组合优化理论

主讲教师：陈安龙

自我介绍

- 姓　　名：陈安龙
- 研究方向：机器学习与数据挖掘
- 电　　话：13688434459
- 课程QQ群：1097630363
- E-Mail: chenanlong@foxmail.com

群名称：组合优化理论课程群-2020
群　号：1097630363

课程简介

- 组合优化是研究从离散问题的所有可能方案中选择最合理的一种方案，以达到最佳目标的科学.
- 达到最佳目标的方案是最优方案，寻找最优方案的方法-----最优化方法(算法)
- 这些方法的数学理论即为组合最优化理论.
- 是运筹学的方法论之一，是其重要组成部分.

最优化首先是一种理念，其次才是一种方法.

主要包括三部分：

- 模型
- 理论
- 算法

组合优化理论在相关课程中的地位

运筹学方法

最优化/数学规划方法

① 连续优化：线性规划、非线性规划、非光滑优化、全局优化、变分法、二次规划、分式规划等

② **离散优化：组合优化、线性优化、网络优化、整数规划、几何规划、动态规划**

③ 不确定规划：随机规划、模糊规划等

④ 多目标规划

⑤ 对策论等

随机过程方法

① 统计决策理论

② 马尔科夫过程

③ 排队论

④ 更新理论

⑤ 仿真方法

⑥ 可靠性理论等

统计学方法

① 回归分析

② 群分析

③ 模式识别

④ 实验设计

⑤ 因子分析等

学习该课程需具备的知识

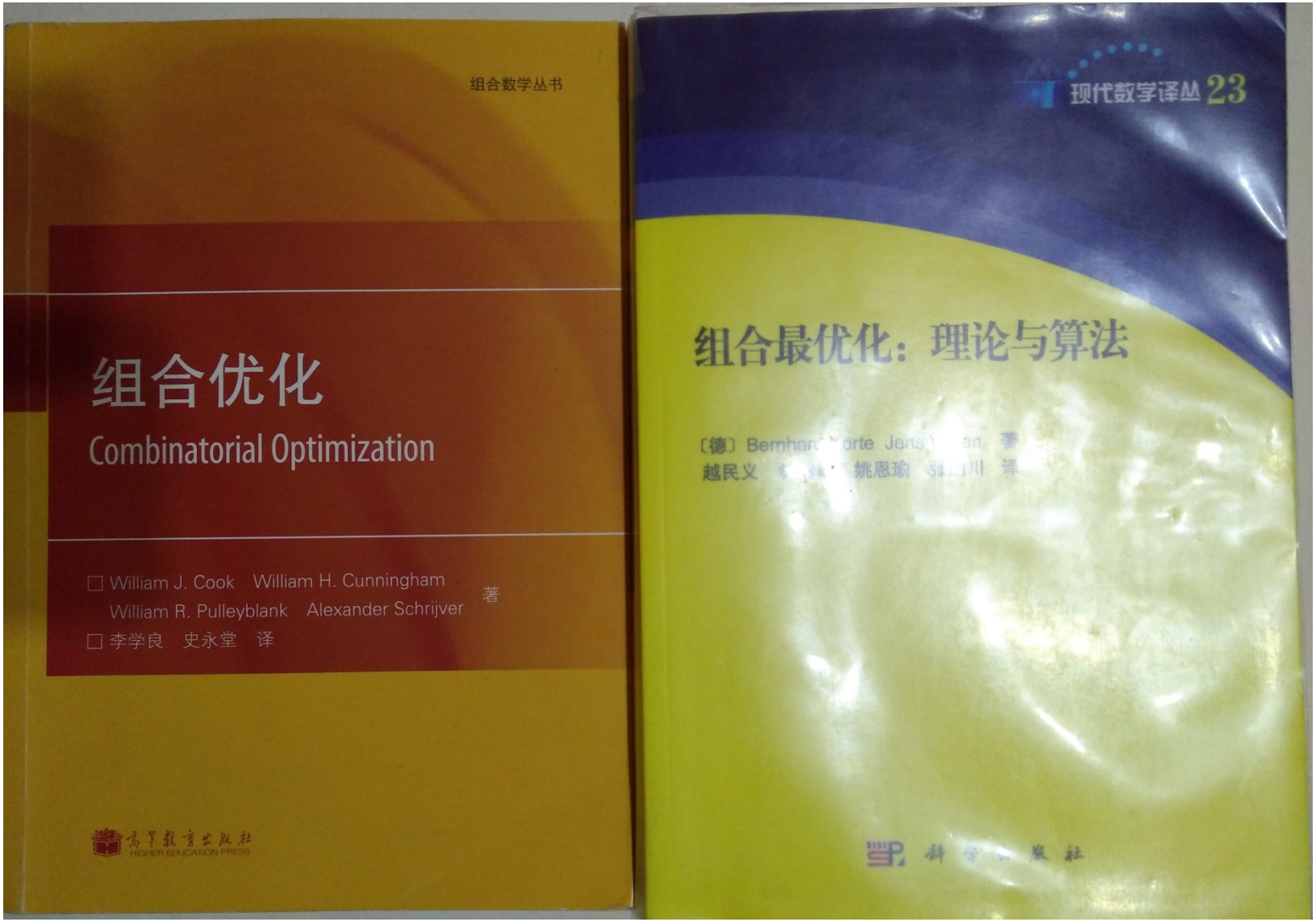
- 组合数学
- 离散数学
- 计算复杂性理论
- 算法分析与设计

教材及参考书

- 最优化理论与算法 （第2版） 陈宝林，清华大学出版社
- 组合优化[Combinatorial Optimization], William J.Cook 等 著,李学良、史永堂 译,高等教育出版社,2011.
- 组合最优化：理论与算法, Bernhard Korte Jens Vygen著, 越民义 姚恩瑜 等译,科学出版社, 2016.01.
- 组合优化导论（第2版）越民义、李荣珩，科学出版社
- 强化学习 邹伟 清华大学出版社

教材及参考书

教材及参考书



课时数及考试

- 讲授40学时
- 考核方式：？卷
- 成绩评定：平时成绩（作业及出勤）占30%，期末考试成绩占70%。
- 作业用 A4 纸手写，不能复印；拍照插入Word文件，并上传QQ群的作业里。

讲授的主要内容

- 相关数学基础
- 单纯形法、对偶单纯型法
- 整数规划、最优指派问题
- 动态规划法（在强化学习中的应用）
- 深度强化学习的基本算法
- 装箱问题、运输问题
- 背包问题
- 作业排序问题
- 旅行售货商问题
- 拟阵理论、NP完全概念

重点讲授上述问题的数学模型和算法的理论基础.

组合优化问题 – 示例1

例1.1　0-1背包问题(knapsack problem)

设有一个容积为b的背包，n个体积分别为a_i (i=1,2,…,n)，价值分别为c_i (i=1,2,…,n)的物品，如何以最大的价值装包？这个问题称为0-1背包问题．用数学模型表示为：

$$\max \sum_{i=1}^{n} c_i x_i \qquad (1.1)$$

$$s.t. \sum_{i=1}^{n} a_i x_i \le b \qquad (1.2)$$

$$x_i \in \{0,1\}, i = 1, \cdots, n \qquad (1.3)$$

其中$x_i = 1$表示装第i个物品，$x_i = 0$表示不装．此时，$D = \{0, 1\}^n$，F为D中满足(1.2)的可行解集.

