

# 现代优化计算方法

清华大学数学科学系 邢文训

答疑时间：周四下午4：00-5：00

地点：理科楼数学系A416.

[wxing@tsinghua.edu.cn](mailto:wxing@tsinghua.edu.cn)

Tel: 62787945

# 1.5 NP, NP完全和NP难

## ■ 三个概念

- NP——nondeterministic polynomial
- NP完全——NP-Complete
- NP难——NP-hard

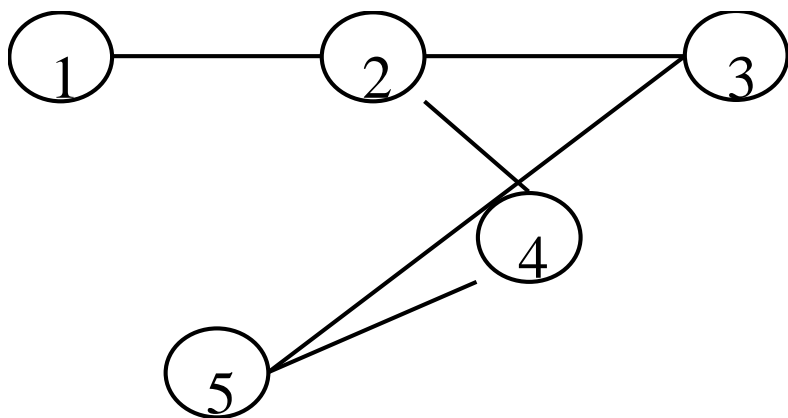
## ■ 复习

- 问题——problem
- 实例——instance
- 规模——size
- 多项式问题

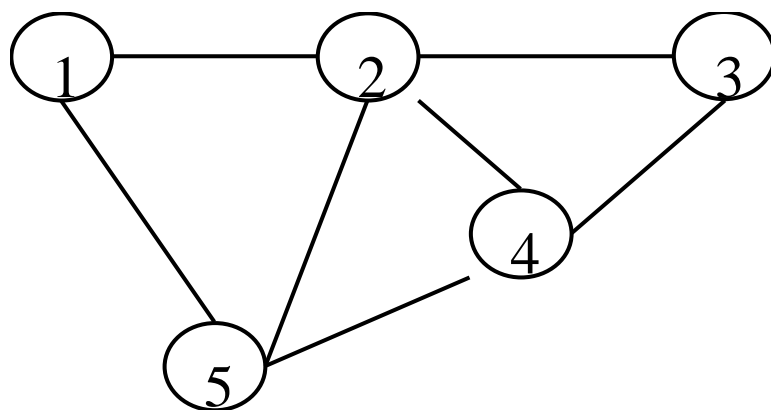
# 判定问题(Decision problem)

- 如果一个问题的每一个实例只有“是”或“否”两种答案，则称这个问题为判定问题。
  - 答案“是”的实例为“是”实例。
  - 答案为“否”的实例为“否”或“非”实例。
- 现在是否在下雨？
  - 判定问题
- 今天能否下雨？
  - 非判定问题

**Hamilton回路问题：** 给定 $n$ 个节点及一些节点间的连线构成的图，是否存在一个连接所有节点的闭路(即圈)，使得每个节点在闭路（圈）上只出现一次？



“否” 实例



“是” 实例

## 3-exact cover

三精确覆盖：已知集合  $S = \{u_1, u_2, \dots, u_{3m}\}$  的  $n$  个子集构成的子集族  $F = \{S_1, S_2, \dots, S_n\}$ ，其中每个子集恰好包含  $S$  中三个元素， $F$  中是否存在  $m$  个子集  $S_{i_1}, S_{i_2}, \dots, S_{i_m}$ ，使得  $\bigcup_{j=1}^m S_{i_j} \supseteq S$ ？若存在  $m$  个子集  $S_{i_1}, S_{i_2}, \dots, S_{i_m}$  满足  $\bigcup_{j=1}^m S_{i_j} \supseteq S$ ，称  $m$  个子集  $S_{i_1}, S_{i_2}, \dots, S_{i_m}$  覆盖  $S$ 。

在逻辑运算中，布尔变量  $x$  的取值只有两个：“真”和“假”，逻辑运算有“或 (+)”，“与 ( $\bullet$ )”和“非 ( $\neg$ )”。如果“真”用“1”表示，“假”用“0”表示，布尔运算遵循。

运算 “+”

变量值		结果
0	0	0
0	1	1
1	0	1
1	1	1

运算 “ $\bullet$ ”

变量值		结果
0	0	0
0	1	0
1	0	0
1	1	1

运算 “ $\neg$ ”

变量值	结果
0	1
1	0

$n$ 个布尔变量:  $\{x_1, x_2, \dots, x_n\}$

文字集:  $\{x_1, x_2, \dots, x_n; \bar{x}_1, \bar{x}_2, \dots, \bar{x}_n\}$

句子: 文字集中任意一子集各元素的“或”运算。

表达式:  $m$ 个句子的“与”运算。

适定的: 存在真值分配的表达式

**适定性问题:** 给定包含  $n$  个布尔变量的  $m$  个句子  $c_1, c_2, \dots, c_m$ , 布尔表达式  $c_1 \bullet c_2 \bullet \dots \bullet c_m$  是适定的吗?

# Satisfiability

- 最优化问题与判定问题的关系：两个阶段处理，首先，在给定某个目标值后，讨论优化问题对应的判定问题，然后再通过这个目标值的变化确定原问题的最优值。

- 最优化问题 
$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \geq 0 \\ & x \in D. \end{aligned}$$

- 转换后的判定问题

$$S = \{x \in D \mid f(x) \leq z, g(x) \geq 0\} \neq \emptyset?$$



线性规划问题(LP)对应的判定问题——LP 判定问题: 给定  $z$ , 是否有  $\{x | c^T x \leq z, Ax = b, x \geq 0\} \neq \emptyset$ ?

TSP判定问题为: 给定  $z$ , 是否存在  $n$  个城市的一个排列  $W = (i_1, i_2, \dots, i_n)$ , 使得

$$f(W) = \sum_{j=1}^n d_{i_j i_{j+1}} \leq z ?$$

# 表达式(structure)——解的形式

- 针对一个问题。
- 存在一个多项式函数 $g(x)$ 。
- 解的形式：一个字符串结构(structure)。
  - 对每一个实例，长度 $O(g(l(I)))$ 。
- 一个验证算法
  - 对上述结构的每一字符串，计算告知“是”或“否”。  
计算复杂性 $O(g(l(I)))$ 。
- “是”实例
  - 充分必要存在结构的一个字符串，经过验证算法告知“是”。

# “结构” 举例1

- 给定 $n$ 个正整数 $\{a_i\}$ 及 $b$ ，问题：这 $n$ 个数中最大数不超过 $b$ 吗？
  - 结构1：  $n$ 个数的结构。
    - 算法： 逐一同 $b$ 比较。
    - “是”实例充分必要条件在这个结构的字符串中每一个数都不超过 $b$ 。
  - 结构2： 一个数 $c$ 。
    - 算法：  $c$ 与 $b$ 比较。
    - “是”实例充分必要条件 $c \leq b$ 。
    - 如何得到 $c$ 并没有明确给出。

# “结构” 举例2

- 背包问题
- 优化问题对应的判定问题：给定 $c$ (价值),  $a$ (尺寸),  $b$ (容量)和一个给定值 $z$ , 是否有
$$\{x \in \{0,1\}^n \mid a^T x \leq b, c^T x \geq z\} \neq \emptyset?$$
- 结构 $n$ 个 $\{0, 1\}$ 字符串:  $(x_1, x_2, \dots, x_n)$ 。
- 算法：验证 $a^T x \leq b, c^T x \geq z$ 是否满足。
- “是”实例充要条件为有一个字符串满足。
- 如何找到这个字符串？不关心！！！！

# 求解实例的两个阶段

- 第一阶段：猜测阶段(guessing stage)
  - 对于“是”实例，目标是能够找到回答“是”实例的该结构一个字符串（解）。
- 第二个阶段：检查或验证阶段(verifiability)
  - 一旦字符串确定，将猜测的“是”的字符串作为输入，验证此是否为该实例“是”的回答。
- 寻求一个实例“是”表达字符串的难度是由问题本身所决定的。

