile. By tiling of an  $n \times n$  square it means to select and arrange  $n^2$  tiles to cover the square so that the symbols on the common sides of adjacent tiles are the same. The size of a tile is the length of the binary representation of the four symbols at each edge in a fixed order.

### BOUNDED TILING PROBLEM (BT)

*Instance.* A finite set of tiles  $T$ , a unitary notation  $0^n$  representing positive integer  $n$ , and a sequence  $S = s_1 s_2 \dots s_k$  ( $k \leq n$ ) of tiles that match each other, *i.e.* the symbol on the right side of  $s_i$  is the same as that on the left side of  $s_{i+1}$ . (The size of the instance is  $n$  plus the summation of the sizes of all tiles in  $T$  plus the summation of the sizes of all members in  $S$ .)

*Question.* Can  $S$  be extended to tile an  $n \times n$  square using tiles from  $T$ ?

Denote by  $BT$  the positive instances of bounded tiling, namely,

$$BT = \{ \langle T, S, 0^n \rangle \mid S \text{ can be extended to tile an } n \times n \text{ square using tiles from } T \}.$$

**Theorem 5** *Bounded tiling is NP-complete.*

**Proof.** That  $BT \in NP$  is straightforward (the reader can easily fill in the details). We will now polynomially reduce every problem in  $NP$  to  $BT$ . Let  $A \in NP$ . Then there is a (nondeterministic) Turing machine  $M_A = (Q, \Sigma, \delta, q_0, \{h_a, h_r\})$  that accepts  $A$  in polynomial time. Without loss of generality, we assume that  $\Sigma = \{0, 1, B\}$ , and all computation paths of  $M_A$  on input  $x$  have length strictly less than  $p(|x|)$ , where  $p$  is a fixed polynomial. Here  $Q$  is the finite set of states of  $M_A$ ,  $\delta$  the transition function,  $q_0$  the initial state,  $h_a$  the accepting state,  $h_r$  the rejecting state, and  $B$  the blank symbol. We construct the following set  $T(M_A)$  of legal tiles:

for  $\alpha \in \{0, 1, B\}$  :



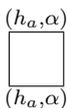
for  $\alpha, \beta, \gamma \in \{0, 1, B\}, q \in Q - \{h_a, h_r\}, p \in Q, \delta(q, \alpha) = (p, \beta, R)$ :



for  $\alpha, \beta, \gamma \in \{0, 1, B\}, q \in Q - \{h_a, h_r\}, p \in Q, \delta(q, \alpha) = (p, \beta, L)$ :



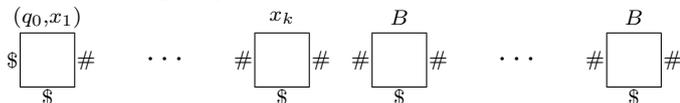
for  $\alpha \in \{0, 1, B\}$ :



and for  $\alpha \in \{0, 1, B\}$ :



The desired reduction is defined by  $f(x) = \langle T(M_A), S, 0^{p(|x|)} \rangle$ , where if  $x$  is written as  $x_1x_2 \dots x_k$  for  $x_i \in \{0, 1\}$ ,  $S$  is the following sequence of tiles with  $p(|x|) - k$  blanks:



From the construction, it is easy to see that  $f$  is polynomial-time computable. If  $S$  extends to a set of legal tiles which tile a  $p(|x|) \times p(|x|)$  square, then  $S$  will have to occupy the bottom row of the square. In this case, the symbols at the top of the bottom row represents the initial configuration of  $M_A$  on input  $x$ . Likewise, the symbols at the top of the  $i$ th row from the bottom will represent the configuration of a position reachable by  $M_A$  within  $i$  steps. Since  $M_A$  will reach the accepting or rejecting state in less than  $p(|x|)$  steps, and the tiling can only duplicate the accepting state,  $S$  can extend to a tiling only if  $M_A$  accepts  $x$ . Similarly, if  $M_A$  accepts  $x$ , then  $S$  can be extended to a tiling of a  $p(|x|) \times p(|x|)$  square. This completes the proof.  $\blacksquare$

### 3 Boolean Satisfiability

We consider Boolean formulas in conjunctive normal form. Namely, each formula  $F$  of  $n$  variables is given as a set of clauses  $\{C_1, C_2, \dots, C_m\}$ , where each  $C_i$  is a set of literals  $\{l_{i,1}, \dots, l_{i,j_i}\}$ . A literal is a variable or negation of a variable. Then  $F = \prod_{i=1}^m \sum_{k=1}^{j_i} l_{i,j_k}$ , where  $\prod$  denotes conjunction, and  $\sum$  denotes disjunction.

