第三章 NP完全理论

金人超

华中科技大学计算机科学与技术学院
jrc@hust.edu.cn

§1计算复杂度

- 仅从可计算性理论的角度，似乎可制造出一台能通过图灵测试的机器：

图灵测试的对话时间是有限的，因此任何问题（序列）是一个有限长的字符串。原则上，可以构建一个巨大的查找表，存储对每个可能问题（序列）的人类响应。然后机器只需查询该表来回答问题。

- 这个查表方法的问题是该表必须非常大。所需的存储量将比可观测宇宙的大小（可能有 10^{80} 个原子）大很多倍。因此，创建这样的系统是不现实的。
- 由于该方法在理论上是可行的，因此问题是所需的**资源**量是否合理。
- 我们需要一种衡量算法效率的方法。研究此类问题的领域称为**计算复杂性**理论。
- 与我们之前看到的不同，计算复杂性是一个几乎所有基本问题仍未得到解答的领域——但我们所知道的一些知识也处于数学的前沿。

§1计算复杂度

时间复杂度、空间复杂度，统称为**时空开销**。

设f为图灵机M所计算的m元函数，M计算$f(x_1,x_2,...,x_m)$的**时间复杂度**定义为

$$T(x_1,\cdots,x_m):(A^*)^m \to N\cup\{\uparrow\}$$

$$T(x_1,\cdots,x_m)=\begin{cases}\text{执行指令次数}, & \text{若} f(x_1,\cdots,x_m)\downarrow \\ \uparrow, & \text{若} f(x_1,\cdots,x_m)\uparrow\end{cases}$$

§1计算复杂度

空间复杂度：设f为图灵机M所计算的m元函数，若$f(x_1,x_2,\ldots,x_m)$有定义，将M对输入$x_1,x_2,\ldots x_m$的计算过程中读写头所“注视”过的最左边的格子和最右边的格子之间的纸带上的部分，称为工作部分。M计算$f(x_1,x_2,\ldots,x_m)$的空间复杂度定义为

$$S(x_1,\cdots,x_m):(A^*)^m \to N \cup \{\uparrow\}$$

$$S(x_1,\cdots,x_m)=\begin{cases}\text{工作部分的格子数，若} f(x_1,\cdots,x_m)\downarrow \\ \uparrow, \qquad\qquad \text{若} f(x_1,\cdots,x_m)\uparrow\end{cases}$$

注意：格子可重复利用。

§1 计算复杂度

- **问题：**

图灵机模型下的时空开销是否能反映现实计算机上的时空开销？

真实的时间开销以时、分、秒为单位，而图灵机的时间开销以计算步数为单位，这样合适吗？

时间复杂度的大小和空间复杂度的大小之间有什么联系？

§1 计算复杂度

考虑满足如下条件的数论函数*f*:

$$\lim_{n\to+\infty} f(n) = +\infty \qquad \textbf{(1.1)}$$

我们想用这样的函数作为计算复杂度的上界。*n*为问题的描述规模的大小 。可以明确地定义为

$$\boldsymbol{n=|x_1|+ |x_2|+ \ldots+|x_m|}$$

注：| · | 表示字符串长度.

§1 计算复杂度

定义： $f(n)=O(g(n))$，是指存在常数c使 $f(n)\leq c\cdot g(n)$ 几乎处处成立（a.e.，即存在M，使得对任意$n>M$，$f(n)\leq c\cdot g(n)$ 成立）。此时称f的增长速度不超过g。

注：若对任意$n\in N$，$g(n)>0$，则上述定义中的“几乎处处”可以去掉，定义的含义不变。

若$f(n)=O(g(n))$且$g(n)=O(f(n))$,则称f与g有相同的增长速度。

若$f(n)=O(g(n))$且$g(n)\neq O(f(n))$,则称f比g增长慢。

若$f(n) \neq O(g(n))$且$g(n)=O(f(n))$,则称f比g增长快。

§1 计算复杂度

例 $1320n^2+710n+8527=O(n^2)$

因为当n充分大后

$1320n^2+710n+8527\le 1321n^2$皆成立，

于是 $1320n^2+710n+8527\le 1321n^2$ a.e.

