第7章 NP完全性

主要内容

Turing机

计算复杂性理论

NP完全性理论的基本概念

用NP完全性理论分析问题

NP难度

Turing机的定义

基本模型　双向无限带的**Turing**机

$M = < Q, \Sigma, \Gamma, \delta, q_0, B, F >$, 其中

Q 有穷状态集

Γ 有穷带字符集

Σ 输入字符集　　$\Sigma \subset \Gamma$

B 空白字符,　　$B \in \Gamma - \Sigma$

q_0 初始状态,　　$q_0 \in Q$

F 终结状态集,　$F \subset Q,\ q_Y, q_N \in F$

$\delta: (Q-F) \times \Gamma \to Q \times \Gamma \times \{L,R\}$　　状态转移函数

瞬时描述-格局(ID)

$\alpha_1 q \alpha_2$ 表示此刻Turing机的FSC处于状态q, 读写头指在串α_2的第一个字符.

例如Turing机 M 的某时刻的状态转移函数是

$$\delta(q, x_i) = (p, Y, L)$$

带上的字符串为 $x_1 x_2 \ldots x_i \ldots x_n$, 读写头指向字符 x_i, 则它的瞬间描述是:

$$x_1 x_2 \ldots x_{i-1} q\, x_i \ldots x_n \vdash x_1 x_2 \ldots x_{i-2} p\, x_{i-1} Y x_{i+1} \ldots x_n$$

$\vdash$ 表示由左边的ID一步达到右边的ID

$\vdash^*$ 表示由左边的ID经有限步达到右边的ID

实例

设 $L=\{ 0^n1^n \mid n \geq 1 \}$, 设计接受 L 的 Turing机如下:

$M = <Q, \Sigma, \Gamma, \delta, q_0, F>$

$Q = \{ q_0, q_1, q_2, q_3, q_Y, q_N \}$,

$\Sigma = \{ 0, 1\}$,

$\Gamma = \{ 0, 1, X, Y, B \}$,

$F = \{ q_Y, q_N\}$

初始将字符串放在从 1 到 n 方格中, FSC处在状态 q_0, 读写头指向方格 1. 将第一个0改写成 X, 然后带头向右扫描. 遇到第一个1, 将1改为Y, 然后带头向左扫描. 遇到第一个 X 改为向右扫描. 这时进入下一个巡回. 每个巡回将一对 0 和 1 改为 X 和 Y, 直到接受或拒斥停机.

北京大学

实例（续）

转移函数如下

	0	1	X	Y	B
q_0	(q_1,X,R)	q_N	q_N	(q_3,Y,R)	q_N
q_1	$(q_1,0,R)$	(q_2,Y,L)	q_N	(q_1,Y,R)	q_N
q_2	$(q_2,0,L)$	q_N	(q_0,X,R)	(q_2,Y,L)	q_N
q_3	q_N	q_N	q_N	(q_3,Y,R)	q_Y

例如输入 0011，Turing 机动作如下：

$$q_00011 \vdash Xq_1011 \vdash X0q_111 \vdash Xq_20Y1 \vdash q_2X0Y1$$

$$\vdash Xq_00Y1 \vdash XXq_1Y1 \vdash XXYq_11 \vdash XXq_2YY \vdash Xq_2XYY$$

$$\vdash XXq_0YY \vdash XXYq_3Y \vdash XXYYq_3 \vdash XXYYq_Y$$

北京大学

Turing机接受语言

被M接受的语言记作$L(M)$，是Σ^*上的字的集合.

当这些字左端对齐方格 1 放在带上, M处于状态 q_0, M 的带头指向方格 1 时, 经过有限步 M 将停机在接受状态 q_Y, 即

$$L(M)=\{\ \omega \mid \omega\in\Sigma^*, \exists\alpha_1,\alpha_2\in\Gamma^*(\ q_0\omega \vdash^*\alpha_1 q_Y\alpha_2)\ \}$$

如果字ω不是$L(M)$中的字, M可以不停机或停机在拒斥状态 q_N.

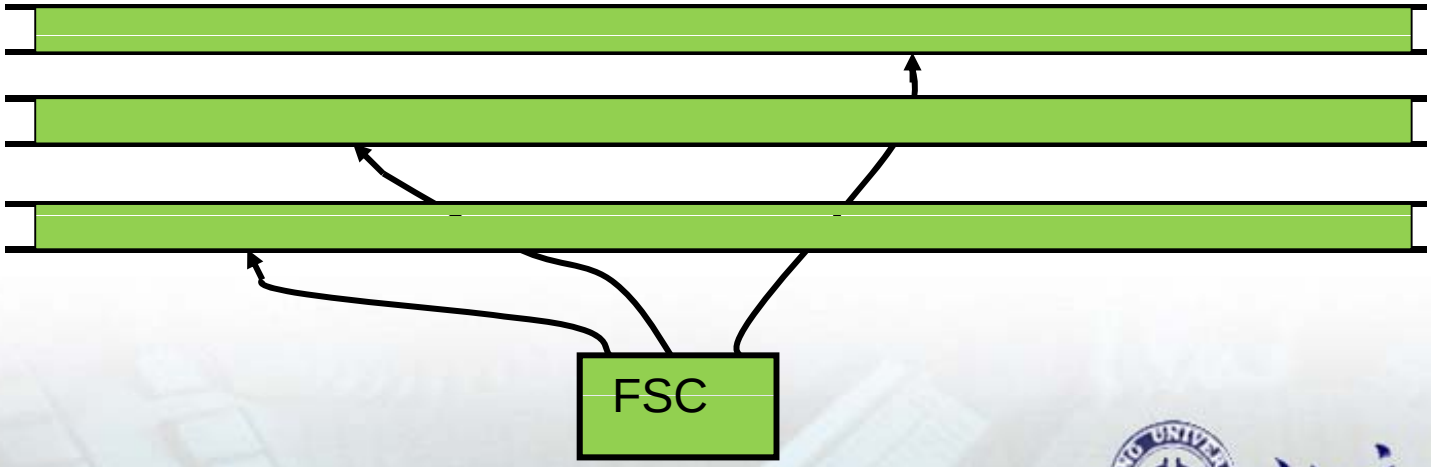
基本 Turing 机可以推广为 k 条带的 Turing 机 DTM.

确定型 Turing机可以推广到非确定型 Turing机 NDTM.

基本Turing机的变种

单向无限带的 Turing 机 带方格从1开始， 向右无限长. 其它与基本 Turing 机相同.