### BOOLEAN SATISFIABILITY PROBLEM (SAT)

*Instance.* A boolean formula  $\psi$  in conjunctive normal form.

*Question.* Is  $\psi$  satisfiable?

For convenience, denote by SAT the positive instances of Boolean satisfiability problem; namely,

$$SAT = \{\psi \mid \psi \text{ is satisfiable}\}.$$

**Theorem 6 (Cook's Theorem)** *SAT is NP-complete.*

**Proof.** We will polynomially reduce bounded tiling to SAT. Then by Proposition 2(2), we know that every problem in NP is polynomially reducible to SAT.

Let  $\langle T, S, 0^n \rangle$  be an instance of bounded tiling. Label the different tiles occurred in  $S$  as  $t_1, \dots, t_\ell$ , and the tiles in  $T$  but not in  $S$  as  $t_{\ell+1}, \dots, t_{|T|}$ . Let  $H_T$  denote the set of tile pairs  $[t_i, t_j]$  such that  $t_i$  can be placed horizontally left to  $t_j$ , and  $V_T$  the set of tile pairs  $[t_i, t_j]$  such that  $t_i$  can be placed vertically below  $t_j$ . Create  $N = n^2|T|$  variables  $v_{i,j,k}$ , where  $1 \leq i, j \leq n$ , and  $1 \leq k \leq |T|$ . We will later want  $v_{i,j,k}$  to be 1 if the  $(i, j)$ th cell of the square is covered by the  $k$ th tile. We construct a formula  $\psi = \psi_1\psi_2\psi_3$  as follows.

1. Let  $S = s_1 \cdots s_m$ . Assume that  $s_j = t_{k_j}$  for  $j = 1, \dots, m$ , where  $k_j \in \{1, \dots, \ell\}$ . Let

$$\psi_1 = \prod_{j=1}^m v_{1,j,k_j}. \quad (1)$$

Then  $\psi_1 = 1$  if and only if the  $j$ th cell in the bottom row is covered by  $t_{k_j}$  for  $1 \leq j \leq m$ .

2. For all  $k, k'$  with  $[t_k, t_{k'}] \notin H_T$ , where  $1 \leq k, k' \leq |T|$ , let

$$\psi_{2,1} = \prod_{i=1}^n \prod_{j=1}^{n-1} \prod_{[t_k, t_{k'}] \notin H_T} (\neg v_{i,j,k} + \neg v_{i,j+1,k'}). \quad (2)$$

For all  $k, k'$  with  $[t_k, t_{k'}] \notin V_T$ , where  $1 \leq k, k' \leq |T|$ , let

$$\psi_{2,2} = \prod_{i=1}^{n-1} \prod_{j=1}^n \prod_{[t_k, t_{k'}] \notin V_T} (\neg v_{i,j,k} + \neg v_{i+1,j,k'}). \quad (3)$$

Let  $\psi_2 = \psi_{2,1}\psi_{2,2}$ . Then  $\psi_2 = 1$  if and only if the adjacency conditions of tiling are satisfied.

3. Finally, let  $\psi_3 = \psi_{3,1}\psi_{3,2}$ , where

$$\psi_{3,1} = \prod_{i=1}^n \prod_{j=1}^n \sum_{k=1}^{|T|} v_{i,j,k} \quad (4)$$

and

$$\psi_{3,2} = \prod_{i=1}^n \prod_{j=1}^n \prod_{k \neq k'} (\neg v_{i,j,k} + \neg v_{i,j,k'}) \quad (5)$$

Then  $\psi_{3,1} = 1$  if and only if each cell is covered by at least one tile, and  $\psi_{3,2} = 1$  if and only if each cell is covered by at most one tile. Thus,  $\psi_3 = 1$  if and only if each cell is covered by exactly one tile.

We note that  $|\psi| = \Theta(n^2|T|^2)$ . Let  $f(\langle T, S, 0^n \rangle) = \psi$ , then it is easy to see that  $f$  is polynomial-time computable. If  $\langle T, S, 0^n \rangle$  is a positive instance of bounded tiling, then for all  $i$  and  $j$  with  $1 \leq i \leq n$  and  $1 \leq j \leq n$ , there must be a  $1 \leq k \leq |T|$  such that the square at location  $(i, j)$  is tiled by  $t_k$ . Set  $v_{i,j,k} = 1$ . We conclude that every clause in (4) and (5) is satisfied. Clearly, every unit clause in (1) is satisfied. Now for each pair  $(k, k')$  with  $(t_k, t_{k'}) \notin H_T$ , and for all  $(i, j)$  with  $1 \leq i, j \leq n$ , at least one of  $v_{i,j,k}$  and  $v_{i,j,k'}$  has not been set to 1. Set that variable to 0. Thus, every clause in (2) is satisfied. Similarly, every clause in (3) is also satisfied. Conversely, we can show that if  $\psi$  is satisfiable then  $\langle T, S, 0^n \rangle$  is a positive instance of bounded tiling. This completes the proof. ■

## 4 3-Satisfiability

We consider CNF Boolean formula  $F$  of  $n$  variables  $\{C_1, C_2, \dots, C_m\}$  in which each clause consists of exactly three literals. We call such a formula a 3CNF formula.

