

- 教材讨论

- TC第34章第2节、第3节前3页、第5.1节

# 问题1：判定问题和优化问题

- Although P, NP, and NP-complete problems are confined to the realm of decision problems, we can take advantage of a convenient relationship between optimization problems and decision problems.
  - 你怎么理解这句话？
  - 以下这些优化问题对应的判定问题分别是什么？
    - traveling salesperson problem
    - set cover problem
    - knapsack problem
  - 优化问题和对应的判定问题哪个更难？

# 问题1：判定问题和优化问题 (续)

- 优化问题有自己的“NP”和“P”，你理解了吗？

**Definition 2.3.3.21.** **NPO** is the class of optimization problems, where  $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, \text{cost}, \text{goal}) \in \text{NPO}$  if the following conditions hold:

- (i)  $L_I \in \text{P}$ ,
- (ii) there exists a polynomial  $p_U$  such that
  - a) for every  $x \in L_I$ , and every  $y \in \mathcal{M}(x)$ ,  $|y| \leq p_U(|x|)$ , and
  - b) there exists a polynomial-time algorithm that, for every  $y \in \Sigma_O^*$  and every  $x \in L_I$  such that  $|y| \leq p_U(|x|)$ , decides whether  $y \in \mathcal{M}(x)$ , and
- (iii) the function  $\text{cost}$  is computable in polynomial time.

Informally, we see that an optimization problem  $U$  is in NPO if

- (i) one can efficiently verify whether a string is an instance of  $U$ ,
- (ii) the size of the solutions is polynomial in the size of the problem instances and one can verify in polynomial time whether a string  $y$  is a solution to any given input instance  $x$ , and
- (iii) the cost of any solution can be efficiently determined.

**Definition 2.3.3.23.** **PO** is the class of optimization problems  $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, \text{cost}, \text{goal})$  such that

- (i)  $U \in \text{NPO}$ , and
- (ii) there is a polynomial-time algorithm that, for every  $x \in L_I$ , computes an optimal solution for  $x$ .

# 问题2: P

- 关于P, 你如何理解这几句话:
  - Very few practical problems require time on the order of a high-degree polynomial.
  - For many reasonable models of computation, a problem that can be solved in polynomial time in one model can be solved in polynomial time in another.
    - 特别地, 对于parallel computer, 为什么要强调the number of processors grows **polynomially** with the input size
  - The class of polynomial-time solvable problems has nice closure properties.

## 问题2: P (续)

- 为什么关于复杂性的讨论会涉及到编码？你能举个例子吗？
- 如何将编码从复杂性的讨论中隔离出去？
  - Rule out “expensive” encodings.
  - If two encodings  $e_1$  and  $e_2$  of an abstract problem are **polynomially related**, whether the problem is polynomial-time solvable or not is independent of which encoding we use.

# 问题2: P (续)

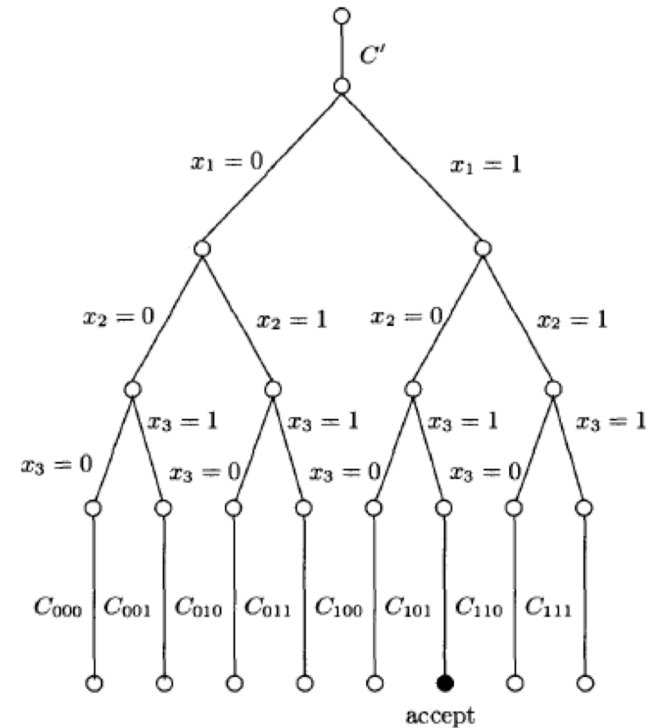
- decide in polynomial time和accept in polynomial time的区别是什么?
  - To accept a language, an algorithm need only produce an answer **when provided a string in L**, but to decide a language, it must correctly accept or reject **every string in  $\{0, 1\}^*$** .
- 联系又是什么? 为什么会有这种联系?
  - $P = \{L \subseteq \{0, 1\}^* : \text{there exists an algorithm } A \text{ that decides } L \text{ in polynomial time}\} .$
  - $P = \{L : L \text{ is accepted by a polynomial-time algorithm}\} .$
- 在模拟接受算法的判定算法中, 如果接受算法不停机, 判定算法可以在什么时候停下来?