组合优化问题 – 示例2

例1.2 旅行商问题(TSP,traveling salesman problem)

一个商人欲到n个城市推销商品，每两个城市i和j之间的距离为d_{ij}，如何选择一条道路使得商人**每个城市走仅一遍后回到起点且所走路径最短**.

TSP可分为对称和非对称距离两大类问题.

当 $d_{ij}=d_{ji}, \forall i,j$ 时，称为**对称距离TSP**，否则称为**非对称距离TSP**. 对一般的TSP，它的一种数学模型描述为：

组合优化问题 – 示例2(续)

$$\min \sum_{i \neq j} d_{ij} x_{ij} \qquad (1.4)$$

$$s.t. \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, \cdots, n \qquad (1.5)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, 2, \cdots, n \qquad (1.6)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = n \qquad (1.7)$$

$$x_{ij} \in \{0, 1\}, i, j = 1, \cdots, n, i \neq j \qquad (1.8)$$

其中(1.8)中的决策变量 x_{ij}=1 表示商人行走的路线包含从城市i到城市j 路径，x_{ij}=0 表示商人没有选择走这条路．$i \neq j$ 的约束可以减少变量的个数，使得共有$n \times (n-1)$个决策变量．(1.5)要求商人从**城市i 出来**一次，(1.6)要求商人**走入城市 j**只有一次.

组合优化的数学模型

$Min\ f(x)$或$Max\ f(x)$　　其中x为决策变量　$f(x)$为目标函数

$s.t.\ g(x) \geq 0$　　$g(x)$为约束函数

$x \in D$　　D为决策变量的定义域

$F=\{x|x \in D, g(x) \geq 0\}$ —— 可行域(有限集)

线性规划是连续模型，但由于它的解的特殊结构，也可以作为组合优化问题考虑；也可以给出整数线性规划描述，甚至在一些时候需要利用整数线性规划的技巧来解.

组合优化问题的抽象

可行方案是有限的

组合优化问题是一个**极小化（或极大化）**的问题，它是由以下三部分组成：

（1）实例集合 ；

（2）对每个实例 I，有一个有穷的可行解集合 $F(I)$；

（3）目标函数 f，它对于每个实例 I 和每个可行解 $\sigma \in F(I)$，赋以一个实数 $f(I, \sigma)$. 则实例I的最优解为这样一个可行解 $\sigma^* \in F(I)$,它使得对于所有 $\sigma \in F(I)$，都有：

$$f(I, \sigma^*) \le f(I, \sigma) \quad (f(I, \sigma^*) \ge f(I, \sigma))$$

组合优化的研究

- 怎么才能把一些社会现象、活动等抽象归纳为组合优化问题？数学描述，即抽象建模
- 怎么才能够在尽可能短的时间内求解组合优化问题？ 算法设计，如何设计算法？
- 需要研究各种组合优化问题内在本质？数学规律、定理
- 为了构造快速高效算法，利用哪些性质来设计算法？

  筛选有效的数学规律、定理

组合优化问题的特点

- 如果 D 是有限集合的话，从理论上来讲，只要遍历所有的组合，就能找到最优解。
- 随着问题规模的增大，D 中解的个数会迅速增大，实际上要想遍历所有的解，几乎是不可能的。
- 一般来说，组合优化问题通常带有大量的局部极值点，往往是不可微的、不连续的、多维的、有约束条件的、高度非线性的NP完全(难)问题
- 组合最优化无法利用导数信息精确地求解组合优化问题的全局最优解的“有效”算法一般是不存在的。

组合优化问题的求解方法

- 完全枚举法
- 递归法
- 贪心法
- 随机法
- 松弛法
- 分割法
- 分支定界法
- 邻域搜索法
- 人工智能法

算法分类

- 最优解的算法
  - ① 可以得到问题的最优解，对小规模问题适用，当问题的规模较大时，一般无法在有限时间内获得问题的最优解；
  - ② 对于组合优化问题，通常采用近似求解算法。
- 近似解的算法
  - ① 最大可能地得到更优解，同时满足一定精度；
  - ② 启发式算法(如：神经网络、遗传算法、退火算法等)

最优化算法

对于一个极小化（极大化）优化问题π，如果给定任意一个实例I，算法A总能找到一个可行解$\sigma^* \in D(I)$；使得

$$f(I, \sigma^*) \le f(I, \sigma) \text{ 或者 } f(I, \sigma^*) \ge f(I, \sigma)$$

组合优化总存在最优算法，但它以时间为代价

近似求解算法

设A是优化问题，记问题A的任何一个实例 I 的最优解和启发式算法H解的目标值分别为$Z_{opt}(I)$和$Z_H(I)$,于是对某个$a>0$,称H是A的a-近似算法（*a-approximation algorithm*）,当且仅当

$$|Z_H(I)-Z_{opt}(I)| \le a|Z_{opt}(I)| \quad \forall I \in A$$

用启发式概念定义的算法集合包含了近似算法概念定义的算法集合，近似算法强调给出算法最坏情况的误差界限，而启发式算法不需考虑偏差程度。

启发式算法

- 启发式算法（heuristic algorithm）定义1 一种基于直观或经验构造的算法，在可接受的耗费（指计算时间、占用空间等）下给出待解决优化问题每一实例的一个可行解，该可行解与最优解的偏离程度未必可事先估计。

- 启发式算法定义2 启发式算法是一种技术，该技术使得能在可接受的计算费用内去寻找尽可能好的解，但不一定能保证所得解的可行性和最优性，甚至在多数情况下，无法描述所得解与最优解的近似程度。

启发式算法

启发式算法仿自然体算法为主，主要有:模拟退火算法(SA)、遗传算法(GA)、列表搜索算法(ST)、进化规划(EP)、进化策略(ES)、蚁群算法(ACA)、人工神经网络(ANN)。

邻域的概念

邻域的定义

$$\delta(x_0)=\{x\,|\,|x-x_0|\le\delta\}$$

在光滑函数极值的数值求解中，邻域是一个非常重要的概念。**函数的下降或上升都是在一点的邻域中寻求变化方向**，组合优化中，距离的概念通常不再使用，但是**在一点附近搜索另一个下降的点仍然是组合优化数值求解的基本思想。**

组合问题的邻域定义

① 函数优化问题：

邻域$N(x)$通常定义为在给定距离空间内，以一点x为中心的一个球体

② 组合优化问题： $N: x \in X \rightarrow N(x) \in 2^X$

且 $x \in N(x)$ ，称为一个邻域映射，其中 2^X 表示X所有子集组成的集合。

$N(x)$称为x的邻域，$y \in N(x)$称为x的一个邻居。

邻域例子

在1.2已给出TSP的一种数学模型，由模型 $D=\{x|x\in\{0,1\}^{n\times(n-1)}\}$，可以定义它的一种邻域为：

$$N(x)=\left\{y\middle|\left|y-x\right|=\sum_{i,j}\left|y_{ij}-x_{ij}\right|\le k, y\in D\right\}$$

k为一个正整数．这个邻域定义使得x 最多有k 个位置的值可以发生变化．x 的邻居有

$$C_{n(n-1)}^{1}+C_{n(n-1)}^{2}+\cdots+C_{n(n-1)}^{k}\text{ 个 .}$$

邻域例子（续）

TSP问题解的另外一种表示法为

$D = F = \{s = (i_1, i_2, \cdots, i_n) \mid i_1, i_2, \cdots, i_n$ 是 $1,2,\cdots,n$ 的一个排列 $\}$

定义它的邻域映射为**2-opt**，即S中的**两个元素**进行**对换**， $N(S)$中共包含S 的 C_n^2 个邻居．如四个城市的 TSP问题，当 S =(1,2,3,4)时，

$N(S)$={(2,1,3,4),(3,2,1,4),(4,2,3,1),(1,3,2,4),(1,4,3,2),(1, 2,4,3)}.