# NP问题——非确定多项式问题

- 一个判定问题称为NP问题，表达式存在！

# NP问题等价叙述

- 对一个判定问题，若存在一个多项式函数 $g(x)$ 和一个验证算法 $A$ ，满足：
- 存在一个字符串的结构， $I$ 为判定问题“是”实例的充分必要条件：存在一个字符串 $S$ 回答 $I$ 为“是”实例，其规模 $l(S)=O(g(l(I)))$ ，其中 $l(I)$ 为 $I$ 的规模，且算法 $A$ 验证 $S$ 为实例 $I$ 的“是”答案的计算时间为 $O(g(l(I)))$ 。
- 这个判定问题称非确定多项式的，简记为NP。用NP表示所有非确定多项式问题的集合。

# NP问题的进一步讨论

- 判定问题是否属于NP的关键是对“是”的判定实例，“存在”一个回答实例为“是”的字符串的结构和验证算法，这个字符串和算法都为多项式时间复杂性。
- 不需回答“存在”的可行解字符串是怎样得到的。
- 定义中两处用到多项式的限制，可以分别用两个多项式替代。
- 特别需要指出：对“否”实例，我们不讨论它的相关判定的结果。



# NP问题或“结构”的例子

- 有些是自然的0-1字符串
- 有些需要考虑特殊的结构，但都需要回归考虑二进制的长度问题。
  - TSP问题： $\{0, 1\}^{n(n-1)}$  字符串。由图论的模型（第二节）得到。长度： $O(n^2)$

## ■ 考虑有向TSP

## ■ 判定问题:

$$\{x \in \{0,1\}^{n \times (n-1)} \mid \sum_{i,j} d_{ij} x_{ij} \leq z, x \text{ is a Hamilton cycle}\} \neq \emptyset?$$

■ 结构:  $\{x_{ij}\} \in \{0,1\}^{n \times (n-1)}$ . 长度:  $O(n^2)$ 。

## ■ 算法验证:

$$\sum_{i,j} d_{ij} x_{ij} \leq z, x \text{ is a Hamilton cycle}$$

# Hamilton圈的判定

- 采用第二节的整数规划模型作为算法验证肯定不行，因需要验证指数个子圈。
- 选出 $x_{ij}$ 为1所有弧，若 $>n$ 或 $<n$ 结束，非圈，结束。  
当正好等于 $n$ 个，从结点1开始验证：若有 $>1$ 个或0个出弧，结束。否则到达下一个城市，重复上面验证过程，一旦没有走到最后一个城市而回到路线中的一点，结束。最后情况下给出H圈。
- 算法复杂性 $O(n^3)$ .（如何估计？）

# 例

- $n=5, \{x_{12}=1, x_{25}=1, x_{34}=1, x_{42}=1, x_{51}=1\}$ .
- 算法:
  - 五个1, 继续验证。
  - $L=\{1\}, x_{12}=1, L=\{1,2\}$ 。
  - 继续从2开始, 检索 $x_{25}=1, 5 \notin L, L=\{1,2,5\}$ .
  - 继续从5开始, 检索 $x_{51}=1, 1 \in L$ , 停止, 不是Hamilton圈。

有理系数的线性规划（LP）判定问题属于**NP**

判定问题：再给定一个常数 $z$ ,

$$\{x | c^T x \leq z, Ax \leq b, x \geq 0\} \neq \emptyset?$$

■ 不妨考虑标准形：  $\{x | c^T x \leq z, Ax = b, x \geq 0\} \neq \emptyset?$

■ 实例规模

□ 输入系数：  $m, n, z, c, A, b$

□  $l(I) \geq 3 + n + mn + m + \log_2 |P|$

■ **LP**一个基本可行解

□ 线性规划凸分析理论



## 线性规划(LP)的图解法

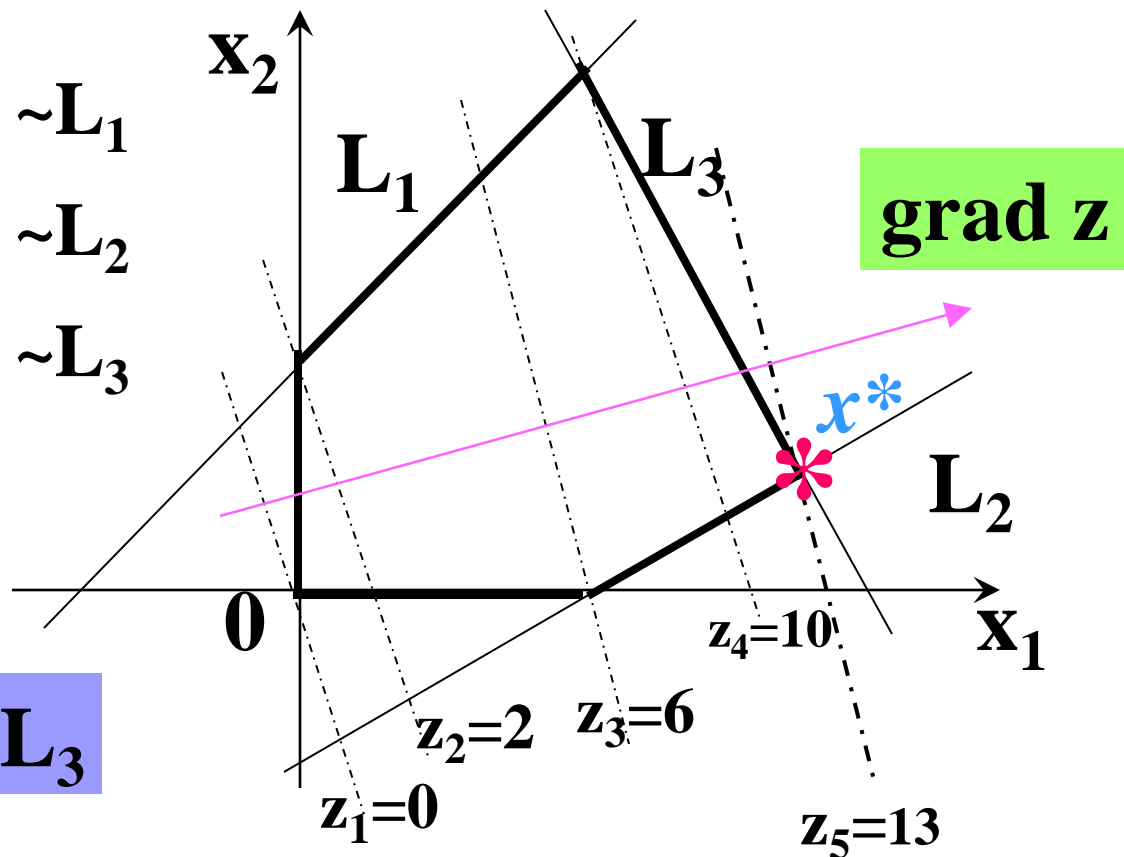
$$\max z = 3x_1 + x_2$$

$$s.t. \quad -x_1 + x_2 \leq 2 \quad \sim L_1$$

$$x_1 - 2x_2 \leq 2 \quad \sim L_2$$

$$3x_1 + 2x_2 \leq 14 \quad \sim L_3$$

$$x_1, x_2 \geq 0$$



起作用约束:  $L_2$ 、 $L_3$

最优解  $(4, 1)$ ，最优值  $z_{\max}=13$

# 线性规划等式与不等式

$$\max z = 3x_1 + x_2$$

$$s.t. \quad -x_1 + x_2 \leq 2$$

$$x_1 - 2x_2 \leq 2$$

$$3x_1 + 2x_2 \leq 14$$

$$x_1, x_2 \geq 0$$

$$\max 3x_1 + x_2$$

$$s.t. \quad -x_1 + x_2 + x_3 = 2$$

$$x_1 - 2x_2 + x_4 = 2$$

$$3x_1 + 2x_2 + x_5 = 14$$

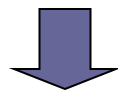
$$x_i \geq 0$$

最优解 (4,1) 对应(4,1,5,0,0)

L1和L3的交点:令  $x_3 = x_5 = 0$ ,

解出: (2, 4, 0, 8, 0), 原解 (2, 4)