另一方面，易知$n^2=O(1320n^2+710n+8527)$

所以称$1320n^2+710n+8527$与n^2增长一样快。

例 n^3比$1320n^2+710n+8527$增长快。

§1 计算复杂度

通常用O(log$n$), O(n), O(n^2), O(n^3), O(2^n)，… 等来描述算法的计算复杂度的上界。

问题：为什么用O(·) 的形式，而不直接用一个函数比如1320n^2+710n+8527来描述算法的计算复杂度的上界？

§1 计算复杂度

定理1 **设*f*,*g*为*N*→*N*的全函数。**

$$\lim_{n\to+\infty} f(n)=+\infty\,,\qquad \lim_{n\to+\infty} g(n)=+\infty$$

且 $\lim\limits_{n\to+\infty}\dfrac{f(n)}{g(n)}=\beta$,则

i. 若β=正实数，则*f*,*g*有相同的增长速率；

ii. 若β=+∞，　则*f*比*g*增长快；

iii. 若β=0，　则*g*比*f*增长快。

§1 计算复杂度

定理2. 设多项式

$P_1(n)=a_r n^r+\ldots+a_1+a_0$,

$P_2(n)=b_s n^s+\ldots+b_1+b_0$,

a_r, $b_s>0$，则

i 若$r>s$, 则P_1比P_2增长快；

若$s>r$, 则P_2比P_1增长快；

ii 若$r=s$, 则P_1与P_2增长速度一样。

定理3 函数k^n（$k>1$）比任何多项式增长快。

§2 P和NP

定义1 给定字母表A，A上语言 L被称为**多项式时间可判定的**，是指：

i） 存在一个Turing机 T，它接受语言 L；

ii） 伴随着 T，存在一个多项式$p(n)$，使得对任一$x\in L$都有：

T接收x所经历的机器运行步数$\le p(|x|)$ 。

定义2 所有多项式时间可判定的语言所构成的类（集合）称为**P**类。

§2 P和NP

定义3 设A是一个字母表，f是A^*到A^*的一个全函数。如果存在计算f的图灵机M和多项式$p(n)$，使得当输入为x时M的计算步数$\leq p(|x|)$,则称函数f是多项式时间可计算的。

通常认为具有多项式时间算法的问题为易解的问题，否则就是难解的。

问题：

为什么定义中只提到全函数？

多项式时间可判定的与多项式时间可计算的有什么区别和联系？

试证明：语言$L\in P$当且仅当L的特征函数是多项式时间可计算函数。

§2 P和NP

Cook - karp论题 任给一字母表A,对A^*上的任一全函数$f:A^*\to A^*$，若世间有某种（不管何种）装置能在多项式时间内计算函数f，则必存在一台Turing机，它能在多项式时间内计算函数f。

问题：为什么普遍接受Cook-karp论题？为什么将“易解”和“难解”的界限划在多项式时间和非多项式时间之间？

根据Cook-karp论题，要证明某语言$L\in P$，只需证明存在多项式时间算法，对任意字符串x，该算法能判定x是否属于L。这种算法不必是图灵机算法，而可以是现实计算机上的算法。

例：最长递增子序列问题

- 设 $X_1,X_2,\dots,X_n$ 是一个整数列表。 任务是找到最长的递增子序列，即 $X_{i1} < X_{i2} < \dots < X_{ik}$ 其中 $1 \le i1 < i2 < \dots < ik \le n$。 例如：

  3,8,2,4,9,1,5,7,6

有若干长度为 4 的最长子序列：(2,4,5,7), (2,4,5,6), (3,4,5,6), (3,4,5,7)

- 可以尝试所有可能性，但这种方法效率不高。 尝试所有大小为 k 的可能性需要 $\binom{n}{k}$ 时间，并且 $\sum \binom{n}{i} = 2^n$. 这是一个指数算法。
- 动态规划方法：

  $O(n^2)$ 时间。

例：稳定婚姻问题

- 考虑到 n个男人和 n个女人，每个男人对每个女人进行排名，反之亦然。
- 我们希望避免不稳定，即存在一对男女都更喜欢对方而不是他们的实际配偶。
- 没有任何这种不稳定性的匹配被称为稳定的婚姻。
- 解决这个问题最简单的方法是遍历所有 n! 可能的匹配并在找到时输出一个稳定的匹配。
- 存在一个高效的算法来寻找稳定的婚姻，同时解决隐含的问题：稳定的婚姻是否一定存在？
- 注意输入由 2n 个列表组成，每个列表有 n 个数字。高效算法需要 $O(n^2)$ 时间，因此相对于输入大小是线性时间。

§2 P和NP

在第一章图灵机的定义中，将“协调性条件”取消，当某个格局存在多于一个图灵机指令符合当前格局时，从中任选一条执行。如此定义的图灵机称为非确定型的图灵机。此后将满足“协调性条件”的图灵机改称为确定型图灵机。

§2 P和NP

定义 给定字母表A，若Turing机M对某个输入$x\in A^*$，存在至少一条接受路径（或称存在一个M接受x的计算），则称非确定Turing机M接受字x。

定义 给定字母表A，非确定Turing机M所接受的语言定义为

$$L=\{x\in A^* \mid M\text{接受}x\}$$

定义 设A是给定的字母表，$L\subseteq A^*$，如果存在接受语言 L 的非确定型Turing机 M, 并伴随一个多项式$p(n)$，使得对于每一个$x\in L$，存在 M对x的接受路径$c_1,c_2,\ldots c_m$, 其中m为计算的步数，满足$m\le p(|x|)$，则称语言 L 属于NP类。

§2 P和NP

Cook - karp论题的非确定计算形式

任给一字母表 A，语言$L \subseteq A^*$，若存在着某种（不管何种）非确定型计算的装置能在多项式时间内解决问题$x \in ?\ L$，则必有$L \in$NP，即必定存在一台非确定型的Turing机M，它能在多项式时间内解决问题$x \in ?\ L$ 。

注：非确定的计算装置目前世界上并不存在。非确定的图灵机只是一种理论模型，没有什么实用价值。提出这一模型是为了刻画NP这一类问题，而不是想用这种机器去解决NP中的问题。

§2 P和NP

例 $A=\{0,1\}$, $C=\{x\in A^*|x$是一个合数的二进制表示$\}$

例如 $001001\in C$， $0101\notin C$

可构造如下一个接受语言C的非确定型算法：

输入：x $//x\in A^*$

1. 设x表示的二进制数为n，任意挑选一个正整数m，$1<m<n$;
2. 若m整除n，则停机；否则执行3.
3. 死循环（比如*while*$(x==x)\{;;;\}$）