启发式算法---局部搜索算法

假设算法用以解决如下组合优化问题：

$$\min f(x)$$

$$s.t.\ g(x) \geq 0$$

$$x \in D$$

其中，$f(x)$为目标函数，$g(x)$为约束函数，D定义域.

局部搜索算法可以简单的表示为：

局部搜索算法

Step1 选一个初始可行解x^0；记录当前最优解

$x^{best}:=x^0$，令$P=N(x^{best})$；

Step2 当$P=\varnothing$时，或满足其他停止运算准则时，输出计算结果，停止运算；否则，从$N(x^{best})$中选一集合S，得到S中的最优解x^{now}；若$f(x^{now})<f(x^{best})$，则$x^{best}:=x^{now}$，$P:=N(x^{best})$；否则；$P:=P-S$；重复step2.

Step1的初始可行解选择可以采用随机的方法，也可用一些经验的方法或其他算法所得到的解．Step2中的集合S选取可以大到是$N(x^{best})$本身，也可以小到只有一个元素，如用随机的方 法在$N(x^{best})$中选一点．

局部搜索算法---TSP求解示例

例：5个城市（A,B,C,D,E）的对称TSP数据，对应的距离矩阵为

$$D=(d_{ij})=\begin{bmatrix} 0 & 10 & 15 & 6 & 2 \\ 10 & 0 & 8 & 13 & 9 \\ 15 & 8 & 0 & 20 & 15 \\ 6 & 13 & 20 & 0 & 5 \\ 2 & 9 & 15 & 5 & 0 \end{bmatrix}$$

初始解为x^{best}=（ABCDE），$f(x^{best})=45$.邻域映射定义为对换两个城市位置的2-opt．选定A城市为起点，用全邻域搜索，即$S:=N(x^{best})$

局部搜索算法---TSP求解示例（续）

方案一：一步随机搜索

第1步

从$N(x^{best})$中随机选一点，如$x^{now}=(ACBDE)$，

对应目标函数为$f(x^{now})=43<45$

$x^{best}:=x^{now}=(ACBDE)$

第2步

从$N(x^{best})$中又随机选一点，如$x^{now}=(ADBCE)$，

对应目标函数为$f(x^{now})=44>43$

$x^{best}:=x^{now}=(ACBDE)$

局部搜索算法---TSP求解示例（续）

方案二：全邻域搜索

第1循环： 初始解为x^{best}=(*ABCDE*)，$f(x^{best})$=45，定义邻域映射为对换两个城市位置的2-opt，选定A城市为起点。

$N(x^{best})$={(*ABCDE*)，(*ACBDE*)，(*ADCBE*)，(*AECDB*)，(*ABDCE*)，(*ABEDC*)，(*ABCED*)}，对应目标函数为f(x)={45, 43, 45, 60, 60, 59, 44}， }. $x^{best} := x^{now}$=(ACBDE）

第2循环 $N(x^{best})$={(*ACBDE*)，(*ABCDE*)，(*ADBCE*)，(*AEBDC*)，(*ACDBE*)，(*ACEDB*)，(*ACBED*)}，对应目标函数为*f*(*x*)={43, 45, 44, 59, 59, 58, 43}， $x^{best}:=x^{now}$=(*ACBDE*)

此时，$P=N(x^{best})-S$ 为空集，于是所得解为(ACBDE），目标值为43．局部搜索算法的优点是简单易行，容易理解，但其缺点是无法保证全局最优性．因其只按下降规则转移状态．

蒙特卡罗算法

改变局部搜索算法中只按下降规则转移状态的一个方法是蒙特卡罗方法，主要变化是局部搜索算法的第二步为：

Step2 当满足停止运算准则时，停止运算；否则，从$N(x^{best})$中随机选一点x^{now}；若 $f(x^{now}) \le f(x^{best})$，则 $x^{best} := x^{now}$；否则根据 $f(x^{now}) - f(x^{best})$以一定的概率接受x^{now}（ $x^{best} := x^{now}$ ）；返回step2.

蒙特卡罗算法是**以一定的概率接受一个较坏的状态**，如可以均匀的概率

$$p=\frac{1}{\left|N(x^{best})\right|}$$

从$N(x^{best})$中选任意一点，以概率

$$p=\min\left\{1,\exp\left\{-\frac{f(x^{now})-f(x^{best})}{t}\right\}\right\}$$

接受x^{now}．模拟退火算法就是采用这样的搜索方法.

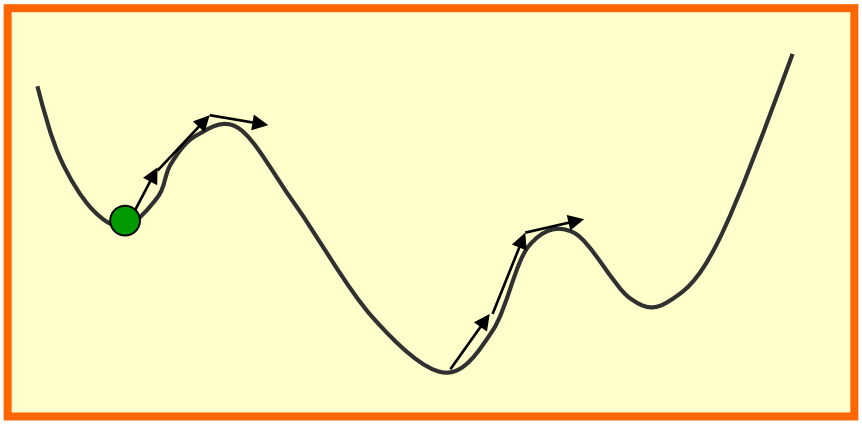
对局部搜索的思考

① 简单易行，但无法保证全局最优性；

② 局部搜索主要依赖起点的选取和邻域的结构；

③ 为了得到好的解，可以比较不同的邻域结构和不同的初始点；

④ 如果初始点的选择足够多，总可以计算出全局最优解。



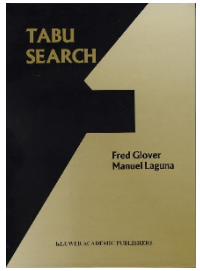
禁忌搜索

禁忌搜索算法（Tabu Search或Taboo Search，简称TS算法）是一种全局性邻域搜索算法，模拟人类具有记忆功能的寻优特征。它通过局部邻域搜索机制和相应的禁忌准则来避免迂回搜索，并通过破禁水平来释放一些被禁忌的优良状态，进而保证多样化的有效探索，以最终实现全局优化。