# LP的约束和目标函数均为线性函数



## 2维

**可行域** 线段组成的凸多边形

**目标函数** 等值线为直线

**最优解** 凸多边形的某个顶点

## n维

超平面组成的凸多面体

等值线是超平面

凸多面体的某个顶点

## 求解LP的基本思想

从可行域的某一顶点开始，只需在有限多个顶点中一个一个找下去，一定能得到**最优解**。

算法：怎样从一点转到下一点，尽快找到最优解。



- LP任何一个基本可行解( $x_B, x_N$ ),  $x_N=0$ 的各分量有界, 且各分量的二进制字符串表达式的规模不超过  
 $2(m+n+mn+3+\log_2|P|)$ , 即基本可行解的规模可以被实例规模的多项式函数所控制, 其中 $P$ 为LP判定问题中所有非零参数的乘积。

$$x_B = B^{-1}b$$

$$1 \leq |\det(B)| \leq m! |P|,$$

$$\left| \sum_{i=1}^m b_i B_{ij} \right| \leq m! |P|.$$

$$B^{-1} = B^* / \det(B)$$

3的含义?

$$m! |P| \leq 2^{nm} |P| = 2^{nm} 2^{\log_2|P|}$$

一个分量（分子和分母）的规模：不超过 $2(3+mn+\log_2|P|)$

基本可行解的规模不超过： $2n(mn+m+n+3+\log_2|P|)$

验证算法计算时间： $O((n+n-1+1)(m+1)+n)$

取： $g(x) = x^2$  即可得到结论

n个乘, n-1个加,  
1个比较

# TSP问题属于NP

实例的规模 $l(I)$ 至少是 $n(n-1)$ 。对任何一个 $n$ 城市的TSP实例，它的解是 $(1,2,\dots,n)$ 的一个排列 $W=(i_1,i_2,\dots,i_n)$ ，该字符串规模不超过 $\sum_{i=1}^n (\lceil \log_2(i+1) \rceil) + 2n < 3n + n \log_2 n$ 。TSP任何一个解的规模不超过 $n(3 + \log_2 n)$ 。

验证是否满足

$$f(W) = \sum_{j=1}^n d_{i_j i_{j+1}} \leq z ?$$


算法有 $n-1$ 个加法和一个比较，算法的计算时间为 $n$ 。于是，定义中的多项式函数可选 $g(x)=x$ ，验证算法的计算时间和判定问题实例可行解的规模都满足定义的要求，故 $\text{TSP} \in \text{NP}$ 。

# 三精确覆盖属于NP

- 三精确覆盖:  $S=\{u_1, u_2, \dots, u_{3m}\}$ ,  $S_i$ 三元素集,  $i=1, 2, \dots, n$ .
- $S_i$ 等价表示:  $a_i=(a_{ij}) \in \{0, 1\}^{3m}$ , 当 $j$ 元素属于 $S_i$ 时,  $a_{ij}=1$ , 否则为 $a_{ij}=0$ .
- 结构: 一个 $3m^2$ 维的0-1字符串, 表示选出的 $m$ 个三元素集。

# 三精确覆盖属于NP

- 三精确覆盖问题任何一个实例的规模至少为 $n$ ，不妨假定 $n > m$ 。
- 三精确覆盖实例的一个解转化为一个字符串向量。精确三覆盖任何一个“是”实例的解可用规模为 $3m^2$ 个字符表示。
- 验证是否为“是”实例算法需 $3m(m-1)$ 个加法和 $3m$ 个比较，算法的计算时间为 $3m^2$ 。
- 多项式函数可选 $g(x) = x^2$ 。
- 三精确覆盖属于NP。

- 
- 简单多项式判定问题类**P**属于**NP**问题类
  - TSP问题也属于NP
  - NP不是非多项式问题集！！！！

# 多项式时间转换

- 思想：如果能够将A1问题化成A2问题，且可以用解决A2问题的方法解决A1问题。我们称这种方法为归类或归约。
- 给定问题A1和A2，若存在一个多项式函数 $g(x)$ 和一个对应关系（算法）T满足：
  - T将A1问题的一个实例 $I_1$ ，计算得到A2问题的一个实例 $I_2$ ，且 $I_2$ 为A2问题的一个“是”实例的充分必要条件是 $I_1$ 为A1问题的一个“是”实例。
  - T是一个多项式算法（关系），即 $I_2$ 的规模为 $O(g(l(I_1)))$ 。
- 则称A1问题多项式时间转换(transformation)为A2问题。

# 直观结论

- 若A1问题多项式时间转换为A2问题且  $A2 \in P$ ，则  $A1 \in P$ 。
  - 求解A1的多项式时间算法为：多项式时间转换算法T加上A2的多项式时间算法。
  - 由A2的实例得到的“是”或“否”可以回答A1实例的“是”与“否”。
- 若A1问题多项式时间转换为A2问题且A1是难的问题，则A2也难！（研究必备！！！！）

# Hamilton回路多项式时间转换为TSP判定问题。

$$d_{ij} = \begin{cases} 1, & \text{若Hamilton节点}(i,j)\text{之间有边相连,} \\ 2, & \text{其他。} \end{cases}$$

**TSP判定问题：**是否存在一个每城市过且仅过一次并回到起点的闭圈使得路长不超过 $n$ ？

Hamilton的一个实例 $\Rightarrow$ TSP的一个实例，其长度： $O(n^2)$ 。 $g(x)=x$ 。

Hamilton的一个“是”实例 $\Leftrightarrow$ TSP的一个“是”实例。如何证明充分性？



# 适定性问题多项式转换为整数线性规划问题

- 适定性问题为A1，对其任一实例，设计对应关系
  - 逻辑变量“真”对应“1”，“假”对应“0”，文字 $x$ 对应一个变量 $y$ ， $\bar{x}$ 对应 $1-y$ .
  - 句子 $C_j$ 对应

$$\sum_{x_i \in C_j} y_i + \sum_{x_i \in \overline{C_j}} (1 - y_i) \geq 1$$

■ 整数规划问题的一个实例，是否存在  $(y_1, y_2, \dots, y_n) \in \{0, 1\}^n$  使得下列方程成立？

$$\sum_{x_i \in C_j} y_i + \sum_{x_i \in \overline{C_j}} (1 - y_i) \geq 1, j = 1, 2, \dots, m.$$

怎么估的？每一个约束  $O(m \log n)$ !

- 适定性问题的任何一个实例规模：  
 $\geq \max(m, n)$ , 或  $mn$ .
- 转换实例规模:  $O(mn)$ , 多项式函数可选:  $x^2$  或  $x$ .
- 整数规划问题的“是”实例充要条件适定性问题的“是”实例

# 0-1背包问题判定问题简化过程

- 优化问题对应的判定问题：给定 $z$ ，集合  $\{x \in \{0,1\}^n \mid c^T x \geq z, a^T x \leq b\} \neq \emptyset$ ?
- 特别取 $z=b, a=c$ ，是判定问题的一个特殊情形。
- 判定问题：给定 $c \in \mathbb{Z}^n, b \in \mathbb{Z}$ 的非负整数（有理数），是否存在 $S \subseteq \{1,2,\dots,n\}$ 使得

$$\sum_{i \in S} c_i = b?$$

### 三精确覆盖问题多项式转换为0-1背包判定问题

0-1背包判定问题：对给定的整数 $c_j, j = 1, 2, \dots, n$ 和 $b$ ,

是否存在 $\{1, 2, \dots, n\}$ 的子集 $B$ , 使得 $\sum_{j \in B} c_j = b$ ?