$x\in C$当且仅当此算法运行时存在一个最终停机的计算过程。在每个最终停机的计算过程中，只需挑选一个不超过$|x|$位的二进制数，然后做一次被除数为$|x|$位二进制数的除法，判断余数是否为0即可，计算时间不超过一个关于$|x|$的多项式。（注：这里$|\cdot|$表示字符串长度）

据此，$C\in$NP

注意：如果将“任意挑选一个”改为“遍历”，可以得到一个指数（为什么？）计算时间的确定型算法。

注：此问题确实存在多项式时间算法，即$C\in$P

。

§2 P和NP

问题：

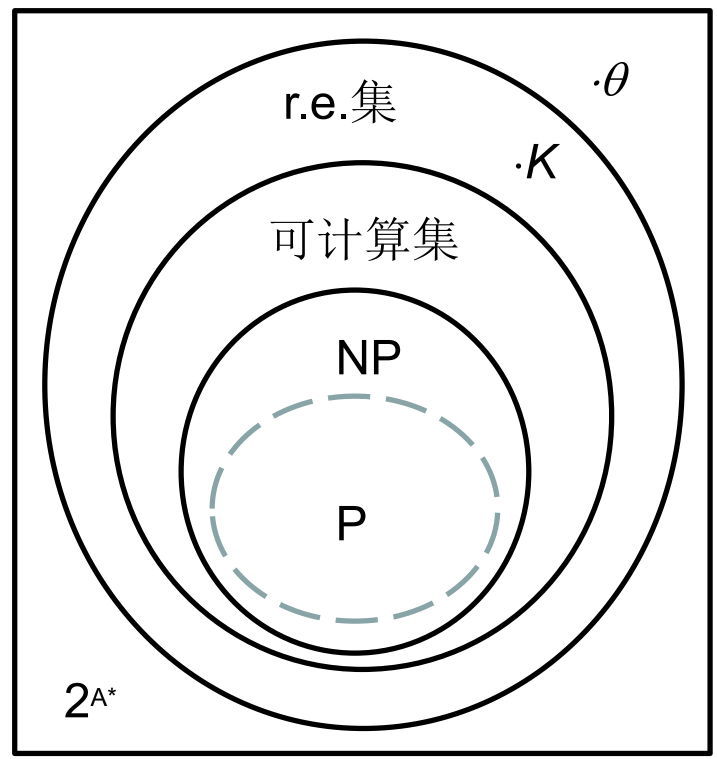
1. 现实中的随机算法、概率型的算法或机器是不是非确定的图灵机？区别和联系在哪里？

2. 每个语言对应一个判定问题。直观上如何理解NP类？什么样的问题是NP中的问题？

§2 P和NP

试证明：

1. 若L是某个非确定型图灵机所接受的语言，则L也是r.e.集。



2. 若$L\in$NP，则 L是可计算集。

3. P$\subseteq$NP

世界难题：P$\subset$NP or P=NP？

§2 P和NP

千禧年大奖难题

（**Millennium Prize Problems**），

是由美国克雷数学研究所（Clay Mathematics Institute,CMI） 于2000年5月24日公布的七个数学难题。根据克雷数学研究所订定的规则，所有难题的解答必须发表在数学期刊上，并经过各方验证，只要通过两年验证期，每解破一题的解答者，会颁发奖金100万美元。

§2 P和NP

大奖题目 (http://www.claymath.org/millennium-problems)

- 杨-米尔斯理论（*Yang-Mills Existence and Mass Gap*）
- 黎曼假设（*The Riemann Hypothesis*）
- P对NP问题（*P versus NP*）
- 斯托克斯方程（*Navier-Stokes Existence and Smoothness*）
- 霍奇猜想（*The Hodge Conjecture*）
- 庞加莱猜想（*The Poincaré Conjecture*）
- 戴尔猜想（*The Birch and Swinnerton-Dyer Conjecture*）

§2 P和NP

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question.

姚期智(Andrew C.Yao，首位华裔图灵奖得主）关于P vs. NP问题何时能解决的回答：

“It’s hard to say when the question will be resolved. I don’t have even an educated guess. Probably the resolution is that P is not equal to NP. I think the mathematical techniques used will be beautiful.”

§2 P和NP

合取范式可满足性（SAT）问题

命题变元：取值为真或假(1/0)的变元，例如：$P_1,P_2,\ldots,P_n$

文字：命题变元或者命题变元的非，例如：P_1，$\neg P_2$，P_3；

子句：若干文字的析取（∨），例如：$P_1\vee\neg P_2\vee P_3$；

合取范式（CNF）：若干子句的合取（∧），例如：

$(P_1\vee\neg P_2\vee P_3)\wedge(P_2\vee\neg P_3\vee P_4)\wedge(\neg P_1\vee P_3)\wedge (P_1\vee\neg P_2\vee\neg P_4)$

指派：对命题变元的一组赋值。

通常用1表示逻辑值“真”，0表示“假”。

例如：$P_1=0, P_2=1, P_3=0, P_4=1$是以上CNF的一组指派，可记为一个0-1向量(0,1,0,1)