禁忌搜索

- **算法的提出**

  禁忌搜索（**Tabu search**）是局部邻域搜索算法的推广，**Fred Glover**在**1986**年提出这个概念，进而形成一套完整算法。



http://spot.colorado.edu/~glover/

- **算法的特点**

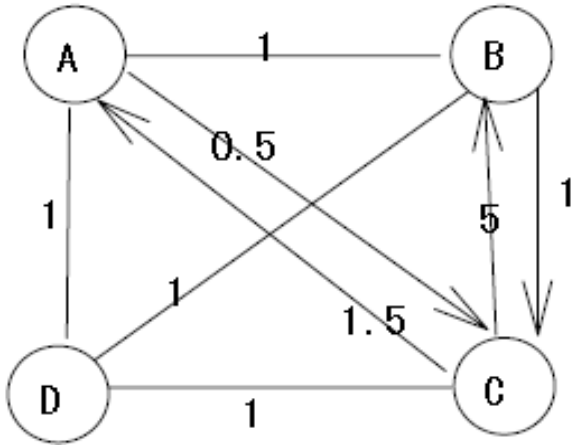
  禁忌——禁止重复前面的工作。

  跳出局部最优点，避免走回头路。

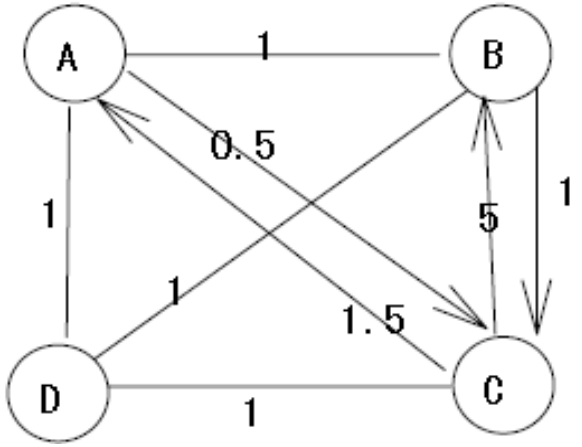
禁忌搜索示例

- **四城市非对称TSP问题**

$$D=(d_{ij})=\begin{bmatrix} 0 & 1 & 0.5 & 1 \\ 1 & 0 & 1 & 1 \\ 1.5 & 5 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