多带的 Turing 机 k 条双向带, k 个读写头, 其中 k 为大于 1 的常数. 初始将输入写在第一条带的方格 1 到 n 内. 其它带为空. 每个读写头扫描一条带,可以改写被扫描方格的字符, 读写头然后向左或向右移动一个方格. 读写头的动作由 FSC 的状态及 k 条带所扫描的 k 个字符来决定.



实 例

例1 $L = \{ wcw^R \mid w$为0-1字符串$\}$, 设计接受 L 的Turing机 M_1 和 M_2, 使得 M_1的时间复杂度为$O(n)$, M_2的空间复杂度为 $O(\log n)$.

M_1有2条带,把 c 左边的 w 复制到第2条带上. 当发现 c 时第2条带的读写头向左, 输入带的读写头向右. 比较两个带头的符号, 如果符号一样, 字符个数一样, M_1 接受 x. M_1 至多作 $n+1$ 个动作. 时间复杂度为 $n+1$. 空间复杂度为$\lceil (n-1)/2 \rceil + 1$.

M_2有2条带, 第2条带作为二进制的计数器. 首先检查输入是否只有1个 c, 以及 c 左边和右边的符号是否一样多. 然后逐个比较 c 左边和右边的字符, 用上述计数器找到对应的字符. 如果所有的字符都一样, M_2接受停机. 空间复杂度为二进制的计数器的占用空间, 即 $O(\log n)$. 时间复杂度为 $O(n^2)$.

计算复杂性理论

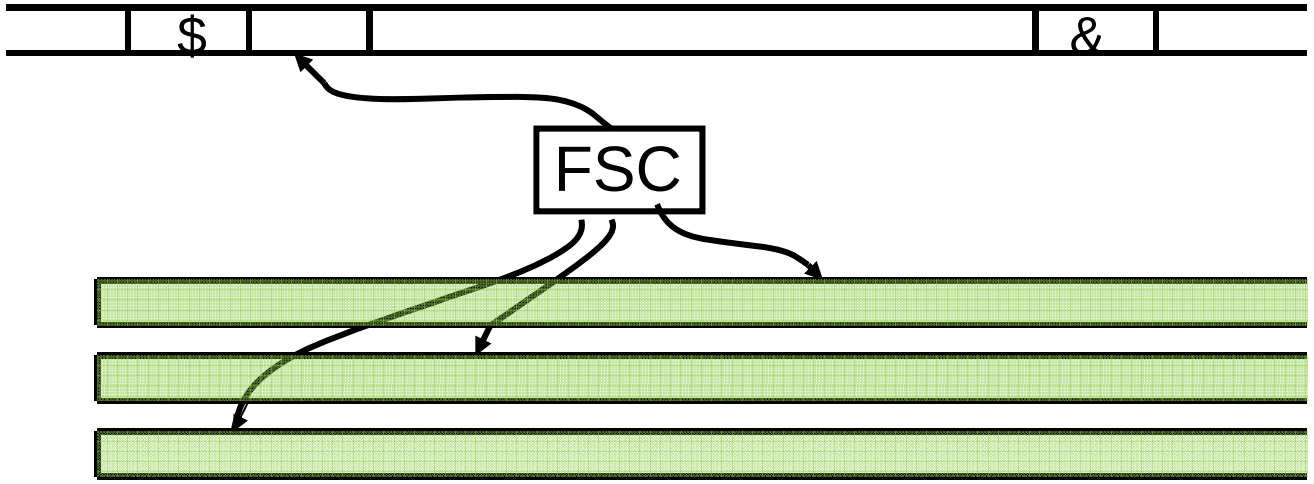
空间和时间复杂度的形式定义

- 确定型Turing机

计算复杂度的有关结果

- 带压缩
- 线性加速
- 带数目的减少

确定型Turing机
空间复杂度的形式定义



离线的**Turing**机 ***M*, 1**条具有端记号的只读输入带, ***k***条半无限存储带. 如果对每个长为***n***的输入串, ***M***在任一条存储带上都至多扫视 ***S*(*n*)**个 单元, 那么称 ***M*** 在最坏情况下的空间复杂度为 ***S*(*n*).**

确定型Turing机时间复杂度的形式定义

k 条双向带的 **Turing**机 ***M***, 一条带包含输入.

如果对于每个长为***n***的输入串, ***M*** 在停机前至多做 ***T*(*n*)** 个动作, 那么称***M***在最坏情况下的时间复杂度为 ***T*(*n*)**.

两条假设:

空间复杂性至少需要 **1**

时间复杂性至少需要读入输入的时间

因此这里作如下规定:

对一切***n***, ***S*(*n*)≥1**, **log*n*** 是 **max { 1,⌈log*n*⌉}** 的缩写.

对一切***n***, ***T*(*n*)≥*n*+1**, ***T*(*n*)**是 **max { *n*+1,*T*(*n*) }** 的缩写.

有关计算复杂度的结果

	带数	M_2 模拟 M_1 的复杂度		类型
M_1	k带	$S(n)$		带压缩
M_2	k带	$cS(n)$, $\forall 0<c<1$		
M_1	k带	$T(n)$ $\liminf_{n\to\infty}\frac{T(n)}{n}=\infty$	cn,$\forall c>1$	时间加速
M_2	k带	$cT(n)$, $\forall 0<c<1$	$(1+\varepsilon)n$	
M_1	k带	$S(n)$		带数目减少 空间不变
M_2	1带	$S(n)$		
M_1	k带	$T(n)$		带数目减少 时间增加
M_2	1带	$T^2(n)$		
M_2	2带	$T(n)\log T(n)$		

7.1 P类与NP类

7.1.1 易解的问题与难解的问题

排序$O(n\log n)$，Dijkstra算法 $O(n^2)$，最大团回溯法 $O(n2^n)$

用一台每秒 10亿 (10^9) 次的超大型计算机，上述算法的时间：

10万个数据排序，$10^5\times\log_2 10^5\approx 1.7\times 10^6$，$t=1.7\times 10^{-3}$秒

1万个顶点的图的单源最短路径，$(10^4)^2=10^8$，$t=0.1$秒

100个顶点的图的最大团，$100\times 2^{100}\approx 1.8\times 10^{32}$，

$t=1.8\times 10^{32}/10^9=1.8\times 10^{21}$秒$=5.7\times 10^{15}$年，即5千7百万亿年!