对给定的一组指派，CNF取得一个逻辑值真或假(1/0)。

例如：指派(0,1,0,1)使得上述CNF取值为假。

§2 P和NP

具有n个变元的CNF共有2^n个可能的指派。

若存在某个指派使得一个CNF取值为真，则称该CNF是可满足的。并称此指派满足了该CNF。若所有指派都使得CNF取值为假，则称该CNF是不可满足的，或称之为矛盾式，永假公式。

例如

$$(P_1 \vee \neg P_2 \vee P_3) \wedge (P_2 \vee \neg P_3 \vee P_4) \wedge (\neg P_1 \vee P_3) \wedge (P_1 \vee \neg P_2 \vee \neg P_4)$$

是可满足的，

$$(P_1 \vee P_2) \wedge (\neg P_1 \vee P_2) \wedge (P_1 \vee \neg P_2) \wedge (\neg P_1 \vee \neg P_2)$$

是不可满足的。

合取范式可满足性（SAT）问题：

给任意一个合取范式，判定它是否可满足？

等价的对偶问题：析取范式的永真性问题。

§2 P和NP

任意一个合取范式可编码为某字母表*A*上的一个字符串。

例如 $A=\{P,\wedge,\vee,\neg,(,),0,1,2,\ldots,9\}$

$$(P1\vee\neg P2\vee P3)\wedge(P2\vee\neg P3\vee P4)\wedge(\neg P1\vee P3)\wedge(P1\vee\neg P2\vee\neg P4)$$

定义*A*上的语言$SAT\subseteq A^*$

$SAT=\{x|x$是一个可满足的CNF的编码$\}$

例如：

$$(P1\vee\neg P2\vee P3)\wedge(P2\vee\neg P3\vee P4)\wedge(\neg P1\vee P3)\wedge(P1\vee\neg P2\vee\neg P4)\in SAT$$

$$(P1)\wedge(\neg P1)\notin SAT$$

$$(P1\vee P2)\wedge(\neg P1\vee P2)\wedge(P1\vee\neg P2)\wedge(\neg P1\vee\neg P2)\notin SAT$$

$$4\wedge P)\vee P\neg\notin SAT$$

§2 P和NP

一个**判定问题**可以编码为某个字母表A上语言L的判定问题。所以经常将（判定）问题与语言、集合三个词混用。

定理 $SAT \in \mathrm{NP}$

证 可构造如下接受语言SAT的**多项式时间非确定型**算法：

1. 对给定输入CNF x，“**猜**”一个真值指派；
2. 将上述真值指派代入x，若x取得真值为1，则$x \in SAT$；否则$x \notin SAT$.

§2 P和NP

***SAT*是NP中最有可能不属于P的问题（之一）。这一判断是基于以下概念：**

定义 设L,Q为A上的语言，如果存在从A^*到A^*的多项式时间可计算的全函数f，使得$x\in Q\Leftrightarrow f(x)\in L$，则称**$Q$多项式时间可归约到$L$**,并记作$Q\le_p L$。

例 设字母表$A=\{0,1\}$, L为表示奇数的二进制串的集合；Q为表示偶数的二进制串的集合。试证明$Q\le_p L$（或$L\le_p Q$）

问一个数是否是偶数的问题可以很快地转化为问一个数是否为奇数的问题，有办法解答后者，就有办法解答前者。

§2 P和NP

定理 1　若 $Q \le_p L$，则若 $L \in \mathrm{P}$，则 $Q \in \mathrm{P}$ 。

$Q \le_p L$ 意味着只要 L 易解，则 Q 必易解。即在“是否易解”这个意义上，Q 的计算难度不超过 L 的计算难度，这就是 $\le_p$ 记号的由来。

定理2 若 $R \le_p Q$ ， $Q \le_p L$，则 $R \le_p L$

定理3 若 Q，$L \in \mathrm{P}$ 且 Q，$L \ne \phi$，且 Q，$L \ne A^*$，
则 $Q \le_p L$ 且 $L \le_p Q$