变换：三精确覆盖问题中  $S_j$

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (n+1)^{i-1} \quad b = \sum_{j=1}^{3m} (n+1)^{j-1}$$

背包问题实例： $c_j, j=1, 2, \dots, n, b$

复杂性分析：三精确覆盖实例规模 $\geq n$ , 背包实例的规模估计： $n+1$ 个数， $c_j, j=1, 2, \dots, n, b$ ；最大数 $b$ 的二进制长度上界， $9m \log_2(n+1)$ 。

背包问题实例规模： $O(nm^2 \log_2 n), g(x)=x^4$

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (n+1)^{i-1} \qquad b = \sum_{j=1}^{3m} (n+1)^{j-1}$$

$S$  存在一个三精确覆盖  $\{S_{j_1}, S_{j_2}, \dots, S_{j_m}\}$  的充要条件为  $\sum_{i=1}^m c_{j_i} = b$ 。

充分性：（直观）假设存在  $T \subseteq \{1, 2, \dots, n\}$ ，使得  $\sum_{j \in T} c_j = b$ 。  $b$  的每个指数项系数为1。又因为  $T$  的总项数不超过  $n$ ，其同次幂项的系数和不超过  $n$ ，因而，  $n+1$  的每个次幂项系数恰好为1，得  $S_j, j \in T$  是  $S$  的一个三精确覆盖。

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (n+1)^{i-1} \quad b = \sum_{j=1}^{3m} (n+1)^{j-1}$$

观察：

$c_j$ 的每一项含**S**中**3**个互异元素，**T**中指标个数不超过**n**

归纳证明：从 $(n+1)^{3m-1}$ 高次幂到低次幂归纳。

低次幂数之和不可能升级。假设

$\sum_{j \in T} c_j$  中 $(n+1)$ 的不低于**j+2**次幂的系数是**1**，对**j+1**次幂，

小于等于**j**次幂的所有数之和不超过

$$n + n(n+1) + \cdots + n(n+1)^j = (n+1)^{j+1} - 1 < (n+1)^{j+1}$$

必要性明显。所以，三精确问题多项式时间转换为背包问题。

适定问题多项式转换为三精确覆盖问题。

# 其他的数学转换

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (3n+1)^{i-1} \quad b = \sum_{j=1}^{3m} (3n+1)^{j-1}$$

$S$  存在一个三精确覆盖  $\{S_{j_1}, S_{j_2}, \dots, S_{j_m}\}$  的充要条件为  $\sum_{i=1}^m c_{j_i} = b$ 。

充分性：假设存在  $T \subseteq \{1, 2, \dots, n\}$ ，使得  $\sum_{j \in T} c_j = b$ 。  $b$  的每个指数项系数

为1。因  $\sum_{j \in T} c_j$  中求和总项数不超过  $3n$ ，从  $b$  中  $(3n+1)$  的最高次幂开始归

纳得到每个次幂前的系数都为1。得  $S_j, j \in T$  是  $S$  的一个三精确覆盖。



# 多项式转换的进一步理解

- 多项式转换是将问题**A1**的所有实例通过多项式时间的计算对应到**A2**问题中的一部分实例，并不一定是**A2**中的所有实例。然后，通过**A2**的算法对这些来自**A1**对应的实例求解。如果求解**A2**映射后的一个实例为“是”，那么，**A1**中对应的实例为“是”，否则，**A1**中对应的实例为“否”。
- 若**A2**为多项式 $g(x)$ 可解，那么，**A1**也就多项式可解。
  - **A1**的求解算法是先进行多项式时间 $f(x)$ 转换，然后调用一次**A2**的算法即可。算法复杂性： $O(g(f(l(I))))$
- 如果**A1**算法将**A2**作为子程序多次调用？
  - 多项式归约的概念

# 多项式归约(reduction)

- 判定问题A1多项式时间归约为A2：存在一个A1的算法和多项式函数 $g(x)$ ，算法求解A1任何实例 $I$ 的过程中，将A2的算法作为子程序多次调用。如果将一次调用A2算法看成一个单位（1次基本计算量），则这个A1算法的计算量为 $O(g(l(I)))$ ，其中 $l(I)$ 为实例 $I$ 的规模。
- 调用A2算法时，需要将A1的一个实例计算到A2的一个实例，才可能调用A2的算法。这些计算量全部记录在A1算法的计算量中。

若 A2 存在多项式时间算法，多项式控制函数为  $g_2(x)$ ，存在 A1 问题的算法，其计算时间  $O(g(l(I))g_2(g(l(I))))$ ，是 A1 的一个多项式时间算法。

## ■ A1问题

对给定 $A_{m \times n}$ 和 $b_{m \times 1}$ , 集合 $\{x | Ax \leq b \text{ 且至少有一个等式成立}\}$ 是否不为空集?

## ■ A2问题

对给定 $C_{k \times l}$ 和 $d$ , 集合 $\{y | Cy \leq d\}$ 是否不为空集?

## ■ A1算法 $m$ 次调用A2。

$\{x | Ax \leq b, -(A)_i x \leq -(b)_i\}$ 是否不为空集?

## ■ A1多项式归约为A2。

# NP完全和NP难

- **Cook**在1971年给出并证明了有一类问题具有下述性质：(1)这类问题中任何一个问题至今未找到多项式时间算法；(2)如果这类问题中的一个问题存在多项式时间算法，那么这类问题都有多项式时间的算法，这类问题中的每一个问题称为**NP完全**（**NP-Complete**），这个问题的集合简记**NPC**。
- 如果判定问题**A**满足： $A \in NP$ 且**NP**中的任何一个问题都可在多项式时间内归约为**A**，则称**A**为**NP完全**。若**NP**中的任何一个问题都可在多项式时间归约为判定问题**A**，则称**A**为**NP难**（**NP-hard**）。简记**NP难**问题的集合为**NPH**，显然有**NPC**  $\subseteq$  **NPH**。

- **NP完全和NP难问题的区别是NP难问题无须判断A是否属于NP。验证一个问题A是否为NPC的关键有两点，一是NP中的任何一个问题是否可在多项式时间内归约为A，其次，是否存在一个字符串，其规模为实例规模的多项式函数，以及是否存在一个多项式时间的验证算法。**

# NP-C或NP-H问题分类的基本程序

- 遇到的新问题
  - 来源于实际。
- 查找文献发现类似的问题（已知问题）是NPC或NPH
- 可以多项式归约为新问题，则知新问题是NP难。
- 当可以验证新问题是属于NP时，则该问题属于NPC。

# 基本结论——假设

- 适定性问题是NP完全。
- Hamilton回路是NP完全。
  - 参考：加里 M R，约翰逊 D S(张立昂等译). 计算机和难解性：NP-C性理论导论. 科学出版社，1987
- 整数线性规划的判定问题是NP完全
  - 适定问题多项式转换为整数线性规划的判定问题
  - 整数线性规划的判定问题属于NP,证明见  
Papadimitriou C H, Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. New Jersey: Prentice-Hall INC, 1982, p.320, Theorem13.4 。

## ■ 三精确覆盖是NP完全

- 三精确覆盖属于NP，前面已证“结构”存在。
- 适定性问题多项式转换三精确覆盖（书pp.32-33）

## ■ 背包判定问题是NP完全

- 三精确覆盖多项式转换背包问题
- 解可用n个0-1分量表示，属于NP

集合划分(partition)问题：给定(正)整数 $c_1, c_2, \dots, c_n$ ，是否存在 $\{1, 2, \dots, n\}$ 的一个子集 $S$ ，使得 $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$ ？



## ■ 属于NP

□ 1表示在集合S，0表示在余集中

## ■ 0-1背包判定问题多项式转换为集合划分问题

0-1 背包判定问题的任何一个实例， $c_1, c_2, \dots, c_n, b$ ，构造集合划分问题的实例

例  $c_1, c_2, \dots, c_n, c_{n+1} = M, c_{n+2} = 2b$ ，其中， $M = \sum_{j=1}^n c_j > b$ 。

存在 $(1, 2, \dots, n)$ 的子集 S，使得  $\sum_{j \in S} c_j = b$  的充分必要条件

是存在 $\{1, 2, \dots, n+2\}$ 的一个子集 S'使得  $\sum_{j \in S'} c_j = \sum_{j \notin S'} c_j$

必要性：令  $S' = S \cup \{n+2\}$

充分性：n+1, n+2元素不可能在同一个S'中。  $\sum_{j \in S} c_j = b$

# 充分性证明中的关键点

- 已将 $n+2$ 物品平衡分成两个部分。
- 第 $n+1, n+2$ 两个物品不能同时在一边。
- $c_{n+1}=M, c_{n+2}=2b$ 构造具有技术性。

$$\sum_{j \in S} c_j + M = \sum_{j \notin S} c_j + 2b \Rightarrow \sum_{j \in S} c_j + M = M - \sum_{j \in S} c_j + 2b$$

$$\sum_{j \in S} c_j = b, S \subseteq N = \{1, 2, \dots, n\}$$

# 最小差异问题——NP难

给定正整数 $c_1, c_2, \dots, c_n$ ，求 $\{1, 2, \dots, n\}$ 的一个子集 $S$ ，使得 $|\sum_{j \in S} c_j - \sum_{j \notin S} c_j|$ 最小。

- 寻找多项式转换问题
  - 集合划分问题
- 最小差异问题的判定问题
  - 给定常数0，是否存在 $\{1, 2, \dots, n\}$ 的一个子集 $S$ ，使得

$$|\sum_{j \in S} c_j - \sum_{j \notin S} c_j| \leq 0?$$

# 另一种变形问题NPC

给定正整数 $c_1, c_2, \dots, c_n$ ,  $\delta > 0$ , 是否存在 $\{1, 2, \dots, n\}$ 的一个子集 $S$ , 使得 $|\sum_{j \in S} c_j - \sum_{j \notin S} c_j| \leq \delta$ ?