1分钟能解多大的问题．1分钟 6×10^{10} 次运算

可给2×10^9 =20亿个数据排序

用Dijkstra算法可解2.4×10^5个顶点的图的单源最短路径问题.

回溯法1天只能解41个顶点的图的最大团问题.

北京大学

算法的时间复杂度

函数 f 和 g 是多项式相关的: 如果存在多项式 p 和 q 使得对任意的 $n \in \mathbf{N}$, $f(n) \le p(q(n))$ 和 $g(n) \le q(f(n))$.

例如 $n\log n$ 与 n^2, n^2+2n+5 与 n^{10} 都是多项式相关的,

$\log n$ 与 n, n^5 与 2^n 不是多项式相关的.

问题Π 的实例 I 的规模: I 的二进制编码的长度, 记作$|I|$.

定义7.1 如果存在函数 $f:\mathbf{N}\to\mathbf{N}$使得 对任意的规模为 n 的实例 I, 算法 A 对 I 的运算在 $f(n)$ 步内停止, 则称算法 A的时间复杂度为 $f(n)$.

多项式时间算法: 以多项式为时间复杂度.

易解的问题: 有多项式时间算法.

难解的问题: 不存在多项式时间算法.

北京大学

几点说明

1. 当采用合理的编码时, 输入的规模都是多项式相关的.

“合理的”是指在编码中不故意使用许多冗余的字符.

例如, 设实例 I 是一个无向简单图 $G = \langle V,E \rangle$,

$V=\{ a, b, c, d \}$, $E=\{ (a,b), (a,d), (b,c), (b,d), (c,d) \}$

用邻接矩阵表示, 编码 e_1=0101/1011/0101/1110/, 长度 20.

用关联矩阵表示, 编码 e_2=11000/10110/00101/01011/, 长度 24.

G 有n个顶点m条边,

用邻接矩阵时$|I|=n(n+1)$, 用关联矩阵时$|I|=n(m+1)$.

两者多项式相关.

2. 自然数应采用k ($k\geq 2$)进制编码, 不能采用一进制编码.

n 的二进制编码有$\lceil \log_2(n+1) \rceil$位, 一进制编码有$n$位, 两者不是多项式相关的.

北京大学

几点说明

3. 时间复杂度常表成计算对象的某些自然参数的函数，如图的顶点数或边数的函数. 实例的二进制编码的长度与这些自然参数通常是多项式相关的.

4. 运行时间通常是计算执行的 操作指令数，执行的指令数与实际运行时间是多项式相关的.

(1) 要求每一条指令的执行时间是固定的常数.

(2) 规定一个基本操作指令集，可由位逻辑运算与、或、非组成，任何可用这个基本操作指令集中常数条指令实现的操作都是合理的指令，由有限种合理的指令构成的操作指令集是合理的操作指令集.

在上述约定下, 算法是否是多项式时间的与采用的编码和操作指令集无关， 从而一个问题是易解的、还是难解的也与采用的编码和操作指令集无关.

易解的问题与难解的问题

易解的问题.

如排序、最小生成树、单源最短路径等

已证明的难解问题.

(1) 不可计算的, 即根本不存在求解的算法, 如希尔伯特第十问题——丢番图方程是否有整数解.

(2) 有算法 但至少需要指数或更多的时间或空间，如带幂运算的正则表达式的全体性, 即任给字母表A上的带幂运算的正则表达式R, 问:$\langle R\rangle=A^*$?这个问题至少需要指数空间.

既没有找到多项式时间算法、又没能证明是难解的问题.

如哈密顿回路问题、货郎问题、背包问题等

7.1.2 判定问题

判定问题: 答案只有两个——是, 否.

判定问题 $\Pi = <D_\Pi, Y_\Pi>$, 其中D_Π是实例集合, $Y_\Pi \subseteq D_\Pi$是所有答案为“Yes”的实例.

哈密顿回路 (HC): 任给无向图G, 问G有哈密顿回路吗?

货郎问题 (TSP): 任给 n 个城市, 城市 i 与城市 j 之间的正整数距离 $d(i,j)$, $i \neq j$, $1 \leq i, j \leq n$, 以及正整数 D, 问有一条每一个城市恰好经过一次最后回到出发点且长度不超过 D 的巡回路线吗? 即, 存在 $1, 2, \ldots, n$ 的排列 σ 使得

$$\sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1)) \leq D ?$$

0-1背包的判定问题与优化问题

0-1背包: 任给 n 件物品和一个背包, 物品 i 的重量为 w_i, 价值为 v_i, $1 \le i \le n$, 以及背包的重量限制 B 和价值目标 K, 其中 w_i, v_i, B, K 均为正整数, 问能在背包中装入总价值不少于 K 且总重量不超过 B 的物品吗? 即, 存在子集 $T \subseteq \{1,2,\ldots,n\}$ 使得

$$\sum_{i \in T} w_i \le B \quad \text{且} \quad \sum_{i \in T} v_i \ge K ?$$

搜索问题, 组合优化问题与判定问题的对应.

如果搜索问题, 组合优化问题有多项式时间算法, 则对应的判定问题也有多项式时间算法; 通常反之亦真.

北京大学

组合优化问题与判定问题

组合优化问题 Π^* 由3部分组成:

(1) 实例集D_{Π^*}

(2) $\forall I \in D_{\Pi^*}$, 有一个有穷非空集 $S(I)$, 其元素称作 I 的可行解

(3) $\forall s \in S(I)$, 有一个正整数 $c(s)$, 称作 s 的值

如果 $s^* \in S(I)$, 对所有的 $s \in S(I)$, 当Π^*是最小 (大) 化问题时,

$$c(s^*) \le c(s) \qquad (\ c(s^*) \ge c(s)\ )$$

则称 s^*是 I 的最优解, $c(s^*)$是 I 的最优值, 记作$\mathrm{OPT}(I)$.

Π^*对应的判定问题 $\Pi = \langle D_\Pi, Y_\Pi \rangle$定义如下:

$D_\Pi = \{ (I, K) \mid I \in D_{\Pi^*}, K \in Z^* \}$, 其中$Z^*$是非负整数集合.

当Π^*是最小化问题时, $Y_\Pi = \{ (I, K) \mid \mathrm{OPT}(I) \le K \}$;

当Π^*是最大化问题时, $Y_\Pi = \{ (I, K) \mid \mathrm{OPT}(I) \ge K \}$.

北京大学

7.1.3 NP类 (nondeterministic polymial)

定义7.2 所有多项式时间可解的判定问题组成的问题类称作 P类.