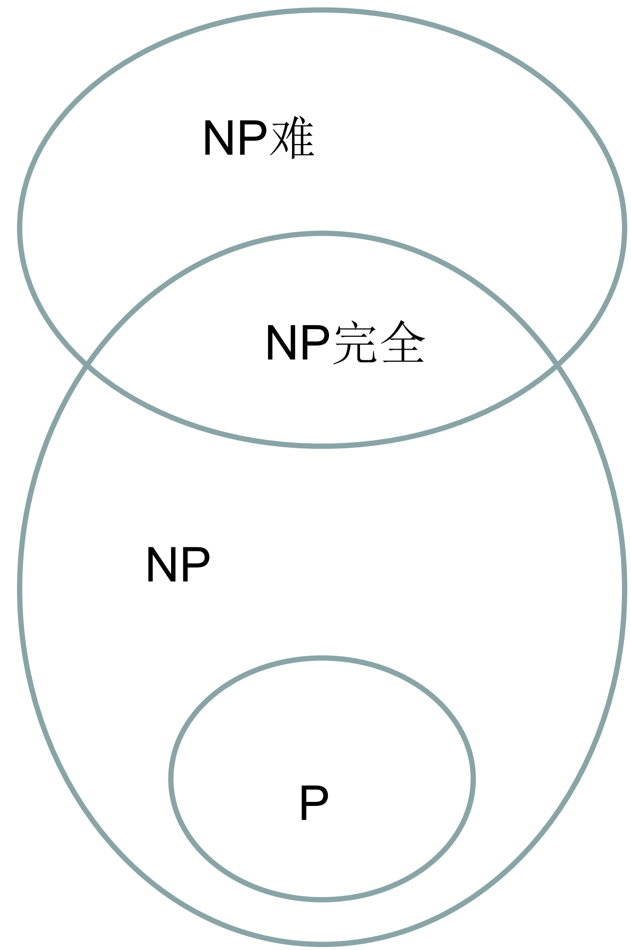
§2 P和NP

定义 语言L被称作为**NP难**的（**NP-hard**）是指对任一语言Q∈NP都有$Q \le_p L$；

若语言L为NP难的，则：若L的成员判定问题易解，则对任何属于NP的语言Q， Q的成员判定问题都易解。 在这个意义上，L具有不不低于NP中所有语言的判定难度。

如果L为NP难的且L∈NP，则称L是**NP完全的**（**NP-complete**）。

语言L为NP完全的，其意义是指L是NP中判定难度最高的语言。



§2 P和NP

定理4 若L，$Q \in$ NP-complete，

则$L \leq_p Q$ 且$Q \leq_p L$

即NP完全的语言的判定难度都相同。

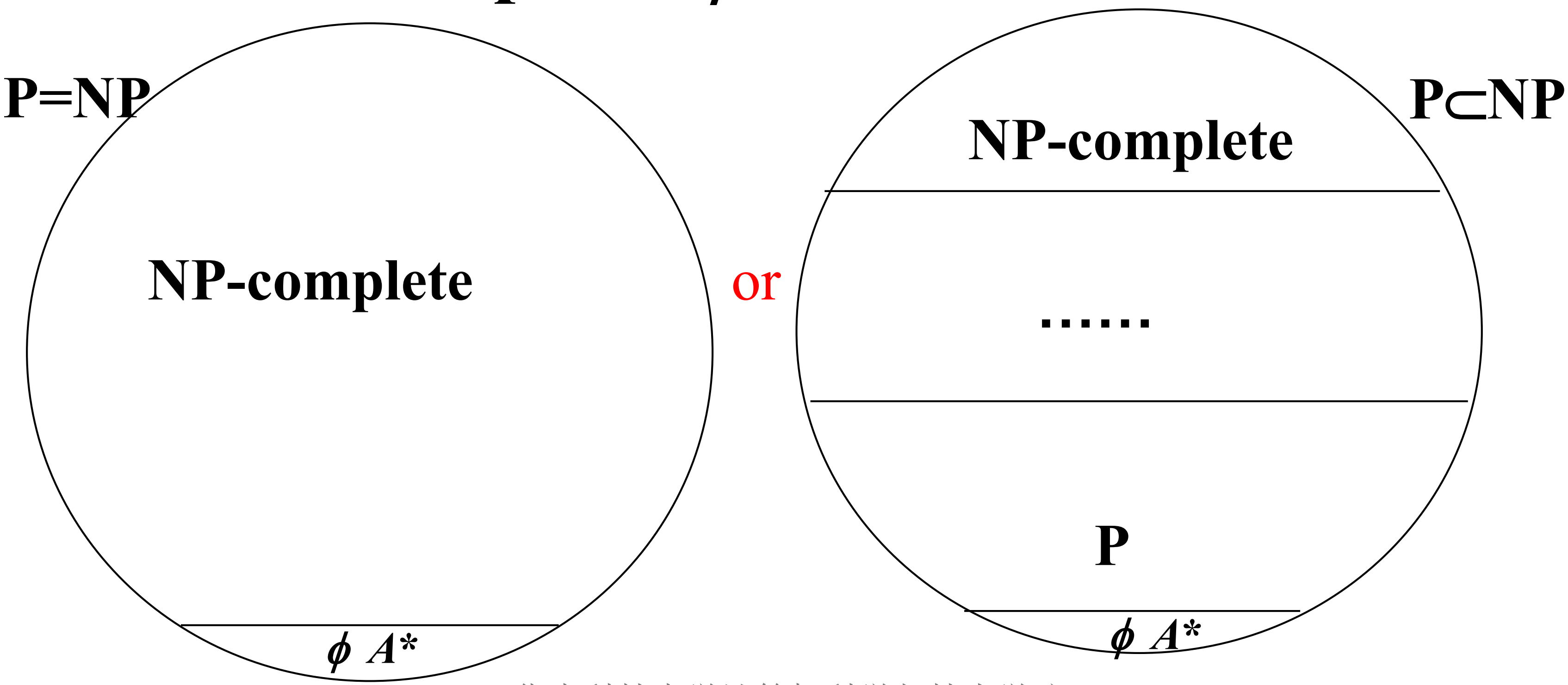
对比定理3， NP-complete与P-$\{\phi, A^*\}$在这一性质上完全一致。