集合划分(partition)问题: 给定正整数 $c_1, c_2, \dots, c_n$ , 是否存在 $\{1, 2, \dots, n\}$ 的一个子集 $S$ , 使得 $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$ ?

- 集合划分问题来多项式时间转换: 给定正整数 $K$ 使得 $K > \delta$ , 得到最小差异变形问题的  $\delta$ ,  $c_i' = K c_i, i=1, 2, \dots, n$ .

## ■ 装箱问题是NPC

给定 $K$ 及 $n$ 个物品，尺寸为 $0 < b_1, b_2, \dots, b_n \leq 1$ ，是否能在 $K$ 个尺寸为1的箱子中装入这 $n$ 个物品？

## ■ 属于NP？

□  $(1, 2, \dots, n)$ 的一个排列。

□  $x_{ik}, i=1, 2, \dots, n; k=1, 2, \dots, K$ .

□ 结构是否为实例的多项式长度？变尺寸装箱问题会出现问题。

- 由谁来转换（归约）？

- Partition问题: 给定 $n$ 个正整数  $c_1, c_2, \dots, c_n$ , 是否存在一个集合 $S$ , 使得

$$\sum_{j \in S} c_j = \sum_{j \notin S} c_j$$

- 装箱问题（实例）：计算  $b_i = \frac{2c_i}{\sum_{j=1}^n c_j}, i = 1, 2, \dots, n$

- 去掉包含 $b_i > 1$ 的实例（简单实例）。是否能在两个箱子（即 $K=2$ ）内装完所有物品？

## ■ 约束单机排序问题：NP难

$$\min T$$

$$s.t. \sum_{t=1}^T x_{it} = 1, i = 1, 2, \dots, n,$$

$$\sum_{i=1}^n d_i x_{it} \leq c_t, t = 1, 2, \dots, T,$$

$$x_{it} \in \{0, 1\}, i = 1, 2, \dots, n; t = 1, 2, \dots, T,$$

# NP难——证明1

- 对于装箱问题的判定问题：给定 $K$ 及物品  $0 < b_1, b_2, \dots, b_n \leq 1$ ，是否存在一个装箱方法使  $n$  个物品装入  $K$  个箱子中？（ $1 \leq K \leq n$ ）
- 多项式时间转换为：给定时段 $K$ 及 $n$ 个产品  $0 < d_i = b_i, i=1, 2, \dots, n$ ，每个时段加工能力 $c_t=1$ ，是否存在一个加工方法使 $n$ 个产品在 $K$ 时段内完成？



# NP难——证明2

- 多项式时间转换问题partition问题：给定 $n$ 个正整数  $a_1, a_2, \dots, a_n$ , 是否存在一个集合 $S$ , 使得

$$\sum_{j \in S} a_j = \sum_{j \notin S} a_j$$

# 得到约束单机排序问题判定问题的一个子集

■ 令 
$$d_i = \frac{2a_i}{\sum_{j=1}^n a_j}, i = 1, 2, \dots, n$$

- 去掉包含  $d_i > 1$  的实例（简单实例）。是否能在两个时段（即  $K=2$ ）内加工完所有物品？
- Partition问题多项式时间转换成约束单机排序问题判定问题。
- 约束单机排序问题判定问题：NP难。

# 约束单机排序问题NP完全？

- 约束单机排序问题判定问题：NP完全
- 判定问题：给定 $K$ 及 $0 < d_1, d_2, \dots, d_n, c_1, c_2, \dots, c_K$ ，是否存在一个调度方法使 $n$ 个产品在 $K$ 时段内加工完成？
- 解的结构： $(1, 2, \dots, n)$ 的一个全排列 $(i_1, i_2, \dots, i_n)$ 。与TSP的解结构相同，多项式长度。
- 验证算法：从 $t=1$ 至 $K$ 按 $d_{i_1}, d_{i_2}, \dots, d_{i_n}$ 顺序逐一满足加工能力约束加工，不满足则用一个新时段。
- 解的结构：长度为 $K$ 的0-1字符串？（错误！多项式时间长度？）

## ■ TSP判定问题是NP完全

- TSP的判定问题属于NP

- 由谁来多项式转换？

- Hamilton回路问题

$$d_{ij} = \begin{cases} 1, & \text{若Hamilton节点}(i,j)\text{之间有边相连,} \\ 2, & \text{其他。} \end{cases}$$

TSP的一个判定实例为：是否存在一个TSP回路，使得它的距离不超过 $n$ ？

# 练习——证明

## ■ 背包问题为NP难或TSP问题为NP难。

- 先将优化问题写成判定问题。
- 再通过已知的NP完全问题多项式时间转换为对应的判定问题。
- 得到结论。

## ■ 背包或TSP判定问题为NP完全。

- 先证明该判定问题属于NP。
- 再通过已知的NP完全问题多项式时间转换为对应的判定问题。
- 得到结论。

# 人们猜想

- NPC中的任何一个问题没有多项式时间算法，即 $P \cap NPC = \emptyset$ 。
- $P \neq NP$  (21世纪百万美金题目)
- 论文中常用的一句话：不存在...，除非 $P=NP$  (unless  $P=NP$ ).

# 优化问题与判定问题关系

- 当存在一个优化问题的判定问题为**NP**难或**NP**完全，则优化问题为**NP**难。
- 当优化问题的判定问题为多项式时间问题时，优化问题是否是多项式时间问题？
  - 判定问题只给出实例的“是”和“否”。
  - 优化问题给出精确解（最优解）。
  - 优化与判定问题之间的关系？

# 线性规划问题与其判定问题的关系

线性规划问题 (LP):

$$\min c^T x$$

$$s.t. Ax = b$$

$$x \geq 0$$

判定问题 (DP):

给定整数 $z$ , 是否有

$$\{x \mid c^T x \leq z, Ax \geq b, Ax \leq b, x \geq 0\} \neq \emptyset?$$

**LP**存在多项式时间最优算法的充分必要条件是**DP**为多项式时间问题。

通过二分法的逼近求出最优值。二分法迭代的次数是实例规模的多项式函数。



- 假设LP中的所有系数为整数。
- LP任何一个基本可行解( $x_B, x_N$ ),  $x_N=0$ 的各分量

$$x_B = B^{-1}b \qquad B^{-1} = B^* / \det(B)$$

$$1 \leq |\det(B)| \leq m! |P|,$$

$$\left| \sum_{i=1}^m b_i B_{ij} \right| \leq m! |P|. \qquad m! |P| \leq 2^{nm} |P| = 2^{mn} 2^{\log_2 |P|}$$

- 一个基本可行解每个分量的分母都写成 $\det(B)$ , 分母上界为

$$m! |P| \leq 2^{nm} |P| = 2^{mn} 2^{\log_2 |P|} \leq 2^{mm+m+n+3+\log_2 |P|} = 2^L$$

假设  $x^1$  和  $x^2$  是LP的两个基本可行解，并且对某一个整数K满足

$$K2^{-2L} < c^T x^1, c^T x^2 \leq (K+1)2^{-2L},$$

其中，  $L = mn + m + n + 3 + \log_2 |P|$ ，  $P$  为  $A, b, c$  中非零分量的

乘积， 则有  $c^T x^1 = c^T x^2$ 。

证明(反证)： 每一个基本可行解分母的绝对值不超过  $2^L$ 。 若

$c^T x^1 \neq c^T x^2$ ， 有  $|c^T x^1 - c^T x^2| \geq 2^{-2L}$ ， 此时与已知条件矛盾。

$$\left| \sum_{j=1}^n c_j x_j \right| \leq \sum_{j=1}^n |c_j x_j| \leq \sum_{j=1}^n 2^{\log_2(|c_j|) + L_1} \leq 2^{\log_2 n + \log_2 |P_2| + L_1} \leq 2^{2L}$$