**3-SATISFIABILITY PROBLEM (3SAT)**

*Instance.* A 3CNF formula  $\psi$ .

*Question.* Is  $\psi$  satisfiable?

Denote by 3SAT the positive instances of 3-Satisfiability problem; namely,

$$3SAT = \{\psi \mid \psi \text{ is a 3CNF and is satisfiable}\}.$$

**Theorem 7** *3SAT is NP-complete.*

**Proof.** It suffices to reduce a CNF formula to a 3CNF formula. Let  $F$  be a CNF formula. We introduce three new variables  $y_1, y_2, y_3$  and seven new clauses:

$$\{y_1, y_2, y_3\}, \{y_1, y_2, \neg y_3\}, \{y_1, \neg y_2, y_3\}, \{\neg y_1, y_2, y_3\}, \{y_1, \neg y_2, \neg y_3\}, \{\neg y_1, \neg y_2, y_3\}, \{\neg y_1, y_2, \neg y_3\}.$$

Namely, these seven new clauses are all possible 3-literal (nontrivial) clauses over variables  $y_1, y_2, y_3$  except  $\{\neg y_1, \neg y_2, \neg y_3\}$ . It is easy to see that the only truth-value assignment to satisfy all these seven clauses is  $y_1 = y_2 = y_3 = 1$ .

Let  $C$  be a clause of  $F$ . Then  $C = \{z_1, z_2, \dots, z_k\}$  for some  $k \geq 1$ , where  $z_i$  is a literal. We will replace  $C$  by 3-clause(s) as follows.

*Case 1:*  $k = 1$ . Replace  $C$  by  $\{z_1, \neg y_1, \neg y_2\}$ .

*Case 2:*  $k = 2$ . Replace  $C$  by  $\{z_1, z_2, \neg y_1\}$ .

*Case 3:*  $k = 3$ . Do nothing.

*Case 4:*  $k = 4$ . Introduce one new variable  $u_1 = y_C$ , and replace  $C$  by the following 3-clauses:

$$\{z_1, z_2, u_1\}, \{z_3, z_4, \neg u_1\}, \{\neg z_3, u_1, \neg y_1\}, \{\neg z_4, u_1, \neg y_1\}.$$

Note that the last three 3-clauses imply that  $z_3 + z_4$  is equivalent to  $u_1$ , and so  $z_1 + z_2 + u_1$  is equivalent to  $z_1 + z_2 + z_3 + z_4$ .

*Case 5:*  $k > 4$ . Inductively apply Case 4 until only clauses of 3 literals left. For example, when  $k = 5$ , first replace  $C$  with the following clauses:

$$\{z_1, z_2, z_3, u_1\}, \{z_4, z_5, \neg u_1\}, \{\neg z_4, u_1, \neg y_1\}, \{\neg z_5, u_1, \neg y_1\}.$$

Then the last three clauses of three literals imply that  $z_4 + z_5$  is equivalent to  $u_1$ . Thus,  $z_1 + z_2 + z_3 + z_4 + z_5$  is equivalent to  $z_1 + z_2 + z_3 + u_1$ . Now replace the first clause of four literals  $\{z_1, z_2, z_3, u_1\}$  by the following 3-clauses, where  $u_2$  is a new variable:

$$\{z_1, z_2, u_2\}, \{z_3, u_1, \neg u_2\}, \{\neg z_3, u_2, \neg y_1\}, \{\neg u_1, u_2, \neg y_1\}.$$

As shown before that the last three clauses imply that  $u_2$  is equivalent to  $z_3 + u_1$ , and so  $z_1 + z_2 + z_3 + u_1$  is equivalent to  $z_1 + z_2 + u_2$ .

Let  $G$  be the product of all new clauses. It is straightforward to see that  $F$  is satisfiable if and only if  $G$  is satisfiable, and the above transformation from  $C$  to a set of 3-clauses can be carried out in time polynomial in  $k$ . Thus,  $SAT \leq_m^p 3SAT$ . This completes the proof. ■