§2 P和NP

试证明：

1）若$L\in$NP-complete，则$L\in$P $\Leftrightarrow$ P=NP

2）P$\cap$NP-complete$\neq\phi\Leftrightarrow$ P=NP



jrc@hust.edu.cn

§2 P和NP

P、NP、NP-hard、NP-complete等概念都是定义为语言的集合。由于语言与判定问题的对应关系，它们经常也被视作判定问题的集合。

SAT问题对应的最优化问题MAX-SAT：

对任意一个CNF，求一组指派，使得被满足的子句最多。

讨论：判定问题通常有一个最优化问题的“版本”，人们经常将判定问题的判定难度作为其对应的最优化问题的求解难度。其中的道理是什么？

§3 Cook定理

定理1(Stephen Arthur Cook) SAT**是NP完全的。**

证 已证$SAT \in$**NP，现只须证明**SAT**为NP难度的。即要证明，对NP中任意语言**L**，**$L \leq_p SAT$**。**

即要证明存在一个多项式时间可计算函数f**，它将**A^***中的任一个字**x**转换成一个合取范式**$f(x)$**，使得**$x \in L$**当且仅当**$f(x) \in SAT$**.**

因$L \in$**NP，则存在不确定型图灵机**M**和多项式**$p(n)$**，使得对任意**$x \in A^*$**，**$x \in L$**当且仅当**M**对输入**x**存在一条最终停机的长度不超过**$p(|x|)$**的计算路径。**

§3 Cook定理

证明思路：

将"存在一条长度不超过*p*(|*x*|)的格局序列，正好构成*M*对输入*x*的一条最终停机的计算路径。"这句话写成：

存在对一些命题变元的一组指派，使得一些约束条件得到满足。

将这些约束条件用合取范式的形式表示成*f*(*x*)，即定义了归约函数*f*。

§3 Cook定理

记$t=p(|x|)$,因为M每算一步，向左右移动至多一格。当步数$\leq t$，M的指针左右移至多只有t格。只须考虑带子上的$2t+1$个格子。

由于只须考虑t步的动作，我们只须记录$t+1$条纸带（每纸带上$2t+1$个格子）。

即只须一张$(t+1)\times(2t+1)$个格子的长方格纸即可展现全部计算过程。

§ 3 Cook定理

设图灵机M的字母表$A=\{s_1,s_2,\ldots,s_r\}$, 内部状态集 $\{q_1,q_2,\ldots,q_m\}$，

对任意$x\in A^*$，计算$t=p(|x|)$后，构造命题变元集：

$\{\rho_{h,j,k},\ \sigma_{i,j,k}\ |h=1,2,\ldots,m;\ i=0,1,2,\ldots,r;$

$j=1,2,\ldots,2t+1;\ k=0,1,2,\ldots,t\}$

$\rho_{h,j,k}$表示第k根带上指针在第j个位置，内部状态为q_h，共$m\times(2t+1)\times(t+1)$个；

$\sigma_{i,j,k}$表示第k根带上第j个位置上符号为s_i，共 $(r+1)\times(2t+1)\times(t+1)$个；

§3 Cook定理

"一条长度不超过*p*(|*x*|)的格局序列，正好构成*M*对输入*x*的一条最终停机的计算路径."这句话可分解成5句话的合取：

1。每条带上指针指在恰一个位置，恰有一个内部状态；

2。每条带上每一格恰有一个符号；

3。第0条带上记录了输入为*x*时的初始格局；

4。第*t*条带上记录了一个停机格局；

5。对任意*k*=0,1,…,*t*-1,第*k*条带到第*k*+1条带的变化符合图灵机*M*的指令；

§3 Cook定理

将"$x_1,x_2,\ldots,x_n$中恰有一个为真"这句话记为 $\nabla\{x_1,x_2,\ldots,x_n\}$，该句子可用如下合取范式写出：

$$\nabla\{x_1,x_2,\ldots,x_n\}=(x_1\vee x_2\vee\ldots\vee x_n)\wedge$$

$$\wedge(\neg x_1\vee\neg x_2)\wedge(\neg x_1\vee\neg x_3)\wedge\ldots\wedge(\neg x_{n-1}\vee\neg x_n)$$

使用此∇记号，则：

1。"每条带上指针指在恰一个位置，恰有一个内部状态"可写成合取范式：

$$\bigwedge_{0\le k\le t}\nabla\left\{\rho_{h,j,k}\ \middle|\ \begin{matrix} h=1,\cdots,m \\ j=1,\cdots,2t+1\end{matrix}\right\} \quad (1)$$

§3 Cook定理

2。"每条带上每一格恰有一个符号。"也可写成一个合取范式：

$$\bigwedge_{0\le k\le t}\ \bigwedge_{1\le j\le 2t+1} \nabla\left\{\sigma_{i,j,k} \mid i=0,1,\cdots,r\right\} \quad (2)$$

§3 Cook定理

设$x = S_{u_1} S_{u_2} \cdots S_{u_{|x|}}$

3。“第0条带上记录了输入为*x*时的初始格局”可写为：

$$\bigwedge_{1 \le j \le t+1} \sigma_{0,j,0} \wedge \bigwedge_{1 \le j \le |x|} \sigma_{u_j, t+1+j, 0} \wedge \bigwedge_{1 \le j \le t-|x|} \sigma_{0, t+1+|x|+j, 0} \wedge \rho_{1,t+1,0} \quad (3)$$