# 问题3： NP

- 你怎么理解certificate?
- 这些NP问题中的certificate是什么？ 如何来验证？
  - longest simple path
  - Hamiltonian cycle
  - 3-CNF satisfiability
  - circuit satisfiability
  - formula satisfiability
  - clique problem
  - vertex cover problem
  - traveling salesman problem
  - subset sum problem
  - primality testing
    - [http://en.wikipedia.org/wiki/Primality\\_certificate](http://en.wikipedia.org/wiki/Primality_certificate)

# 问题3: NP (续)

- 你能结合这个例子，从以下两个方面来解释P和NP吗？
  - 可解 vs. 可验证
  - 确定性 vs. 非确定性
- 因此，为什么 $P \subseteq NP$ ？





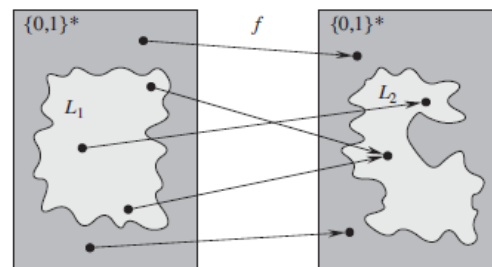
# 问题4: NP-hard与NPC

- 你怎么理解 $L_1$  is polynomial-time reducible to  $L_2$ ?

if there exists a polynomial-time computable function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  such that for all  $x \in \{0, 1\}^*$ ,

$x \in L_1$  if and only if  $f(x) \in L_2$ .

- 作为一种二元关系,  $\leq_P$  具有什么性质?
- 为什么这条引理成立?



*Lemma 34.3*

If  $L_1, L_2 \subseteq \{0, 1\}^*$  are languages such that  $L_1 \leq_P L_2$ , then  $L_2 \in P$  implies  $L_1 \in P$ .

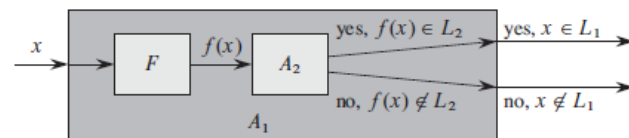
- 如何基于  $\leq_P$  来定义NP-hard?

$L' \leq_P L$  for every  $L' \in NP$

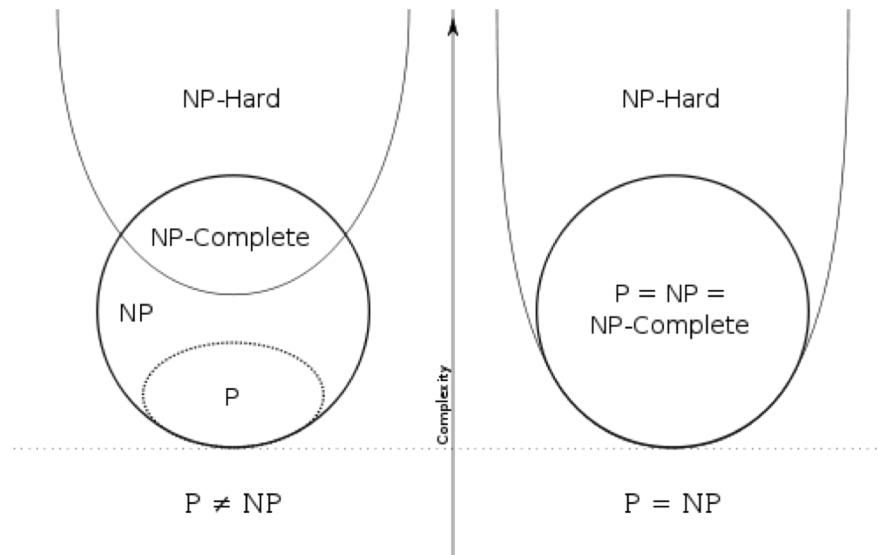
- NPC又是怎么定义的? 为什么这条定理成立?