初始解$x^0=(ABCD)$，$f(x^0)=4$，邻域映射为两个城市顺序对换的2－opt，始、终点都是A城市。



- **四城市非对称TSP问题**

第1步

解的形式

A	B	C	D

$f(x^0)=4$

禁忌对象及长度

	B	C	D
A			
B			
C			

候选解

对换	评价值
CD	4.5 √
BC	7.5
BD	8

第2步

解的形式

A	B	D	C

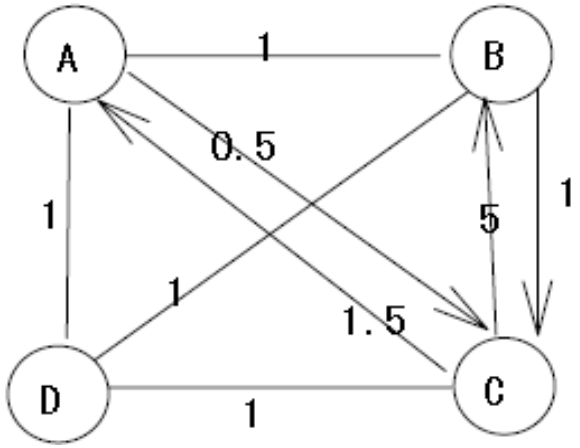
$f(x^1)=4.5$

禁忌对象及长度

	B	C	D
A			
B			
C			3

候选解

对换	评价值
CD	4.5 T
BC	3.5 √
BD	4.5



第3步

解的形式

A	C	D	B

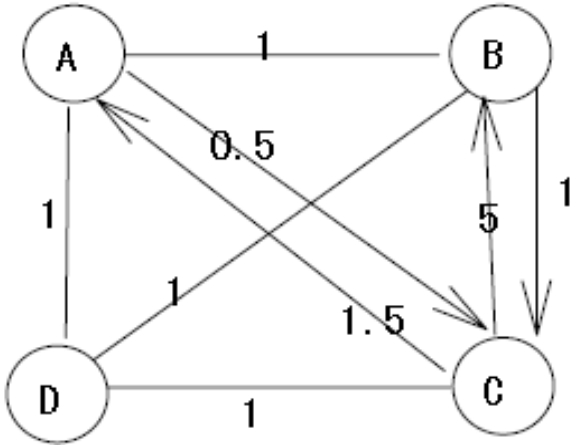
$f(x^2)=3.5$

禁忌对象及长度

	B	C	D
A			
B	3		
C		2	

候选解

对换	评价值	
CD	8	T
BC	4.5	T
BD	7.5	√



四城市非对称TSP问题

第4步

解的形式　　禁忌对象及长度　　候选解

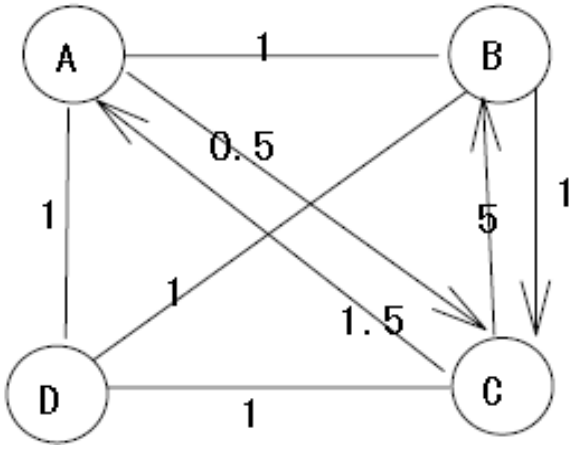
A	C	B	D

$f(x^3)=7.5$

	B	C	D
A			
B	2	3	
C		1	

对换	评价值
CD	4.5T
BC	4.5T
BD	3.5T

禁忌长度的选取



四城市非对称TSP问题

第4步（如果减小禁忌长度）

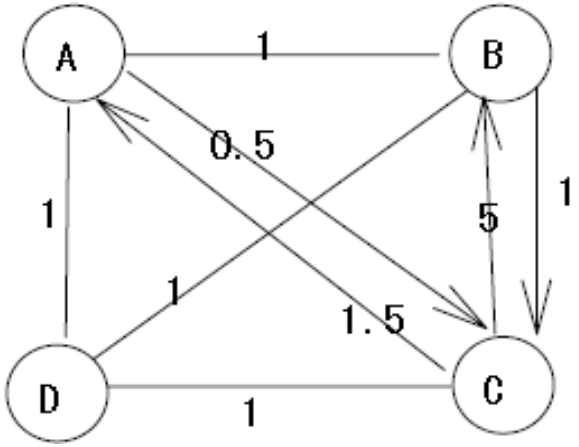
解的形式　　禁忌对象及长度　　候选解

A	C	B	D

$f(x^3)=7.5$

	B	C	D
A			
B	1	2	
C		0	

对换	评价值
CD	4.5√
BC	4.5T
BD	3.5T



四城市非对称TSP问题

第5步

解的形式

A	D	B	C

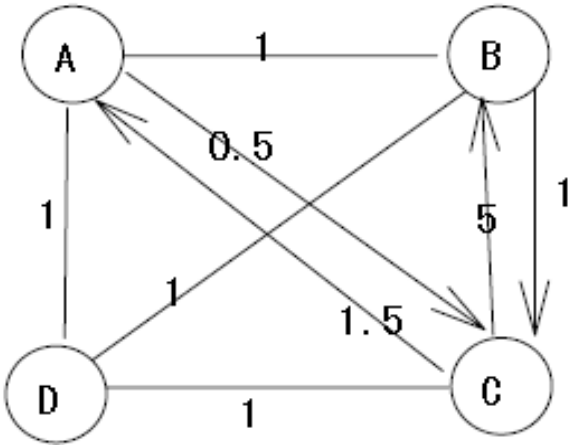
$f(x^4)=4.5$

禁忌对象及长度

	B	C	D
A			
B	0	1	
C		2	

候选解

对换	评价值
CD	7.5T
BC	8 √
BD	4.5T



四城市非对称TSP问题

第6步

解的形式

A	D	C	B

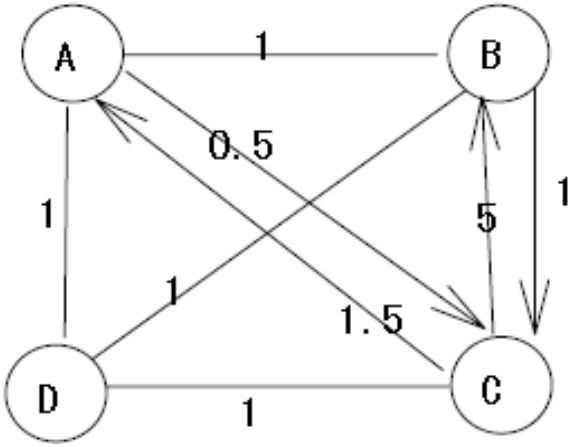
$f(x^5)=8$

禁忌对象及长度

	B	C	D
A			
B	2	0	
C		1	

候选解

对换	评价值
CD	3.5T
BC	4.5T
BD	4 √



禁忌搜索的关键参数和操作

变化因素

- **禁忌表的主要指标（两项指标）**

  禁忌对象：禁忌表中被禁的那些变化元素

  禁忌长度：禁忌的步数

- **状态变化（三种变化）**

  解的简单变化

  解向量分量的变化

  目标值变化

- **解的简单变化**

假设$x, y \in D$，邻域映射为N，其中D为优化问题的定义域，则简单解变化 $x \to y \in N(x)$ 是从一个解变化到另一个解。

- **向量分量的变化**

设原有的解向量为$(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$，向量分量的最基本变化为

$$(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \to (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$$

即只有第i个分量发生变化。

也包含多个分量变化的情形。

- **目标值的变化**

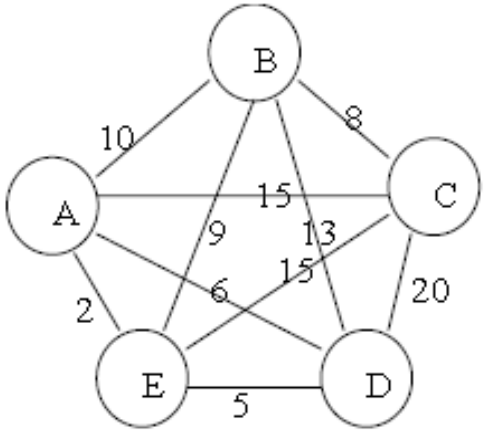
目标值的变化隐含着解集合的变化。

禁忌表

- ## 禁忌对象的选取

情况1：禁忌对象为简单的解变化

$$D=(d_{ij})=\begin{bmatrix}0&10&15&6&2\\10&0&8&13&9\\15&8&0&20&15\\6&13&20&0&5\\2&9&15&5&0\end{bmatrix}$$



禁忌长度为4，从2－opt邻域中选出最佳的5个解组成候选集Can_$N(x^{now})$，初始解$x^{now}=x^0=(ABCDE)$，$f(x^0)=45$，$H=\{(ABCDE;45)\}$。

- **禁忌对象的选取**

情况1：禁忌对象为简单的解变化

第1步——

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\{(ABCDE;45)\}$

$\text{Can\_}N(x^{now})=\{(ACBDE;43)$，$(ABCDE;45)$，$(ADCBE;45)$，$(ABEDC;59)$，$(ABCED;44)\}$。

$x^{next}=(ACBDE)$

- **禁忌对象的选取**

情况1：禁忌对象为简单的解变化

第2步——

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{(ABCDE;45),(ACBDE;43)\}$

$\text{Can\_}N(x^{now})=\{(ACBDE;43)$，$(ACBED;43)$，$(ADBCE;44)$，$(ABCDE;45)$，$(ACEDB;58)\}$。

$x^{next}=(ACBED)$

- **禁忌对象的选取**

情况1：禁忌对象为简单的解变化

第3步——

$x^{now}=(ACBED)$，$f(x^{now})=43$，$H=\{(ABCDE;45)$，$(ACBDE;43)$，$(ACBED;43)\}$

$Can\_N(x^{now})=\{(ACBED;43)$，$(ACBDE;43)$，$(ABCED;44)$，$(AEBCD;45)$，$(ADBEC;58)\}$。

$x^{next}=(ABCED)$

- **禁忌对象的选取**

情况1：禁忌对象为简单的解变化

第4步——

$x^{now}=(ABCED)$，$f(x^{now})=44$，$H=\{(ABCDE;45)$，$(ACBDE;43)$，$(ACBED;43)$，$(ABCED;44)\}$

$Can\_N(x^{now})=\{(ACBED;43)$，$(AECBD;44)$，$(ABCDE;45)$，$(ABCED;44)$，$(ABDEC;58)\}$。

$x^{next}=(AECBD)$

- **禁忌对象的选取**

情况1：禁忌对象为简单的解变化

第5步——

$x^{now}=(AECBD)$，$f(x^{now})=44$，$H=\{(ACBDE;43)$，$(ACBED;43)$，$(ABCED;44)$，$(AECBD;44)\}$

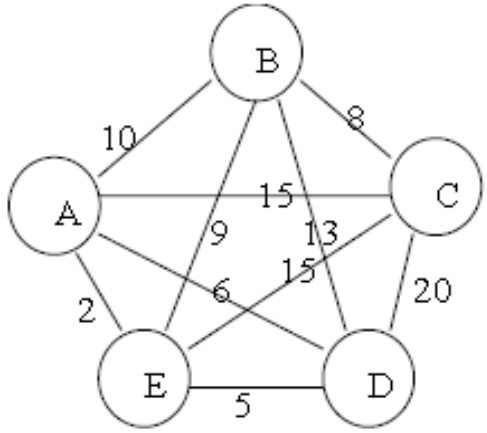
$\text{Can\_}N(x^{now})=\{(AEDBC;43)$，$(ABCED;44)$，$(AECBD;44)$，$(AECDB;44)$，$(AEBCD;45)\}$。