§3 Cook定理

不妨假设当且仅当内部状态为q_m时，图灵机进入停机格局。

不妨假设恰在第t步停机。

思考：为什么可以“不妨假设”？

4。“第t条带上记录了一个停机格局”可写为：

$$\bigvee_{1 \le j \le 2t+1} \rho_{m,j,t} \qquad (4)$$

§3 Cook定理

设图灵机*M*的四重组集分为以下三个子集：

$$\{q_{i_a} s_{j_a} s_{k_a} q_{l_a} \mid a = 1,2,\ldots,\overline{a}\}, \qquad \text{(a)}$$

$$\{q_{i'_b} s_{j'_b} R q_{l'_b} \mid b = 1,2,\ldots,\overline{b}\}, \qquad \text{(b)}$$

$$\{q_{i''_c} s_{j''_c} L q_{l''_c} \mid c = 1,2,\ldots,\overline{c}\}, \qquad \text{(c)}$$

5。“对任意*k*=0,1,…,*t*-1,第*k*条带到第*k*+1条带的变化符合图灵机*M*的指令”可写为：

§ 3 Cook定理

$$\bigwedge_{\substack{1\le j\le 2t+1\\ 0\le k\le t-1}}\left\{\begin{array}{l}
\bigvee_{0\le\gamma\le r}[(\sigma_{\gamma,j,k}\wedge\sigma_{\gamma,j,k+1})\bigwedge_{1\le h\le m}(\neg\rho_{h,j,k})]\vee \\
\text{// 指针不指在第}j\text{格上;} \\
\bigvee_{1\le a\le\bar{a}}(\sigma_{j_a,j,k}\wedge\rho_{i_a,j,k}\wedge\sigma_{k_a,j,k+1}\wedge\rho_{l_a,j,k+1})\vee \\
\text{// 指针指着第}j\text{格,}(a)\text{类指令作用;} \\
\bigvee_{1\le b\le\bar{b}}(\sigma_{j'_b,j,k}\wedge\rho_{i'_b,j,k}\wedge\sigma_{j'_b,j,k+1}\wedge\rho_{l'_b,j+1,k+1})\vee \\
\text{// 指针指着第}j\text{格,}(b)\text{类指令作用;} \\
\bigvee_{1\le c\le\bar{c}}(\sigma_{j''_c,j,k}\wedge\rho_{i''_c,j,k}\wedge\sigma_{j''_c,j,k+1}\wedge\rho_{l''_c,j-1,k+1}) \\
\text{// 指针指着第}j\text{格,}(c)\text{类指令作用.}
\end{array}\right\} \quad (5)$$

§3 Cook定理

对任意$x \in A^*$,

$$f(x)=(1)\wedge(2)\wedge(3)\wedge(4)\wedge(5)$$

是一个合取范式。且可证

$x \in L \Leftrightarrow f(x)$是可满足的合取范式。

即： $x \in L \Leftrightarrow f(x) \in SAT$。

又可证$f(x)$是多项式时间可计算的函数。

故$L \leq_p SAT$。

由L的任意性，知SAT是NP难的，故SAT是NP完全的。 □

§4 另外几个NP完全问题

定理4.1 设Q是NP难度的，$Q \le_p L$，则L是NP难度的。

推论4.2 设Q为NP完全的，$L \in$ NP且$Q \le_p L$，则L为NP完全的。

定义 $k\text{-}SAT := \{x \in A^* | x$表示可满足的CNF，且$x$的每个子句恰为$k$个文字的析取$\}$

通常不拘泥于形式语言和图灵机的术语，直接定义问题：

定义 k-SAT问题：

实例：一个CNF，每个子句恰为k个文字的析取。

问题：该CNF是否可满足？

定理4.3 3-SAT是NP完全的。

练习：试证明2-$SAT \in$ P

§4 另外几个NP完全问题

定义 完全子图问题(CLIQUE):

实例：任给一个图*G*及正整数*K*。

问题：图*G*中是否有*K*阶完全子图？

定理4.4 完全子图问题是NP完全的。

§4 另外几个NP完全问题

定义 集合$S \subseteq V$是图$G=(V,E)$的一个**顶点覆盖**是指：对每个边$\{x,y\} \in E$有$x \in S$或$y \in S$。即牵动了全部边的顶点集合被称作为图的顶点覆盖。

定义 顶点覆盖问题(VC):

实例：任给一图G和一正整数K，

问题：G是否存在大小为K的顶点覆盖？

定理4.5 设$G=(V,E)$为一个图，$G'=(V,E')$为G之补图，对$S \subseteq V$记$V-S=S'$，则有：

S为G中完全子图之顶点集$\Leftrightarrow S'$为G'之顶点覆盖。

推论4.6 顶点覆盖问题是NP完全的。

§4 另外几个NP完全问题

定义 集合覆盖问题(MINIMUM COVER):

实例：任给有穷集类$\{S_1,S_2,\ldots,S_n\}$，和一个正整数$k \le n$。

问题：该集合类是否存在一个k阶子类覆盖？

即是否存在$\{S_{i1},S_{i2},\ldots,S_{ik}\} \subseteq \{S_1,S_2,\ldots,S_n\}$，使得 $S_{i1} \cup S_{i2} \cup \ldots \cup S_{ik} = S_1 \cup S_2 \cup \ldots \cup S_n$

推论4.7 集合覆盖问题是NP完全的。

§4 另外几个NP完全问题

定义 0-1背包问题(0-1 KNAPSACK):