*Theorem 34.4*

If any NP-complete problem is polynomial-time solvable, then  $P = NP$ .



# 问题4: NP-hard与NPC (续)



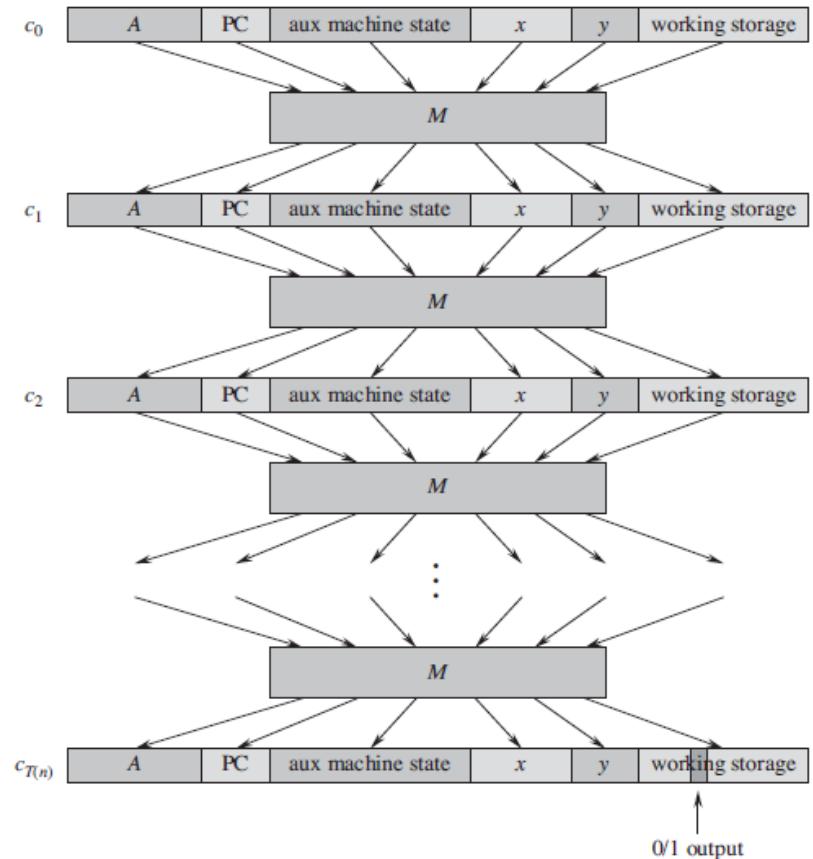
# 问题4: NP-hard与NPC (续)

- 你理解这条引理的证明了吗?

*Lemma 34.6*

The circuit-satisfiability problem is NP-hard.

- 如何建立reduction?
  - 电路的内部结构是什么?
  - 电路的输入输出是什么?
- 如何证明它是polynomial-time computable?

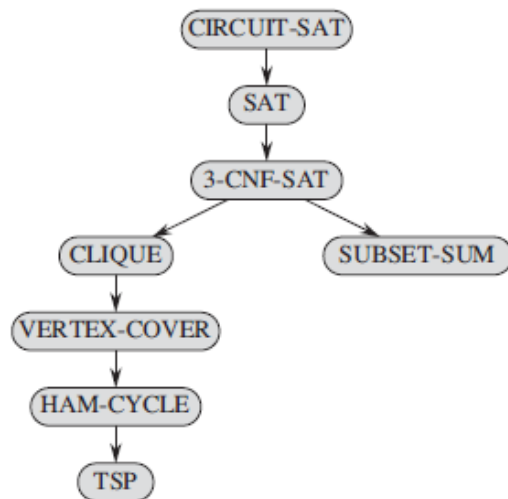


# 问题4: NP-hard与NPC (续)

- 证明NPC的一般方法是什么?