$x^{next}=(AEDBC)$

- **禁忌对象的选取**

情况2：禁忌对象为分量变化

$$D=(d_{ij})=\begin{bmatrix}0&10&15&6&2\\10&0&8&13&9\\15&8&0&20&15\\6&13&20&0&5\\2&9&15&5&0\end{bmatrix}$$



禁忌长度为3，从2－opt邻域中选出最佳的5个解组成候选集Can_$N(x^{now})$，初始解$x^{now}=x^0=(ABCDE)$，$f(x^0)=45$。

- **禁忌对象的选取**

情况2：禁忌对象为分量变化

第1步——

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\Phi$

$\text{Can\_}N(x^{now})=\{(ACBDE;43)$，$(ADCBE;45)$，$(AECDB;60)$，$(ABEDC;59)$，$(ABCED;44)\}$。

$x^{next}=(ACBDE)$

- **禁忌对象的选取**

情况2：禁忌对象为分量变化

第2步——

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{(B,C)\}$

$\text{Can\_}N(x^{now})=\{(ACBED;43)$，$(ADBCE;44)$，$(ABCDE;45)$，$(ACEDB;58)$，$(AEBDC;59)\}$。

$x^{next}=(ACBED)$

- **禁忌对象的选取**

情况2：禁忌对象为分量变化

第3步——

$x^{now}=(ACBED)$，$f(x^{now})=43$，$H=\{(B,C)，(D,E)\}$

$\text{Can\_}N(x^{now})=\{(ACBDE;43)，(ABCED;44)，$

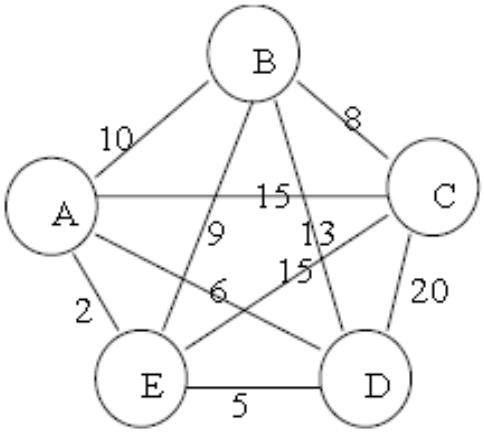
$(AEBCD;45)，(ADBEC;58)，(ACEBD;58)\}$。

$x^{next}=(AEBCD)$

禁忌对象的选取

情况3：禁忌对象为目标值变化

$$D=(d_{ij})=\begin{bmatrix}0 & 10 & 15 & 6 & 2\\ 10 & 0 & 8 & 13 & 9\\ 15 & 8 & 0 & 20 & 15\\ 6 & 13 & 20 & 0 & 5\\ 2 & 9 & 15 & 5 & 0\end{bmatrix}$$



禁忌长度为3，从2－opt邻域中选出最佳的5个解组成候选集Can_$N(x^{now})$，初始解$x^{now}=x^0=(ABCDE)$，$f(x^0)=45$。

- **禁忌对象的选取**

情况3：禁忌对象为目标值变化

第1步——

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\{45\}$

$\text{Can\_}N(x^{now})=\{(ABCDE;45)$，$(ACBDE;43)$，$(ADCBE;45)$，$(ABEDC;59)$，$(ABCED;44)\}$。

$x^{next}=(ACBDE)$

- **禁忌对象的选取**

情况3：禁忌对象为目标值变化

第2步——

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{45，43\}$

$\text{Can\_}N(x^{now})=\{(ACBDE;43)，(ACBED;43)，$

$(ADBCE;44)，(ABCDE;45)，(ACEDB;58)\}$。

$x^{next}=(ADBCE)$

- **禁忌对象的选取**

（1）解的简单变化比解的分量变化和目标值变化的受禁范围要小，可能造成计算时间的增加，但也给予了较大的搜索范围；

（2）解分量的变化和目标值变化的禁忌范围大，减少了计算时间，可能导致陷在局部最优点。

禁忌长度的选取

（1）t可以为常数，易于实现；

（2）$t \in [t_{min}, t_{max}]$，t是可以变化的数，t_{min}和t_{max}是确定的。

t_{min}和 t_{max}根据问题的规模确定，t的大小主要依据实际问题、实验和设计者的经验。

（3） t_{min}和t_{max}的动态选择。

- **禁忌长度的选取**

禁忌长度过短，一旦陷入**局部最优点**，出现循环无法跳出；

禁忌长度过长，造成**计算时间较大**，也可能造成计算无法继续下去。

- **特赦（藐视）原则**

（1）**基于评价值的规则**，若出现一个解的目标值好于前面任何一个最佳候选解，可特赦；

（2）**基于最小错误的规则**，若所有对象都被禁忌，特赦一个评价值最小的解；

（3）**基于影响力的规则**，可以特赦对目标值影响大的对象。

- **候选集合的确定**

（1）从邻域中选择若干目标值最佳的邻居入选；

（2）在邻域中的部分邻居中选择若干目标值最佳的状态入选；

（3）随机选取。

- **评价函数**

（1）**直接评价函数**，通过目标函数的运算得到评价函数；

（2）**间接评价函数**，构造其他评价函数替代目标函数，应反映目标函数的特性，减少计算复杂性。

记忆频率信息

根据记忆的频率信息（禁忌次数等）来控制禁忌参数（禁忌长度等）。

例如：如果一个元素或序列重复出现或目标值变化很小，可增加禁忌长度以避开循环；

如果一个最佳目标值出现频率很高，则可以终止计算认为已达到最优值。

记忆频率信息

可记录的信息：

（1）静态频率信息：解、对换或目标值在计算中出现的频率；

（2）动态频率信息：从一个解、对换或目标值到另一个解、对换或目标值的变化趋势。

- **终止规则**

（1）**确定步数终止**，无法保证解的效果，应记录当前最优解；

（2）**频率控制原则**，当某一个解、目标值或元素序列的频率超过一个给定值时，终止计算；

（3）**目标控制原则**，如果在一个给定步数内，当前最优值没有变化，可终止计算。

启发式算法的长处

1）数学模型是实际问题的简化，有可能使最优算法所得解比启发式算法所得解产生更大误差；

2）不少难的组合优化问题可能还没找到有效的最优算法；

3）一些启发式算法可以用在最优算法中，如在分支定界算法中，可以用启发式算法估界；

4）简单易行，比较直观，易被使用者接受；

5）速度快，在适时管理中非常重要；

6）多数情况下，程序简单，因此易于修改。

启发式算法的不足

1）不能保证求得最优解；

2）表现不稳定；

3）算法的好坏依赖于**实际问题、设计者的经验和技术**，使不同算法之间难以比较。

启发式算法的性能分析

- 评价启发式算法的性能有不同的角度，主要分为：

  （1）对算法复杂性的分析

  （2）对算法计算效能的分析

对随机算法还要考虑稳定性

- 性能分析的常用方法

  （1） 最坏情形分析

  （2） 概率分析

  （3） 大规模计算分析

每个方法都可以从以上两个角度进行分析

性能分析---最坏情形分析

从最坏情形分析评价一个算法的计算效果时，其评价的指标是计算解的目标值同最优目标值最坏情形时的差距，这个差距越小，说明算法越好。

若 D 是一个极大化问题，实例I的最优值为$Z_{opt}(I)$,启发式算法H所得的解值为$Z_H(I)$，记 $R_H(I)=\dfrac{Z_H(I)}{Z_{opt}(I)}$