实例：物体的有穷集U，对每个物体$u\in U$，有一个重量$w(u)$和一个收益值$p(u)$，皆为非负整数。给定背包容量W，期望的总收益值P，皆为正整数。

问题：是否存在U的子集V，使得

$$\sum_{u\in V} w(u) \le W \quad 且 \quad \sum_{u\in V} p(u) \ge P \qquad ?$$

如果对每个u，都有$w(u)=p(u)$，且$W=P$，则问题变成问是否存在U的子集V，使得V中物体的总重量恰好等于W。称此问题为子集和问题(SUBSET-SUM)。

定理 子集和问题是NP完全的，从而0-1背包问题也是NP完全的。

提示：3-SAT$\le_p$SUBSET-SUM $\le_p$0-1 KNAPSACK。

§4 另外几个NP完全问题

提示：证明3-SAT$\leq_p$SUBSET-SUM

设CNF x有n个变元，m个子句，$f(x)$由如下实例构成：

对每个变量p_i，定义两件物体p_i和q_i；每件物体的重量是一个$n+m$位整数：第i位为1；若该文字出现在第j个子句中，则第$n+j$个子句为1；其余位为0；

对每个子句c_j，定义两件物体c_j和d_j；每件物体的重量也是$n+m$位整数：第$n+j$位为1，其它位皆为0；

$$W=\underbrace{11\ldots1}_{n位}\underbrace{33\ldots3}_{m位}$$

§4 另外几个NP完全问题

旅行商问题(TSP):

实例：给定有权图$G=(V,E,f)$，正整数L。其中$f:E\rightarrow\{0,1,2\ldots\}$为边的长度。

问题：是否存在一条长度不超过L的周游所有顶点的回路？

哈密顿回路问题(HC):

实例：给定图$G=(V,E)$

问题：图G中是否存在一条哈密顿回路？

显然$HC\le_p TSP$。

可证HC是NP完全的，从而TSP也是NP完全的。

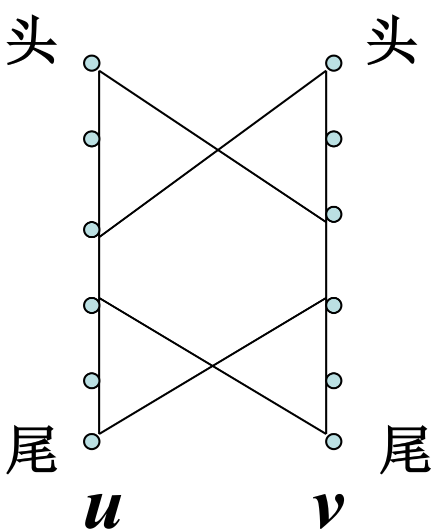
§4 另外几个NP完全问题

定理 ***HC*是NP完全的。**

提示：证明$VC \le_p HC$

对VC的一个实例x，设x中图$G=<V,E>,K\le|V|$. $f(x)$按如下方法算出：

(1)构造K个“选择”结点$a_1,a_2,\ldots,a_K$;



(2)对E中每一边$e=\{u,v\}$，构造一个如图所示的12个点的子图;

(3)对V中每个点的多次出现，按某个任意指定的顺序将它们依次“头尾相连”，而第一个“头”和最后一个“尾”分别与每个a_i，$i=1,2,\ldots,K$相连。

§4 另外几个NP完全问题

现实中的问题：期末到了，全校几百门课要全部考完才能放假。如何安排各门课的考试时间，使得所有考试尽早结束，而且每个同学所选多门课程的考试时间互不冲突？

着色数问题(Chromatic Number/*k*-Coloring)

实例：图$G=(V,E)$，正整数k；

问题：是否可以用不超过k种颜色，给图G的每个顶点着其中一种色，使得任意两个相邻的顶点不同色？

§4 另外几个NP完全问题

定理 颜色数(CN)问题是NP完全的。

证：提示：$3\text{-}SAT \le_p CN$

x： 子句 $c_1, c_2, \ldots, c_m$; 变元 $x_1, x_2, \ldots, x_n$ $(n \ge 4)$

$f(x)$: $G=(V, E)$, $K=n+1$

$$V=\{ x_1, x_2, \ldots, x_n \} \cup \{ -x_1, -x_2, \ldots, -x_n \} \cup \{ y_1, y_2, \ldots, y_n \} \cup \{ c_1, c_2, \ldots, c_m \}$$

$$E=\{ (x_i, -x_i) \mid 1 \le i \le n \} \cup \{ (y_i, y_j) \mid i \ne j \} \cup \{ (y_i, x_j) \mid i \ne j \} \cup \{ (y_i, -x_j) \mid i \ne j \} \cup \{ (x_i, c_j) \mid x_i \notin c_j \} \cup \{ (-x_i, c_j) \mid -x_i \notin c_j \}$$

§4 另外几个NP完全问题

现实中的问题：将上一问题的“所有考试尽早完成”改成“所有考试的平均等待时间最短”

颜色和问题(Chromatic Sum)

实例：图G=(V,E)，正整数k；

问题：是否存在一种使得G的任意相邻顶点不同色的顶点着色方案c:V→N，使得

$$\sum_{v\in V} c(v) \le k \quad ?$$

思考题：最小着色数问题与最小颜色和问题的解总是一致的吗？