定义7.3 设判定问题 $\Pi = <D,Y>$, 如果存在两个输入变量的多项式时间算法 A 和多项式 p，对每一个实例 $I \in D$, $I \in Y$ 当且仅当存在 t, $|t| \leq p(|I|)$, 且 A 对输入 I 和 t 输出“Yes”, 则称Π是多项式时间可验证的, A是Π的多项式时间验证算法, 而当 $I \in Y$ 时, 称 t 是 $I \in Y$ 的证据.

由所有多项式时间可验证的判定问题组成的问题类称作 NP类.

HC(哈密顿回路), TSP(货郎), 0-1背包∈NP

非确定型多项式时间算法

非确定型多项式时间算法

(1) 对给定的实例 I, 首先“猜想”一个 t, $|t| \le p(|I|)$

(2) 然后检查 t 是否是证明 $I \in Y$ 的证据

(3) 猜想和检查可以在多项式时间内完成

(4) 当且仅当 $I \in Y$ 时能够正确地猜想到一个证据 t

*注：非确定型多项式时间算法不是真正的算法

定理7.1　$P \subseteq NP$

问题：P=NP?

7.2 多项式时间变换与NP完全性

7.2.1 多项式时间变换

如何比较两个问题的难度?

定义7.4　设判定问题$\Pi_1 = \langle D_1, Y_1 \rangle$, $\Pi_2 = \langle D_2, Y_2 \rangle$.　如果函数$f: D_1 \to D_2$满足条件:

(1) f是多项式时间可计算的,

(2) 对所有的$I \in D_1$, $I \in Y_1 \Leftrightarrow f(I) \in Y_2$,

则称f是Π_1到Π_2的多项式时间变换.

如果存在Π_1到Π_2的多项式时间变换, 则称Π_1可多项式时间变换到Π_2, 记作$\Pi_1 \leq_p \Pi_2$.

例

例7.1 $HC \leq_p TSP$.

证 对 HC 的每一个实例 I: 无向图 $G=<V,E>$, TSP 对应的实例 $f(I)$ 为: 城市集 V, 任意两个不同的城市 u 和 v 之间的距离

$$d(u,v)=\begin{cases}1, & \text{若}(u,v)\in E,\\ 2, & \text{否则},\end{cases}$$

以及界限 $D = |V|$.

例

最小生成树: 任给连通的无向赋权图 $G=<V,E,W>$ 以及正整数 B, 其中权 $W:E\to Z^+$, 问有权不超过 B 的生成树吗?

最大生成树: 任给连通的无向赋权图 $G=<V,E,W>$ 以及正整数 D, 其中权 $W:E\to Z^+$, 问 G 有权不小于 D 的生成树吗?

例7.2 最大生成树$\le_p$最小生成树.

证 任给最大生成树的实例 I:连通的无向赋权图 $G=<V,E,W>$ 和正整数D, 最小生成树的对应实例 $f(I)$: 图$G'=<V,E,W'>$和正整数 $B=(n-1)M-D$, 其中

$$n=|V|,\quad M=\max\{W(e)\mid e\in E\}+1,\quad W'(e)=M-W(e)$$

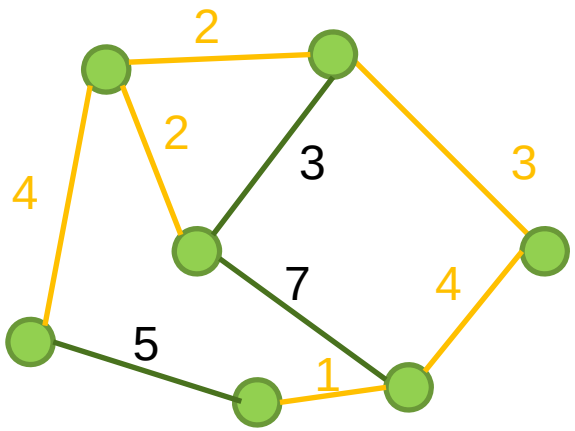
如果存在G的生成树T, 使得 $\sum_{e\in T}W(e)\ge D$, 则

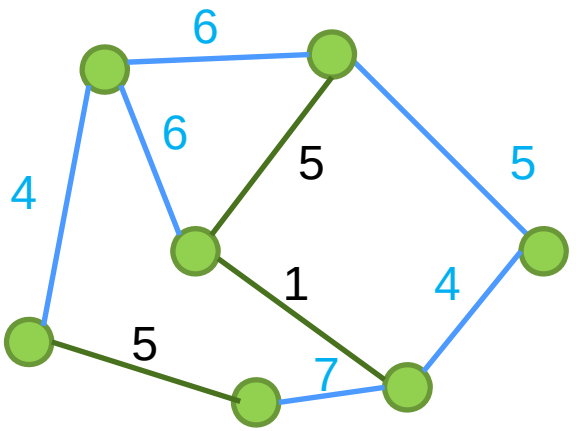
$$\sum_{e\in T}W'(e)=(n-1)M-\sum_{e\in T}W(e)\le(n-1)M-D=B.$$

反之亦然. f 多项式时间可计算.

北京大学

变换实例





$D=12$

最大生成树T
的实例 G

$B=6\times 8-12=36$

最小生成树T'
的实例G'

$$M=8,\quad W(T)=16\geq 12,\quad W(T')=32\leq 36$$

$$\sum_{e\in T} W'(e)=(n-1)M-\sum_{e\in T'} W(e)$$

北京大学

$\leq_p$的性质

定理7.2 $\leq_p$具有传递性. 即, 设 $\Pi_1 \leq_p \Pi_2$, $\Pi_2 \leq_p \Pi_3$, 则$\Pi_1 \leq_p \Pi_3$.

证　设 $\Pi_i = <D_i, Y_i>$, $i=1,2,3$, f 和 g是 Π_1 到 Π_2 和 Π_2 到 Π_3 的多项式时间变换. 对每一个$I \in D_1$, 令$h(I)=g(f(I))$.

计算 f 和 g 的时间上界分别为多项式 p 和 q, 不妨设 p 和 q 是单调递增的. 计算 h 的步数不超过 $p(|I|) + q(|f(I)|)$. 输出作为合理的指令, 一步只能输出长度不超过固定值 k 的字符串, 因而 $|f(I)| \leq k\, p(|I|)$. 于是,

$$p(|I|) + q(|f(I)|) \leq p(|I|) + q(kp(|I|)),$$

得证h是多项式时间可计算的.