则启发式算法H的绝对性能比 R_H 定义为

$$R_H = \sup_I \left\{ r \le 1 \middle| R_H(I) \ge r \right\}$$

性能分析---最坏情形分析

如何求（或证明）$R_H = a$

界

i 寻找（证明）存在常数 a 使得对任意的实例 I ,有

$Z_H(I) \geq aZ_{opt}(I)$ 则 $R_H \geq a$;

ii 构造实例 I 使 $\dfrac{Z_H(I)}{Z_{opt}(I)} = a$

或构造一系列实例$\{I_n\}$ 使 $\dfrac{Z_H(I_n)}{Z_{opt}(I_n)} \to a \quad (n \to \infty)$

则 $R_H \leq a$ 从而 有 $R_H = a$

$$R_H = \sup_I \left\{ r \leq 1 \middle| R_H(I) \geq r \right\}$$

性能分析---最坏情形分析

启发式算法 H 的渐近性能比 R_H^{∞}

$$R_H^{\infty} = \lim_{k\to\infty} \sup_{I} \left\{ \frac{Z_H(I)}{Z_{opt}(I)} \middle| Z_{opt}(I) \geq k \right\}$$

启发式算法的性能比是对算法近似程度的一种最坏情形分析，R_H、R_H^{∞} 越接近1 表示启发式解越接近最优解。

一般有：$R_H \leq R_H^{\infty}$

性能分析---概率分析

最坏情形分析是以最坏的实例来评价一个算法或其解，这样不免会只因一个实例而影响对一个算法或其解的总体评价。

概率分析则是**假设实例的数据服从一定的概率分布**，在这个数据概率分布的假设下，研究其算法或解的平均效果。

最坏情形分析和概率分析方法的共同特点，**用理论方法分析算法同最优解之间的误差界**，要求有较强的数学基础。

性能分析---大规模计算分析

实际中大量的组合优化问题是采用大规模计算分析

大规模计算分析就是通过大量的实例计算评价算法的计算效果

算法的计算效果分成两个方面：

计算复杂性： 它的效果通过计算机的*CPU*的计算时间表现；

计算解的性能： 它通过计算停止时,输出的解表现。

大规模计算分析方法的作用

对一个算法进行评价：

1、*计算时间效果* 往往通过目前的计算机设备能否承受，用户能否接受现有的计算时间来衡量；

2、*计算解效果* 一个简单的要求是用户是否满意，更深一步需要知道问题的最优解或问题的上下界.

设：实例I的最优值$Z_{opt}(I)$，实例I的上界$Z_{UB}(I)$

算法H的目标值$Z_H(I)$

大规模计算分析方法的作用（续）

绝对差 $Z_H(I)-Z_{opt}(I)$ 或 $Z_H(I)-Z_{UB}(I)$

相对差 $\dfrac{Z_H(I)-Z_{opt}(I)}{Z_{opt}(I)}$ 或 $\dfrac{Z_H(I)-Z_{UB}(I)}{Z_H(I)}$

解的稳定性分析 $D=\sum_{I\in S}(Z_H(I)-\overline{H})^2$

其中 $\overline{H}=\dfrac{1}{|S|}\sum_{I\in S}Z_H(I)$

S表示所有数据实例的集合，$|S|$表示S中包含的实例数。

对多个算法进行评价

通过大量数据的计算, 比较分析不同算法的效果, 采用简单或统计的方法比较不同算法的性能. 这种比较不需要已知问题的最优解或上下界

设H_1,H_2是极大化问题的两个算法

对于任意实例I,有 $Z_{H_1}(I) \leq Z_{H_2}(I)$

则称H_2不次于H_1

Note **产生的数据要具有代表性**

一个多项式时间可解的问题一般不必用启发式算法

计算复杂性问题

Definition 若存在一个常数 C，使得对所有 $n \geq 1$, 都有 $|f(n)| \leq C\ |g(n)|$,则称函数 $f(n)$ 是 $O(g(n))$.

设 A 是解某一问题 D的算法，对 D 的任一规模为 n 的实例，可在 n 的多项式时间内求解（即计算复杂性为 $O(n^{\alpha})$）,则称算法 A 为一个解问题 D 的多项式时间算法.（简称多项式算法） 不能这样限制时间复杂性函数的算法称为指数时间算法.

多项式时间算法：$O(n\ logn)$、$O(n^{2.8})$、$O(n^{2})$

指数时间算法：$O(n!)$、$O(3^{n})$

易解问题与难解问题

第一类：如果对一个问题Q 存在一个算法，时间复杂性为$O(n^k)$，其中n是问题规模，k是一个非负整数，则称问题π 存在**多项式时间**算法。这类算法在**可以接受的时间内**实现问题求解， **e.g., 排序、串匹配、矩阵相乘。**

第二类：现实世界里还存在很多问题至今没有找到多项式时间算法，计算时间是**指数和超指数函数**（如2^n和$n!$），随着问题规模的增长而**快速增长**。

通常将前者看作是**易解问题**，后者看作**难解问题**。

易解问题与难解问题的主要区别

在学术界已达成这样的共识：把多项式时间复杂性作为易解问题与难解问题的分界线，主要原因有：

1） 多项式函数与指数函数的增长率有本质差别

问题规模 n	多项式函数					指数函数	
	logn	n	nlogn	n^2	n^3	2^n	n!
1	0	1	0.0	1	1	2	1
10	3.3	10	33.2	100	1000	1024	3628800
20	4.3	20	86.4	400	8000	1048376	2.4E18
50	5.6	50	282.2	2500	125000	1.0E15	3.0E64
100	6.6	100	664.4	10000	1000000	1.3E30	9.3E157

2) 计算机性能的提高对易解问题与难解问题算法的影响

假设求解同一个问题有5个算法A1~A5，时间复杂度T(n)如下表，假定计算机C2的速度是计算机C1的10倍。下表给出了在相同时间内不同算法能处理的问题规模情况：

T(n)	C1 n	C2 n′	变化	n′/n
10n	1000	10000	n′=10n	10
20n	500	5000	n′=10n	10
5nlogn	250	1842	$\sqrt{10}$ n<n′<10n	7.37
$2n^2$	70	223	$\sqrt{10}$ n	3.16
2^n	13	16	n′=n+log10≈n+3	≈1

上面说明：计算机的处理速度对指数时间复杂度算法无显著改善

3） 多项式时间复杂性忽略了系数，但不影响易解问题与难解问题的划分

问题规模n	多项式函数			指数函数	
	n^8	10^8n	n^{1000}	1.1^n	$2^{0.01n}$
5	390625	5×10^8	5^{1000}	1.611	1.035
10	10^8	10^9	10^{1000}	2.594	1.072
100	10^{16}	10^{10}	10^{2000}	13780.6	2
1000	10^{24}	10^{11}	10^{1000}	2.47×10^{41}	1024

观察结论：n≤100时，（不自然的）多项式函数值大于指数函数值，但n充分大时，指数函数仍然超过多项式函数。

P类问题和NP类问题

这个划分标准是基于对所谓判定问题的求解方式。

先看看什么是判定问题。事实上，实际应用中的大部分问题问题可以容易转化为相应的判定问题，如：

排序问题 ⇒ 给定一个实数数组，是否可以按非降序排列?