$L_1 = mn + m + n + 3 + \log_2 |P_1|$ ,  $P_1$  为 A 和 b 中非零数的乘积。

$P_2$  为  $c_j$  中非零数的乘积。

考虑区间  $[-2^{4L}, 2^{4L}]$  内的整数点  $K$ , 分下面三种情况讨论。

情况 1: 当  $K = 2^{4L}$ , 取  $z = K 2^{-2L} = 2^{2L}$  时, 若 DP 是一个“否”实例, 则 LP 无可行解。

情况 2: 当  $K = -2^{4L}$ , 取  $z = K 2^{-2L} = -2^{2L}$  时, 若 DP 还是一个“是”实例, 则 LP 无下界。

情况3： 以上两种情况不满足时，用二分算法：

二分算法

$$\text{STEP1 } u = -2^{4L}, v = 2^{4L}; \quad K = \frac{u+v}{2} = 0;$$

STEP2 当  $v - u \leq 1$  时停止，记录此时的  $(u, v, K)$  为  $(u^*, v^*, K^*)$ ；否则取  $z = K2^{-2L}$ ，

若 DP 为“是”实例，则令  $u := u, v := K$ ；若 DP 为“否”实例，则令  $u := K, v := v$ ；

$$K = \frac{v+u}{2}, \text{ 重复 STEP2.}$$

- 第一和第二种情况说明在第三种情况时一定有有界的可行解。
- 初始 $u=-2^{4L}$  和 $v=2^{4L}$ 的选择保证二分算法每步的 $K$ 一定为整数。
- 二分算法结束时一定有 $v^*-u^*\leq 1$ ，当 $z=v^*2^{-2L}$ 时，判定问题DP为“是”实例，而当 $z=u^*2^{-2L}$ 时，判定问题DP为“否”实例。
- 二分算法的迭代次数为 $\log_2 2^{4L+1}=4L+1$ 。
- 最优目标值 $z^*$ 在区间  $(u^*2^{-2L}, v^*2^{-2L}]$ 内，仍然无法确定最优解。

- 确定最优解的方法是继续求解判定问题

$$D_j = \{x \mid c^T x \leq v^* 2^{-2L}, Ax \geq b, Ax \leq b, x \geq 0; x_j \leq 0, x_i \leq 0, i \in S(j)\} \neq \Phi?$$

$$S(1) = \Phi \quad S(j) = \{i \mid i < j, D_i \neq \Phi\}$$

- 迭代 $n$ 次后，得到 $S(n+1)$ 。
- 由线性规划的理论，以外的变量为基变量(至多为 $m$ 个)，所以可以得到基矩阵。
- 线性方程组的理论可以求出唯一解。
- 这些计算也是实例规模的多项式时间的。
- 只要LP的判定问题有多项式时间算法，则LP优化问题也有多项式时间算法。

# 小结

## ■ NPC,NPH证明的常见方法

### □ Restriction

- Hamilton circuit to TSP
- SAT to IP

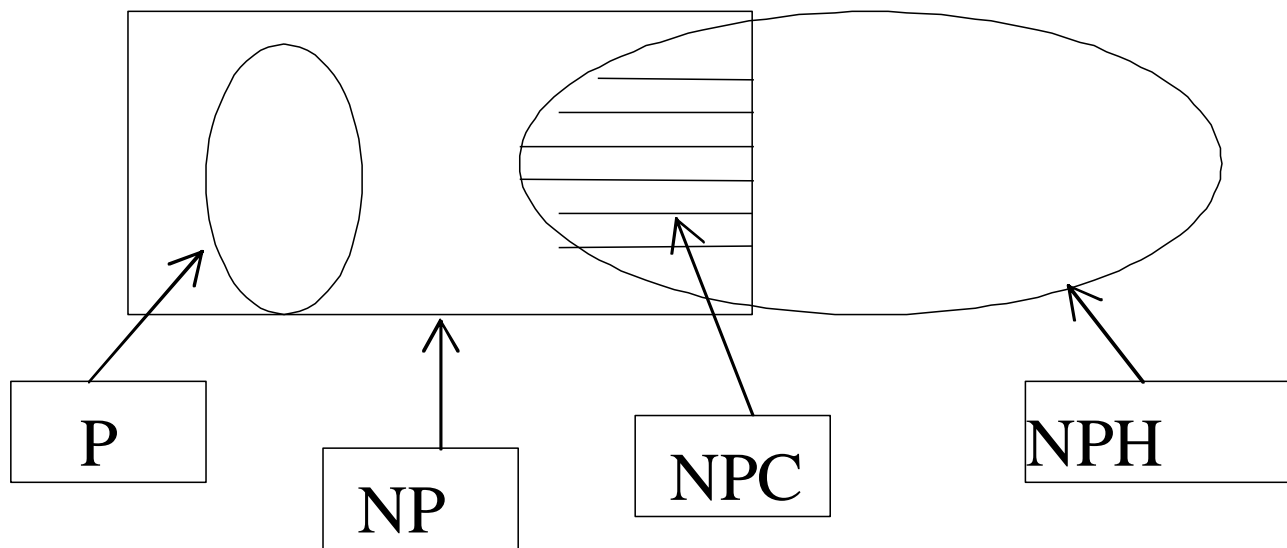
### □ Local replacement

- 3EC to Knapsack
- Partition to bin packing

### □ Component design

- Knapsack to Partition
- SAT to 3EC

- 判定问题是**NP**完全或**NP**难，则知优化问题的难度不低于判定问题，此时，称组合最优化问题为**NP**难。





## 1.6 多项式时间逼近格式

- Polynomial-time approximation scheme, 简记为PTAS。
- 在NPC中进一步分类
  - 简单问题（伪多项式或PTAS）
  - 难问题（strongly NP-C, strongly NP-hard)

# 0-1背包特殊判定问题的动态规划求解

0-1 背包问题：对给定的整数  $c_j, j = 1, 2, \dots, n$  和  $b$ ,

是否存在  $\{1, 2, \dots, n\}$  的子集  $B$ , 使得  $\sum_{j \in B} c_j = b$ ?

令  $d_0^0 = d_1^0 = \dots = d_b^0 = 0$ ，从  $j=1$  到  $n$ （外循环），对每一个  $j$  分别从  $v=1$  到  $b$ （内循环）计算

$$d_v^j = \begin{cases} \max\{d_v^{j-1}, d_{v-c_j}^{j-1} + c_j\}, & v \geq c_j \\ d_v^{j-1}, & v < c_j. \end{cases}$$

含义：以容量  $v$  只考虑装前  $j$  个物品的最佳值。

$$c_1 = 3, c_2 = 2, c_3 = 1, c_4 = 4, b = 7$$

$$j=0: \quad d_0 = d_1 = \cdots = d_7 = 0 \quad d_v^j = \begin{cases} \max\{d_v^{j-1}, d_{v-c_j}^{j-1} + c_j\}, v \geq c_j \\ d_v^{j-1}, v < c_j. \end{cases}$$

$$j=1: \quad d_1 = d_2 = 0, d_3 = d_4 = \cdots = d_7 = 3,$$

$$j=2: \quad d_1 = 0, d_2 = 2, d_3 = d_4 = 3, d_5 = d_6 = d_7 = 5,$$

$$j=3: \quad d_1 = 1, d_2 = 2, d_3 = 3, d_4 = 4, d_5 = 5, d_6 = d_7 = 6,$$

$$j=4: \quad d_1 = 1, d_2 = 2, d_3 = 3, d_4 = 4, d_5 = 5, d_6 = 6, d_7 = 7。$$

由此可以看出，这是一个“是”实例。

- 0-1背包问题的实例为“是”的充分必要条件为  $d_b^n = b$ 。

可以归纳证明  $d_v^j \leq v(\forall j, v)$ ，且  $d_v^j(\forall v)$  关于  $j$  是非减的。

## 必要性

存在  $S = \{j_1, j_2, \dots, j_k\}$  使  $\sum_{j \in S} c_j = b$ ，不妨设  $1 \leq j_1 < j_2 < \dots < j_k \leq n$ 。

$$d_{c_{j_1}}^{j_1} = \max\{d_{c_{j_1}}^{j_1-1}, d_0^{j_1-1} + c_{j_1}\} \geq c_{j_1},$$

$$d_{c_{j_1}+c_{j_2}}^{j_2} = \max\{d_{c_{j_1}+c_{j_2}}^{j_2-1}, d_{c_{j_1}}^{j_2-1} + c_{j_2}\} \geq d_{c_{j_1}}^{j_2-1} + c_{j_2} \geq d_{c_{j_1}}^{j_1} + c_{j_2} \geq c_{j_1} + c_{j_2},$$