对每一个$I \in D_1$,

$$I \in Y_1 \Leftrightarrow f(I) \in Y_2 \Leftrightarrow h(I) = g(f(I)) \in Y_3,$$

得证 h是 Π_1 到 Π_3 的多项式时间变换.

北京大学

$\leq_p$的性质

定理7.3　设 $\Pi_1 \leq_p \Pi_2$, 则 $\Pi_2 \in$P 蕴涵 $\Pi_1 \in$P.

证　设 $\Pi_1=<D_1,Y_1>$, $\Pi_2=<D_2,Y_2>$, f 是 Π_1 到 Π_2 的多项式时间变换, A 是计算 f 的多项式时间算法. 又设 B 是 Π_2 的多项式时间算法. 如下构造 Π_1 的算法 C:

(1) 对每一个 $I \in D_1$, 应用 A 得到 $f(I)$,

(2) 对 $f(I)$ 应用 B,

(3) C 输出“Yes”当且仅当 B 输出“Yes”.

推论7.1　设 $\Pi_1 \leq_p \Pi_2$, 则 Π_1 是难解的蕴涵 Π_2 是难解的.

由例7.2 及最小生成树∈P, 得知最大生成树∈P.

由例7.1, 如果 TSP∈P, 则 HC∈P. 反过来, 如果 HC 是难解的, 则 TSP 也是难解的.

北京大学

7.2.2 NP完全性

定义7.5 如果对所有的 $\Pi' \in$ NP, $\Pi' \leq_p \Pi$, 则称 Π 是NP难的. 如果 Π 是 NP 难的且 $\Pi \in$ NP, 则称 Π 是 NP完全的.

定理7.4 如果存在NP难的问题 $\Pi \in$ P, 则 P=NP.

推论7.2 假设P≠NP, 那么, 如果 Π 是NP难的, 则 $\Pi \notin$ P.

定理7.5 如果存在NP难的问题 Π' 使得 $\Pi' \leq_p \Pi$, 则 Π 是NP难的.

推论7.3 如果 $\Pi \in$ NP并且存在 NP完全问题 Π' 使得 $\Pi' \leq_p \Pi$, 则 Π 是NP完全的.

证明NP完全性的“捷径”

(1) 证明 $\Pi \in$ NP;

(2) 找到一个已知的NP完全问题 Π', 并证明 $\Pi' \leq_p \Pi$.

北京大学

7.2.3 Cook-Levin定理

合式公式：由变元、逻辑运算符以及圆括号按照一定的规则组成的表达式.

变元和它的否定称作文字.

有限个文字的析取称作简单析取式.

有限个简单析取式的合取称作合取范式.

给定每一个变元的真假值称作一个赋值. 如果赋值 t 使得合式公式 F 为真, 则称 t 是 F 的成真赋值.

如果 F 存在成真赋值, 则称 F 是可满足的.

例如 $F_1=(x_1\vee x_2)\wedge(\neg x_1\vee x_2\vee x_3)\wedge\neg x_2$ 是一个合取范式.

令$t(x_1)=1, t(x_2)=0, t(x_3)=1$是$F_1$的成真赋值, F_1是可满足的.

$F_2=(x_1\vee\neg x_2\vee x_3)\wedge(\neg x_1\vee\neg x_2\vee x_3)\wedge x_2\wedge\neg x_3$不是可满足的.

北京大学

Cook-Levin定理

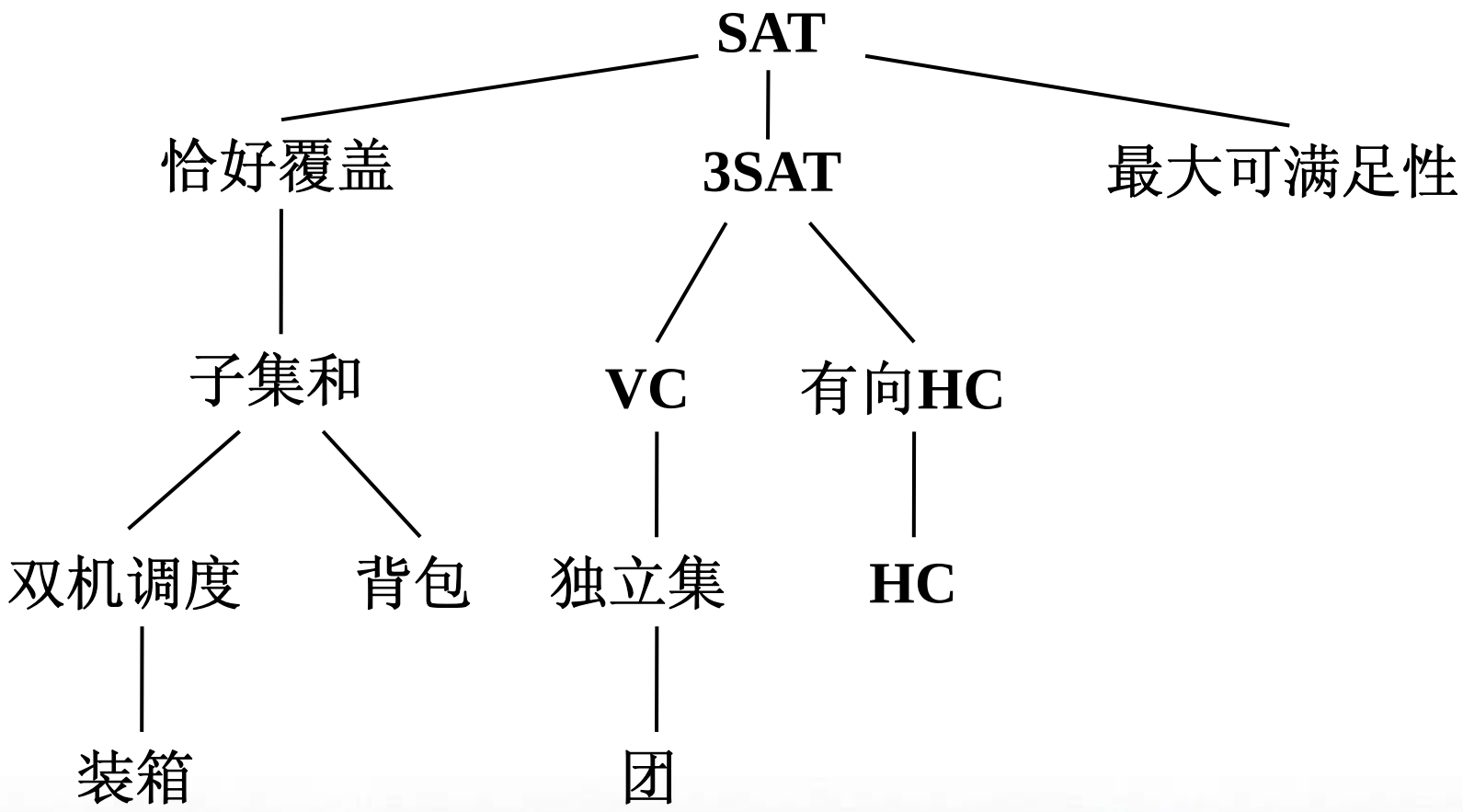
可满足性 (SAT): 任给一个合取范式 F, 问 F 是可满足的吗?