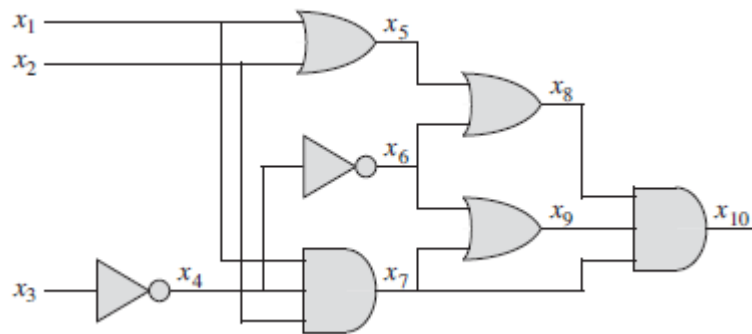
1. Prove  $L \in \text{NP}$ .
2. Select a known NP-complete language  $L'$ .
3. Describe an algorithm that computes a function  $f$  mapping every instance  $x \in \{0, 1\}^*$  of  $L'$  to an instance  $f(x)$  of  $L$ .
4. Prove that the function  $f$  satisfies  $x \in L'$  if and only if  $f(x) \in L$  for all  $x \in \{0, 1\}^*$ .
5. Prove that the algorithm computing  $f$  runs in polynomial time.

- 这张图是什么含义?



# 问题4: NP-hard与NPC (续)

- 如何将circuit satisfiability归约到formula satisfiability, 从而证明后者是NP-hard?

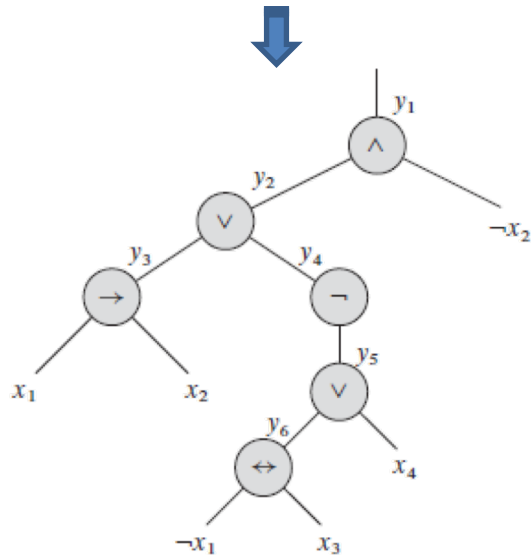


$$\begin{aligned}\phi = & x_{10} \wedge (x_4 \leftrightarrow \neg x_3) \\ & \wedge (x_5 \leftrightarrow (x_1 \vee x_2)) \\ & \wedge (x_6 \leftrightarrow \neg x_4) \\ & \wedge (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4)) \\ & \wedge (x_8 \leftrightarrow (x_5 \vee x_6)) \\ & \wedge (x_9 \leftrightarrow (x_6 \vee x_7)) \\ & \wedge (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9)) .\end{aligned}$$

# 问题4: NP-hard与NPC (续)

- 如何将formula satisfiability归约到3-CNF satisfiability, 从而证明后者是NP-hard?

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2.$$



$$\begin{aligned} \phi' = & y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ & \wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \\ & \wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ & \wedge (y_4 \leftrightarrow \neg y_5) \\ & \wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\ & \wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3)). \end{aligned}$$

$y_1$	$y_2$	$x_2$	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

$$\begin{aligned} \phi'' = & (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \\ & \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2), \end{aligned}$$

If  $C_i$  has 2 distinct literals, that is, if  $C_i = (l_1 \vee l_2)$ , where  $l_1$  and  $l_2$  are literals, then include  $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$  as clauses of  $\phi'''$ .

If  $C_i$  has just 1 distinct literal  $l$ , then include  $(l \vee p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee q) \wedge (l \vee \neg p \vee \neg q)$  as clauses of  $\phi'''$ .