...

$$d_b^{j_k} = d_{c_{j_1}+c_{j_2}+\dots+c_{j_k}}^{j_k} \geq c_{j_1} + c_{j_2} + \dots + c_{j_k} = \sum_{j \in S} c_j = b.$$

## 充分性

已有的  $nb$  个数  $\{d_v^j \mid j=1,2,\dots,n, v=1,2,\dots,b\}$ ，设  $j_k$  为

第一个满足  $d_b^{j_k} = b$  的数，则必有  $d_{b-c_{j_k}}^{j_k-1} = b - c_{j_k}$ 。

可得到  $j_{k-1} > j_{k-2} > \dots > j_1$ ，满足

$$d_{b-c_{j_k}-\dots-c_{j_{i+1}}}^{j_i} = b - c_{j_k} - \dots - c_{j_{i+1}}, d_{b-c_{j_k}-\dots-c_{j_{i+1}}}^{j_i-1} < b - c_{j_k} - \dots - c_{j_{i+1}}, i = k-1, k-2, \dots, 1$$

$$d_{b-c_{j_k}-\dots-c_{j_i}}^{j_i-1} = b - c_{j_k} - \dots - c_{j_i}, i = k-1, k-2, \dots, 1$$

$$d_{b-c_{j_k}-\dots-c_{j_1}}^{j_1-1} = b - c_{j_k} - \dots - c_{j_1} = 0$$

$\{j_1, j_2, \dots, j_k\}$  “是”实例的一个解。

- 参数为 $c_j, j=1,2,\dots,n$ 和 $b$ 的0-1背包问题的动态规划算法计算时间不低于 $nb$ 。

$$\log_2 b + 2 \leq L = \lceil \log_2(b+1) \rceil + 2 \leq \log_2 b + 3$$

- 算法复杂性下界是 $nb \geq 1/8(n2^L)$ , 无法避开指数形式 $2^L$ 。
- 定义:  $I$ 为一个问题的实例,  $I$ 中出现的最大数记为 $\#(I)$ 。
- 定义: 对于问题的一个算法 $A$ , 如果存在2元多项式函数 $g_1(x,y)$ , 使得此算法求解任何实例 $I$ 的计算复杂性 $C_A(I) = O(g_1(l(I), \#(I)))$ , 则称这个算法是伪多项式的(pseudo-polynomial), 实例 $I$ 的规模  $l(I)$ 。

- 若存在一个多项式 $g(x)$ ，当一个问题的所有满足条件 $\#(I)=O(g(l(I)))$ 的实例组成的子问题是NP完全时，则称这个问题是强NP完全(strongly NP-complete)。
  - 限定 $z=n$ ，城市间的距离为1或2的TSP仍然为NP完全,所以TSP是强NP完全。
  - 适定性问题强NP完全（所有系数0，1）。
  - 整数规划判定问题强NP完全（考虑适定性转换后的线性整数规划判定问题）。
  - 3精确覆盖强NP完全（系数0，1）。
- 装箱判定问题，集合划分问题非强NP完全。

Fernandez W de la vega, Lueker GS. Bin packing can be solved within  $1 + \epsilon$  in linear time. Combinatorics 1981; 1:349-355.



■ 除非 $P=NP$ ，强NP完全问题不存在伪多项式时间算法。

□ 证明：假设一个强NP完全问题存在一个伪多项式时间算法，记算法计算量为 $O(g(l(I), \#(I)))$ 。对任意一个多项式 $f(x)$ ，当限定问题为 $\#(I)=O(f(l(I)))$ 的子问题时，该算法以一个多项式时间 $O(g(l(I), f(l(I))))$ 解决了子问题。除非 $P=NP$ ，这是不可能的。

# 多项式转换（归约）与强NP完全

■ 强NP完全A问题多项式时间转换（归约）B问题，B问题是否为强NP难（完全）？


□ 典型例子：3精确覆盖多项式时间转换背包判定问题。

□ 三精确覆盖问题中  $S_j$

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (n+1)^{i-1} \quad b = \sum_{j=1}^{3m} (n+1)^{j-1}$$

背包问题实例：  $C_j, j=1,2,\dots,n,b$

■ 最大数  $b$  不小于  $(n+1)^{3m-1}$ ，无法用三精确覆盖实例长度多项式函数控制！ 特别注意：  $b$  的存储长度和  $b$  数大小的关系！



# 优化0-1背包问题的 伪多项式时间算法

Step 0: 设  $S_0 = \{(\emptyset, 0, 0)\}$ .

Step 1: 从  $j = 1, 2, \dots, n$  分别计算

Step1.1  $S_j = \{(\emptyset, 0, 0)\}$ ;

Step1.2  $S_j = S_{j-1} \cup \{(S \cup \{j\}, z + c_j, a + a_j) \mid (S, z, a) \in S_{j-1} \text{ 且 } a + a_j \leq b\}$ ;

Step1.3 在  $S_j$  中按目标值  $z$  检查: 若有  $(S, z, a'), (T, z, a'') \in S_j$ , 且  $a' \leq a''$ , 则

从  $S_j$  中删除  $(T, z, a'')$ 。

Step 2: 找出  $S_n$  中具有最大目标值  $z^*$  的  $(S, z^*, a)$ ,  $S$  和  $z^*$  分别为一个最优解和最优目标值。

$$z^* = \max \sum_{i=1}^n c_i x_i$$

$$s.t. \sum_{i=1}^n a_i x_i \leq b$$

$$x_i \in \{0, 1\}, \quad j = 1, \dots, n.$$

只考虑前  $j$  个物品装包时的所有可行解，用集合  $S_j$  表示这些可行解。从 Step1.2 可以看出  $S_{j-1}$  自然是  $S_j$  的一部分，再考虑  $S_{j-1}$  中的元素（物品）同物品  $j$  结合后是否为可行解？如果仅有 Step1.2 而没有 Step1.3 的计算，则  $S_j$  中元素的个数可能为  $O(2^j)$ ，则计算出  $S_n$  的复杂性为  $O(2^n)$ 。

考虑 Step1.3 后，每次计算可将  $z$  从小到大排列，顺序检查，删除满足条件的一些可行解。于是  $S_j$  中的元素个数不超过

$S_j$  中目标值的最大数，也就不超过  $z^*$ 。

综合考虑 Step1.2 和 Step1.3, 对每一个固定的  $j$ , Step1.2 对每一个  $z$  的计算或生成、记忆  $S_j$  需要一个比较  $a+a_j \leq b$ , 两个计算  $z=z+c_j$  和  $a=a+a_j$  的计算量。同时, 由 Step1.3, 每一个  $z$  只对应  $S_j$  中一个元素。当 Step1.2 计算得到  $S_j$  中所有元素  $(S, z, a)$  后, 比较  $z$  可能对应的其他  $(S', z, a')$ , 最多有两个重复。因此, Step1.2 和 Step1.3 对一个  $j$  的计算量为  $O(z^*)$ 。A 算法的总计算复杂性为  $O(nz^*)$ 。(写法不严格,  $z^*$  需要计算得到, 所以  $O(n^2 \max \{c_i\})$ ).