定理7.6 (Cook-Levin定理) SAT 是NP完全的.

证明思想：对于任意一个NP类中语言L，存在一个接受它的非确定型图灵机M. 构造L到SAT的多项式变换. 对于L中的任意串x，M对x的接受计算变换成一个SAT问题的肯定实例

定理7.7 P=NP 的充分必要条件是存在 NP 完全问题 $\Pi \in$ P.

7.3 几个NP完全问题



7.3.1 最大可满足性与三元可满足性

最大可满足性(MAX-SAT): 任给关于变元 $x_1, x_2, \ldots, x_n$ 的简单析取式 $C_1, C_2, \ldots, C_m$ 及正整数 K, 问存在关于变元 $x_1, x_2, \ldots, x_n$ 的赋值使得 $C_1, C_2, \ldots, C_m$ 中至少有 K 个为真吗?

3元合取范式: 每一个简单析取式恰好有3个文字的合取范式.
三元可满足性(3SAT): 任给一个3元合取范式F, 问F是可满足的吗?

子问题: 设判定问题 $\Pi = \langle D, Y \rangle$, $\Pi' = \langle D', Y' \rangle$, 如果 $D' \subseteq D$, $Y' = D' \cap Y$, 则 Π' 是 Π 的特殊情况, 称作 Π 的子问题.

SAT是MAX-SAT的子问题(取$K=m$), 3SAT是SAT的子问题.

定理7.8 MAX-SAT是NP完全的
定理7.9 3SAT是NP完全的.

7.3.2 顶点覆盖、团与独立集

设无向图$G=<V,E>$, $V'\subseteq V$. V'是G的一个

顶点覆盖: G的每一条边都至少有一个顶点在V'中.

团: 对任意的$u,v\in V'$且$u\neq v$, 都有$(u,v)\in E$.

独立集: 对任意的$u,v\in V'$, 都有$(u,v)\notin E$.

引理7.1 对任意的无向图$G=<V,E>$和子集$V'\subseteq V$, 下述命题是等价的:

(1) V'是G的顶点覆盖,

(2) $V-V'$是G的独立集,

(3) $V-V'$是补图 $G^c=<V,E^c>$的团.

顶点覆盖、团与独立集

顶点覆盖(VC): 任给一个无向图$G=<V,E>$和非负整数 $K\leq|V|$, 问G 有顶点数不超过 K 的顶点覆盖吗?

团: 任给一个无向图$G=<V,E>$和非负整数 $J\leq|V|$, 问G 有顶点数不小于J 的团吗?

独立集: 任给一个无向图 $G=<V,E>$和非负整数 $J\leq|V|$, 问G 有顶点数不小于J 的独立集吗?

定理7.10　顶点覆盖是 NP 完全的.

定理7.11　独立集和团是 NP 完全的.

7.3.3 -7.34 哈密顿回路、货郎问题与恰好覆盖

有向哈密顿回路: 任给有向图D, 问:D中有哈密顿回路吗?

定理7.12 有向HC是NP完全的.

定理7.13 HC是NP完全的.

定理7.14 TSP是NP完全的.

恰好覆盖: 给定有穷集 $A=\{a_1, a_2,\ldots, a_n\}$和 A的子集的集合$W=\{S_1, S_2,\ldots, S_m\}$, 问: 存在子集 $U\subseteq W$ 使得 U 中子集彼此不交且它们的并集等于A? 称U是 A的恰好覆盖.

例如, 设$A=\{1,2,3,4,5\}$, $S_1=\{1,2\}$, $S_2=\{1,3,4\}$, $S_3=\{2,4\}$, $S_4=\{2,5\}$, 则$\{S_2, S_4\}$是A的恰好覆盖.

定理7.15 恰好覆盖是NP完全的.

北京大学

7.3.5 子集和、背包、装箱与双机调度

子集和: 给定正整数集合 $X=\{x_1, x_2, \ldots, x_n\}$及正整数$N$, 问存在$X$的子集 T, 使得T 中的元素之和等于N 吗?

装箱: 给定 n 件物品, 物品j的重量为正整数 w_j, $1 \le j \le n$, 以及箱子数 K. 规定每只箱子装入物品的总重量不超过正整数 B, 问能用 K 只箱子装入所有的物品吗?

双机调度: 有2台机器和 n 项作业 $J_1, J_2, \ldots, J_n$, 这2台机器完全相同, 每一项作业可以在任一台机器上进行, 没有先后顺序, 作业 J_i 的处理时间为 t_i, $1 \le i \le n$, 截止时间为D, 所有 t_i 和 D 都是正整数, 问能把 n 项作业分配给这2台机器, 在截止时间D内完成所有的作业吗?

NP完全性

定理7.16 子集和是NP完全的.

定理7.17 0-1背包是NP完全的

定理7.18 双机调度是NP完全的.

定理7.19 装箱是NP完全的.

注意: 0-1背包问题优化形式的动态规划算法, 其时间复杂度为$O(nB)$, 其中n是物品的个数, B是重量限制. 这不是多项式时间算法, 而是指数时间算法.

伪多项式时间算法: 算法的时间复杂度以 $|I|$ 和 $\max(I)$ 的某个二元多项式 $p(|I|, \max(I))$ 为上界, 其中 $\max(I)$ 是实例 I 中数的最大绝对值.