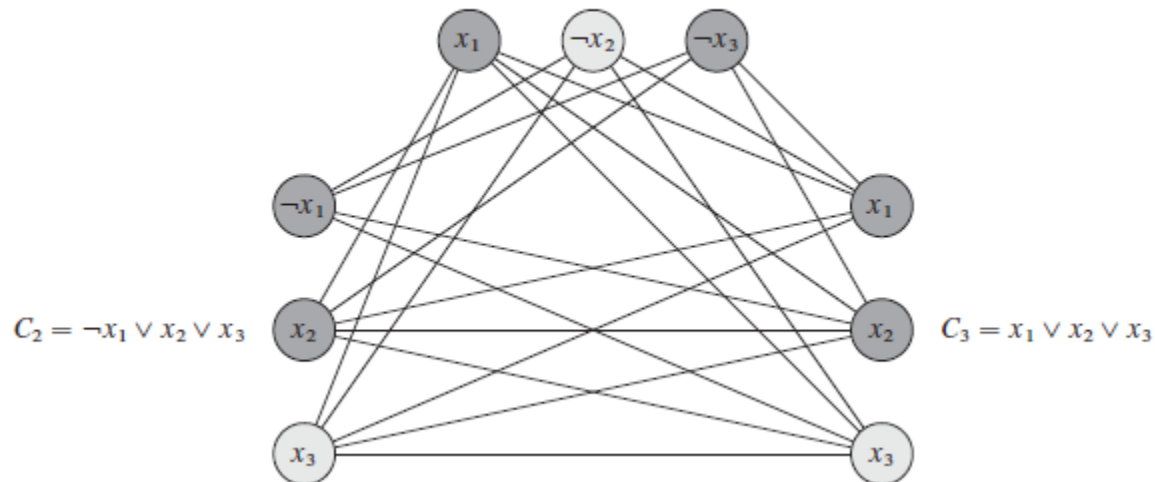
# 问题4: NP-hard与NPC (续)

- 如何将3-CNF satisfiability归约到clique problem, 从而证明后者是NP-hard?

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

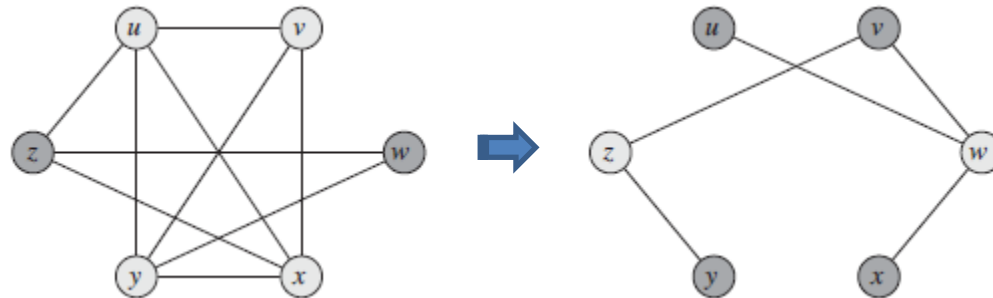


$$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$$



# 问题4: NP-hard与NPC (续)

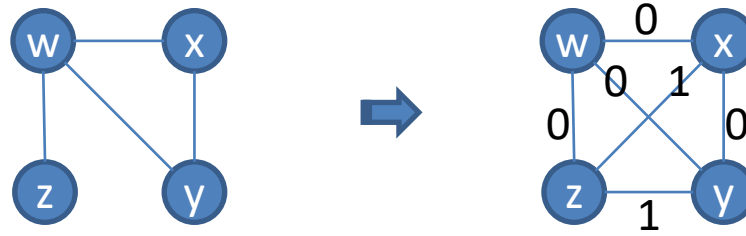
- 如何将clique problem归约到vertex cover problem, 从而证明后者是NP-hard?





# 问题4: NP-hard与NPC (续)

- 如何将hamiltonian cycle归约到traverling salesman problem, 从而证明后者是NP-hard?



# 问题4: NP-hard与NPC (续)

- 如何将3-CNF satisfiability归约到subset sum problem, 从而证明后者是NP-hard?

$\phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$ , where  $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$ ,  $C_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ ,  $C_3 = (\neg x_1 \vee \neg x_2 \vee x_3)$ , and  $C_4 = (x_1 \vee x_2 \vee x_3)$ .



	$x_1$	$x_2$	$x_3$	$C_1$	$C_2$	$C_3$	$C_4$
$v_1 =$	1	0	0	1	0	0	1
$v'_1 =$	1	0	0	0	1	1	0
$v_2 =$	0	1	0	0	0	0	1
$v'_2 =$	0	1	0	1	1	1	0
$v_3 =$	0	0	1	0	0	1	1
$v'_3 =$	0	0	1	1	1	0	0
$s_1 =$	0	0	0	1	0	0	0
$s'_1 =$	0	0	0	2	0	0	0
$s_2 =$	0	0	0	0	1	0	0
$s'_2 =$	0	0	0	0	2	0	0
$s_3 =$	0	0	0	0	0	1	0
$s'_3 =$	0	0	0	0	0	2	0
$s_4 =$	0	0	0	0	0	0	1
$s'_4 =$	0	0	0	0	0	0	2
$t =$	1	1	1	4	4	4	4