# 计算0-1背包问题实例

$$a_1 = 1, a_2 = 1, a_3 = 4, a_4 = 3, c_1 = 20, c_2 = 10, c_3 = 31, c_4 = 20, b = 5$$

$$S_0 = \{(\emptyset, 0, 0)\} \quad j = 1, S_1 = \{(\emptyset, 0, 0), (1, 20, 1)\}$$

$$j = 2, S_2 = \{(\emptyset, 0, 0), (1, 20, 1), (2, 10, 1), (\{1, 2\}, 30, 2)\}$$

$$j = 3, S_3 = \left\{ (\emptyset, 0, 0), (1, 20, 1), (2, 10, 1), (\{1, 2\}, 30, 2), (3, 31, 4), \right. \\ \left. (\{1, 3\}, 51, 5), (\{2, 3\}, 41, 5) \right\}$$

$$j = 4, S_4 = \left\{ (\emptyset, 0, 0), (1, 20, 1), (2, 10, 1), (\{1, 2\}, 30, 2), (3, 31, 4), \right. \\ \left. (\{1, 3\}, 51, 5), (\{2, 3\}, 41, 5), (\{1, 4\}, 40, 4), (\{1, 2, 4\}, 50, 5) \right\}$$

$\{3, 4\}$ ,  $\{1, 3, 4\}$ ,  $\{2, 3, 4\}$ 的组合因超出包容量被删除,  $(4, 20, 3)$ 和 $(\{2, 4\}, 30, 4)$ 因同目标值相同而占用包体积过大被删除。最终得到最优解为 $S=\{1, 3\}$ , 最优值为51。

$$a_1 = 1, a_2 = 1, a_3 = 4, a_4 = 3, c_1 = 20, c_2 = 10, c_3 = 31, c_4 = 20, b = 5$$

$$a_1 = 1, a_2 = 1, a_3 = 4, a_4 = 3, c_1' = 20, c_2' = 10, c_3' = 30, c_4' = 20, b = 5$$

$$j = 4, S_4 = \left\{ (\emptyset, 0, 0), (2, 10, 1), (1, 20, 1), (\{1, 2\}, 30, 2), (3, 31, 4), \right. \\ \left. (\{1, 4\}, 40, 4), (\{2, 3\}, 41, 5), (\{1, 2, 4\}, 50, 5), (\{1, 3\}, 51, 5) \right\}$$

$$j = 4, S_4' = \{(\emptyset, 0, 0), (2, 10, 1), (1, 20, 1), (\{1, 2\}, 30, 2), (\{1, 4\}, 40, 4), (\{1, 3\}, 50, 5)\}$$

最优目标值为50。

后一种情况存储量较前一种情况少，但两个问题得到的目标值的相对偏差约为百分之二。

$$a_1 = 1, a_2 = 1, a_3 = 4, a_4 = 3, c_1' = 20, c_2' = 10, c_3' = 30, c_4' = 20, b = 5$$

$$\bar{c}_1 = 2, \bar{c}_2 = 1, \bar{c}_3 = 3, \bar{c}_4 = 2$$



$$\begin{array}{lll}
 z^* = \max \sum_{i=1}^n c_i x_i & \xrightarrow{\text{c}_j \text{ 用一个不超过它且后 } t \text{ 位为 } 0 \text{ 的最大整数 } c_j' \text{ 替代}} & z_1^* = \max \sum_{i=1}^n c_i' x_i \\
 s.t. \sum_{i=1}^n a_i x_i \leq b & & s.t. \sum_{i=1}^n a_i x_i \leq b \\
 x_i \in \{0,1\}, j = 1, \dots, n. & & x_i \in \{0,1\}, j = 1, \dots, n.
 \end{array}$$

$$z^* = \sum_{j \in S} c_j \geq \sum_{j \in S'} c_j \geq \sum_{j \in S'} c_j' = z_1^* \geq \sum_{j \in S} c_j' \geq \sum_{j \in S} (c_j - 10^t) \geq \sum_{j \in S} c_j - 10^t n$$

$$z^* - z_1^* \leq 10^t n \qquad \frac{z^* - z_1^*}{z^*} \leq \frac{10^t n}{\max \{c_j \mid j = 1, 2, \dots, n\}}$$

## 01KS-PTAS算法:

对给定的  $\varepsilon > 0$

(1) 01KS的实例满足  $\frac{n}{\max\{c_j \mid j = 1, 2, \dots, n\}} \leq \varepsilon$

则取 
$$t = \left\lfloor \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{n} \right\rfloor$$

$c_j$  用一个不超过它且后  $t$  位为 0 的最大整数  $c_j'$  替代, 求  $z_1^*$

$$\frac{z^* - z_1^*}{z^*} \leq \frac{10^t n}{\max\{c_j \mid j = 1, 2, \dots, n\}}$$

(2) 实例满足  $\frac{n}{\max\{c_j \mid j = 1, 2, \dots, n\}} > \varepsilon$  直接求解  $z^*$

$$\frac{z^* - z_1^*}{z^*} \leq \frac{10^t n}{\max\{c_j \mid j = 1, 2, \dots, n\}}$$

$$\frac{10^t n}{\max\{c_j \mid j = 1, 2, \dots, n\}} \leq \varepsilon$$

$$\frac{10^{t+1} n}{\max\{c_j \mid j = 1, 2, \dots, n\}} \geq \varepsilon$$

$$\begin{aligned} \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{10n} &= \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{n} - 1 \\ &\leq t \leq \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{n}, \end{aligned}$$

01KS-PTAS 算法的最终解目标值  $z_2^*$  满足:  $\frac{z^* - z_2^*}{z^*} \leq \varepsilon$  且算法的计算复杂性为  $O(n^3 \frac{1}{\varepsilon})$ 。

## 第一种情况

$$\log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{10n} = \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{n} - 1$$

$$\leq t \leq \log_{10} \frac{\max\{c_j \mid j = 1, 2, \dots, n\} \varepsilon}{n},$$

$$O(n^2 \max\{c_j \mid j = 1, 2, \dots, n\} * 10^{-t}) = O(n^3 \frac{1}{\varepsilon})$$

第二种情况:  $O(nz^*) = O(n^2 \max\{c_j\}) = O(n^3/\varepsilon)$

**定义** 对给定的  $\varepsilon > 0$ , 若一个优化问题存在一个算法满足: 它是近似算法; 且将  $\varepsilon$  看成常数时, 算法复杂性对任何一个实例, 是实例输入规模的多项式函数。则称这个算法为该问题的多项式时间逼近格式, 简称**PTAS**。

# ■ 连续优化问题的复杂性概念

## □ 模型

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0, \\ & x \in \mathbb{R}^n, \end{aligned}$$

□ 系数假设：实数


□ 黑箱假设：**cos(x)**, **sqrt(x)**和**df(x)/dx**分别计一次运算.

□ 实例的输入两部分：输入计算精度 $\varepsilon > 0$ ；实例的维数 $l(I)$ : 变量的个数和系数的个数.

- 基本运算的计量：两个实数间的加、减、乘、除、比较、读或写分别看成一次运算；一些特殊函数的运算以黑箱的形式看成一次运算。

## ■ 可计算(computable)或可求解 (solvable) 问题.

- 存在一个算法A和一个二元多项式函数 $g(x,y)$ , 对任意给定的 $\varepsilon > 0$ , 记算法A求解到 $x_A(I)$ , 目标值为 $v_A(I)$ 。
- $|v_A(I) - v_{\text{opt}}(I)| \leq \varepsilon, \forall I$ , (绝对误差!!!)
- 求解到 $x_A(I)$ 和验证 $x_A(I)$ 落在可行解区域或距可行解区域边沿距离不超过 $\varepsilon$ 的计算量为
- $O(g(l(I), \log_2(1/\varepsilon))), \forall I$

- 
- 特别注意：连续优化的computable与组合优化PTAS算法的差异。

# 第一章小结

- **NP完全或NP难问题**：必须寻求启发式算法
- 最坏情况分析适用于比较简单的组合最优化问题的启发式算法，它需要较深厚的数学基础。
- 概率分析的方法，同样因为具有较深的数学理论基础和较高的研究难度，使得这一方法难以推广使用。



- 大规模数值计算进行评价的方法简单、易行、易于理解和有较好的说服力，因此得到研究和实际应用者的广泛使用。
- 启发式算法是一对矛盾的策略中的平衡：一个策略是集中(intensification)，它的主要目标是集中搜索，求得局部最优解；另一个策略是扩散(diversification)，它的主要目标是扩大搜索区域，以达到全局最优。