NP完全性理论的应用

用NP完全性理论进行子问题分析

子问题的计算复杂性

子问题的**NP**完全性证明

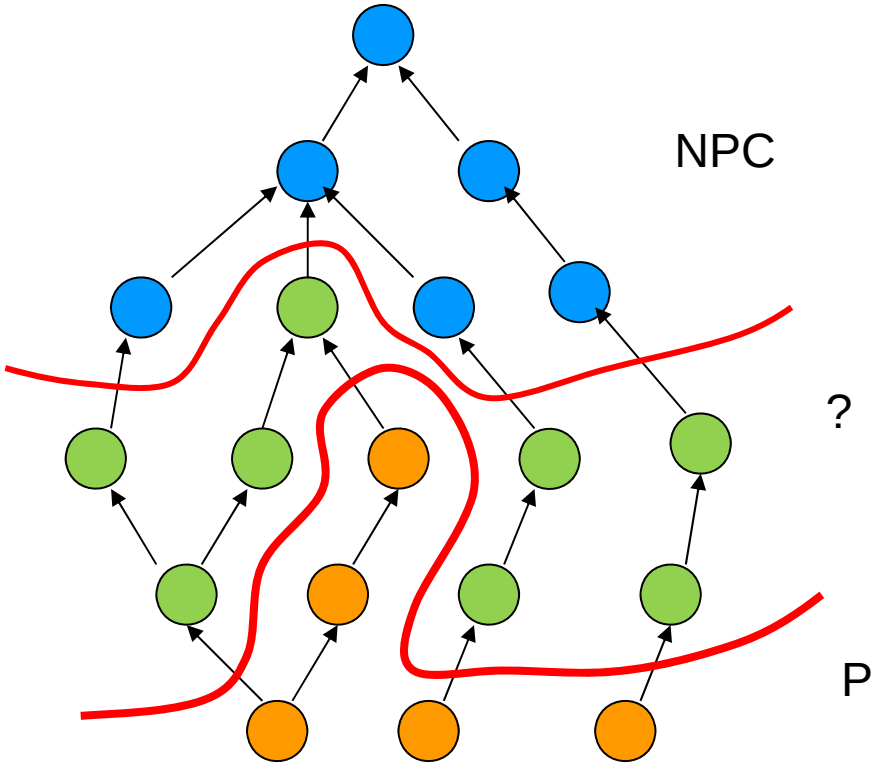
NP难度

搜索问题

Turing归约

NP-hard, NP-easy

子问题的计算复杂性



努力扩大已知区域，缩小未知区域

当P≠NP时，存在不属于NPC也不属于P的问题

有先行约束的多处理机调度问题

优化问题:

给定任务集 T, m 台机器,$\forall t\in T$, $l(t)\in Z^+$, T上的偏序$\prec$.

若 $\sigma: T\to\{0, 1, \ldots, D\}$满足下述条件, 则称 T 为可行调度.

(1) $\forall t\in T,\ \sigma(t)+l(t)\le D$

(2) $\forall i, 0\le i\le D, |\{t\in T : \sigma(t)\le i<\sigma(t)+l(t)\}|\le m$

(3) $\forall t, t'\in T,\ t\prec t' \Leftrightarrow \sigma(t)+l(t)\le\sigma(t')$

求使得 D 最小的可行调度.

条件说明:

任务在截止时间前完成

同时工作的台数不超过 m

有偏序约束的任务必须按照约束先行

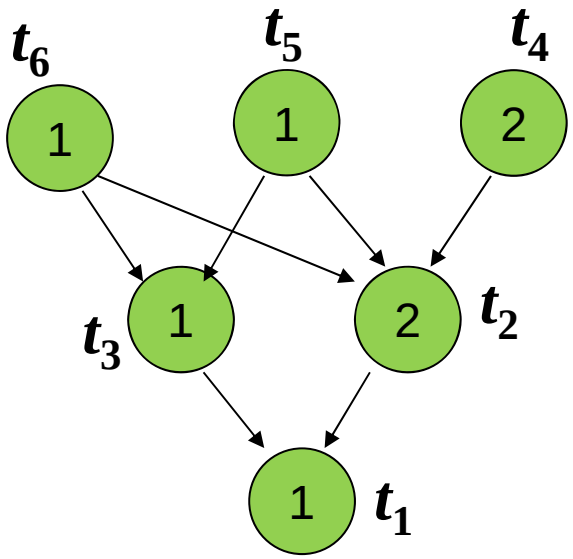
北京大学

实例

任务集如图所示

$m=2$

求使得 D 最小的可行调度.



$D=6$

$D=5$

判定问题

实例：任务集T, m台机器, $\forall t \in T$, $l(t) \in Z^+$, T上的偏序$\prec$,
截止时间 $D \in Z^+$.

问：是否存在小于等于 D 的可行调度？

子问题通过限制参数(机器数、工作时间、偏序)构成

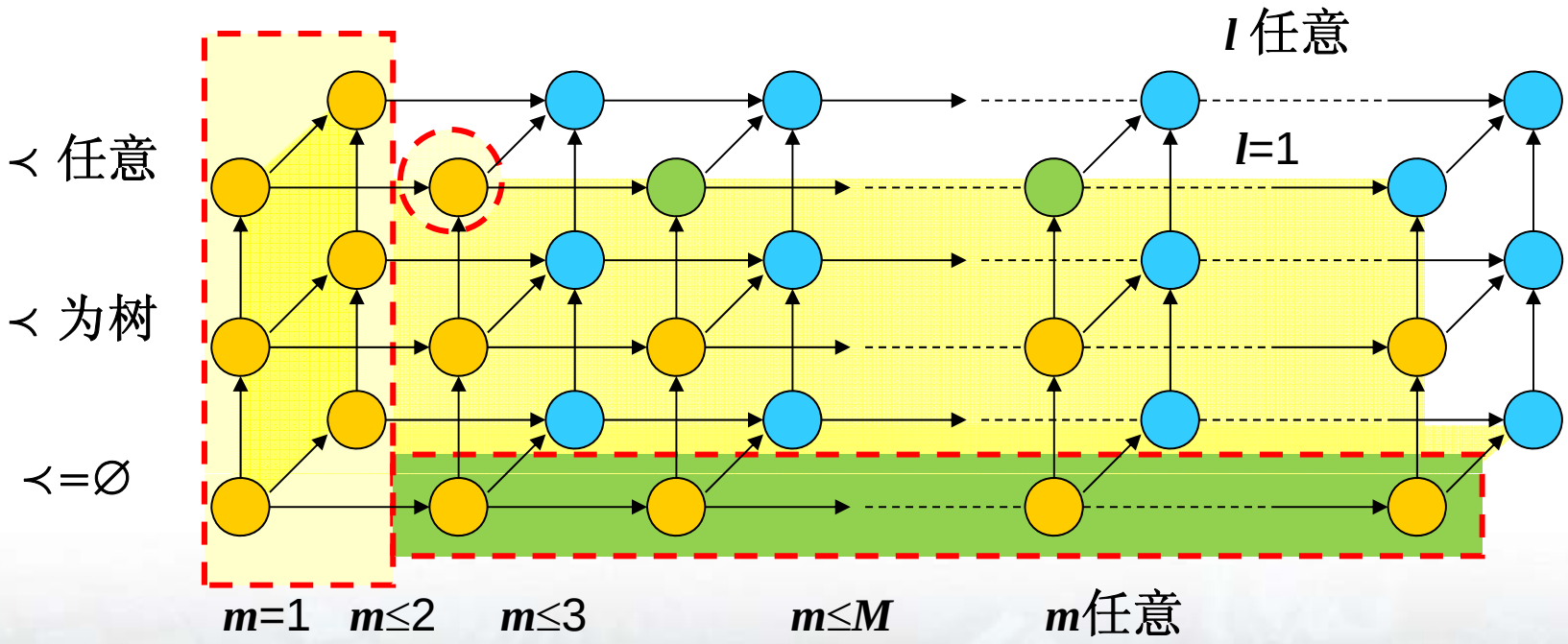
参数	限　制		
偏序	∅	树	任意
m大小	$m \le 1, 2, \ldots,$ 某个常数		m 任意
l 大小	l 为常数		l 任意

调度问题的子问题结构

从上到下：偏序任意、树形偏序、无偏序约束

从左到右：处理器台数限制逐步放大

从前到后：各任务等长工作时间、任意工作时间



北京大学

搜索问题与优化问题的难度

Turing归约

设π_1, π_2是搜索问题，*A* 是利用解 π_2的假想子程序 *s* 解π_1的算法，且只要 *s* 是多项式时间的，*A*也是多项式时间的，则称算法 *A* 是从π_1到π_2的多项式时间的 **Turing归约**. 这时也称 π_1 Turing归约到 π_2，记作 $\pi_1 \propto_T \pi_2$.

NP难度

设π是搜索问题，如果存在 NP完全问题π' 使得 $\pi' \propto_T \pi$，则称π是**NP-hard**. 这意味着在多项式可计算的角度看，π至少和 NPC问题一样难.

许多NP完全问题对应的优化问题都是NP-hard

实例—证明NP等价

例1 货郎问题（TSO）是 NP等价的.

证：易证 **TSO是 NP-hard.** 下面证明 **TSO** 是**NP-easy.**

引入中间问题：巡回售货员的延伸问题（**TSE**）

TSE

实例：有穷城市的集合 $C = \{ c_1, c_2, \dots, c_m \}$

距离 $\forall c_i, c_j \in C, d(c_i, c_j) \in Z^+$,

长度限制 $B \in Z^+$,

部分旅行路线 $\vartheta = < c_{\pi(1)}, \dots, c_{\pi(k)} >$

问：ϑ是否可以延伸成全长不超过 B 的全程旅行

$< c_{\pi(1)}, \dots, c_{\pi(k)}, c_{\pi(k+1)}, \dots, c_{\pi(m)} >$?

易证 **TSE**属于**NP.**

TSO到TSE的Turing归约

设 $s(C, d, \vartheta, B)$ 是解 TSE 的子程序，其中 C 为城市集，d 为距离函数，ϑ为部分旅行，B为长度限制.

下面构造解TSO的算法.

思路:

用二分法确定最短路旅行长度 B^*

旅行长度界于$m \to m \times d,\ d = \max\{ d(c_i, c_j)\}$

每次取中点值验证是否存在能延伸到此长度的旅行

根据最小长度值 B^*确定旅行路线

从c_1开始，依次检查$<c_1,c_2>$,$<c_1,c_3>$…是否能延伸到 B^* 长度的旅行，选择第一个可延伸的顶点c_i.

按照上面方法确定后面的其他顶点.

北京大学

算法 MinLength

设 $s(C,d,\vartheta,B)$是解 TSE 的子程序，其中 C 为城市集，d为距离函数，ϑ为部分旅行，B为长度限制.

算法 Minlength （二分法确定最短旅行长度 B^*）

1. 令$Bmin \leftarrow m$, $Bmax \leftarrow m \cdot \max\{ d(c_i,c_j): c_i, c_j \in C \}$
2. 若$Bmax - Bmin = 1$, 则 $B^* \leftarrow Bmax$, 结束
3. $B \leftarrow [(Bmin+Bmax)/2]$
4. $s(C,d,<c_1>,B)$
5. 若回答”Yes”, $Bmax \leftarrow B$, 否则$Bmin \leftarrow B$
6. 转2.

$B_{min}=m$

$B=(B_{min}+B_{max})/2$

$B_{max}=md$

Yes

$B_{min}=m$

$B_{max}=B$

No

$B_{min}=B$

$B_{max}=md$

return

B_{min} $B^*=B_{max}$

北京大学

算法 FindSolution

算法 FindSolution (找解)

1. $i \leftarrow 2, M \leftarrow \{2, 3, \ldots, m\}$
2. $j \leftarrow M$ 中的最小值
3. $\vartheta = <c_1, c_j>$
4. $s(C, d, \vartheta, B^*)$
5. if 回答”Yes”
6. then $i \leftarrow i+1$, $M \leftarrow M-\{j\}$
7. else
8. 从ϑ中去掉 c_j
9. 从 M 中选择大于 j 的最小值 k
10. 将 c_k 加入到 ϑ 的最后项
11. $M \leftarrow M \cup \{j\}$
12. 如果 $i \leq m$, 转4；否则停机

复杂度分析

至多m–2次调用s可以找到第2个城市，至多m–3次调用s可以找到第3个城市，…，至多1次调用s可以找到第m–1个城市。调用s的总次数至多为

$$(m-2)+(m-3)+...+1=\frac{(m-1)(m-2)}{2}$$

为 m 的多项式，而 TSO 的实例规模为 $m+\log B\max$，所以是从 TSO 到 TSE 的 Turing归约.

而 TSE 是 NPC 问题，因为判定问题 TSP 是 TSE的子问题，相当 $\vartheta = <c_1>$的情况. 因此，TSO Turing归约到 NP 问题 TSE，从而证明了TSO 是 NP-easy.

类似地可以证明六个基本NPC问题对应的优化问题都是 NP-hard.