图着色问题：给定无向连通图G=(V,E)，求最小色数k，使得任意相邻点具有不同的着色 ⇒

给定无向连通图G=(V,E)和正整数k，是否可以用k种颜色..... ?

确定性算法与P类问题

对判定问题求解，可以采用**确定性算法**

- **确定性算法**: 设A是求解问题Q的一个算法，如果在算法的整个执行过程中，每一步只有一个确定的选择，则称算法A是确定性算法。
  - ➢ 特点：对同一输入实例，运行算法A，所得结果是一样的。
- **P类问题**: 如果对于某个判定问题Q，存在一个非负整数k，对于输入规模为n的实例，能够以$O(n^k)$的时间运行一个确定性算法，得到yes或no的答案，则称该判定问题∏是一个P(Polynomial)类问题。
  - ➢ 事实上，所有易解问题都是P类问题。

非确定性算法与NP类问题

- **非确定性算法:**设A是求解问题Q的一个算法，如果算法A以如下猜测并验证的方式工作，称算法A为非确定性(nondeterminism)算法：
- **猜测阶段：**对问题的输入实例产生一个任意字串y，在算法的每次运行，y可能不同，因此猜测是以非确定的形式工作。这个工作一般可以在线性时间内完成。
- **验证阶段：**在这个阶段，用一个确定性算法验证两件事：首先验证猜测的y是否是合适的形式，若不是，则算法停下并回答no；若是合适形式，则继续检查它是否是问题Q的解，如果确实是Q的解，则停下并回答yes，否则停下并回答no。要求验证阶段在多项式时间内完成。

注意对非确定性算法输出yes/no的理解：

若输出no，并不意味着不存在一个满足要求的解，因为猜测可能不正确；若输出yes，则意味着对于该判定问题的某一输入实例，至少存在一个满足要求的解。

NP类问题

NP类问题: 如果对于判定问题Q，存在一个非负整数k，对于输入规模为n的实例，能够以$O(n^k)$的时间运行一个非确定性算法，得到yes/no的答案，则该判断问题∏是一个NP(nondeterministic polynomial)类问题。

※注意：NP类问题是对于判定问题定义的，事实上，可以在多项式时间内应用非确定性算法解决的所有问题都属于NP类问题。

关于P与NP关系的初步思考--从字面含义

1) 若问题Q属于P类，则存在一个多项式时间的确定性算法，对它进行判定或求解；显然，也可以构造一个多项式时间的非确定性算法，来验证解的正确性，因此，问题也属NP类。因此，显然有

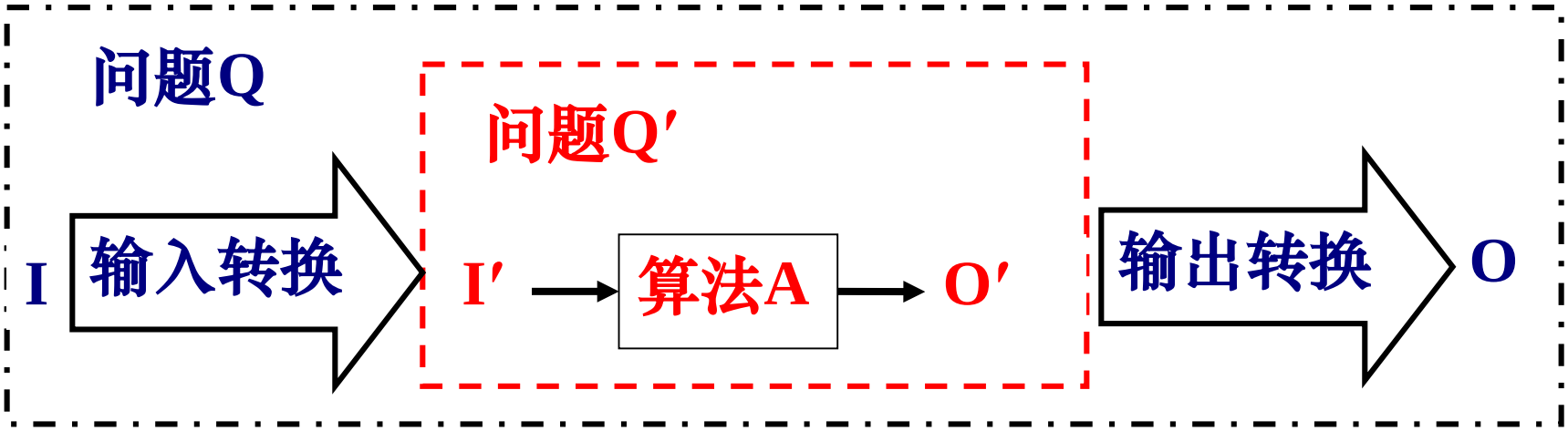
$$P \subseteq NP$$

2) 若问题Q属于NP类，则存在一个多项式时间的非确定性算法，来猜测并验证它的解；但不一定能构造一个多项式时间的确定性算法，来对它进行求解或判定。

因此，人们猜测$P \neq NP$，但是否成立，至今未得到证明。

问题变换

- NP类问题在最坏情况下的时间复杂性一般都是快速增长的指数函数。我们希望能够在NP类问题内部找到一种方法，比较两个问题的复杂性。
- 比较两个问题的计算复杂性的方法是问题变换。



多项式时间变换

- **多项式时间变换**:若在$O(\tau(n))$时间内完成上述输入/输出转换，则称**问题Q以$\tau(n)$时间变换到问题Q′**，记为

$$Q \propto_{\tau(n)} Q'$$

- 其中，n为问题规模；若在多项式时间内完成上述转换，则称问题Q以多项式时间变换到问题Q′，记为

$$Q \propto_{poly} Q'$$

举例：多项式时间变换

排序问题的算法A，输入为$x_1, x_2,..., x_n$, 输出为非降序序列

$$x_{i1} \le x_{i2} \le ... \le x_{in} ;$$

配对问题Q：输入两组数$X=(x_1, x_2,..., x_n,)$和$Y=(y_1, y_2,..., y_n,)$，输出是两组元素的非降序依次配对。

求解配对问题，需要进行三个变换

① 将配对问题的输入X,Y变成排序问题的两个输入$I_{1'}, I_{2'}$；

② 应用**算法A**对$I_{1'}, I_{2'}$分别排序，得到两个排序输出$O_{1'}, O_{2'}$；

③ 将两个排序输出$O_{1'}, O_{2'}$转换成配对问题的输出O。

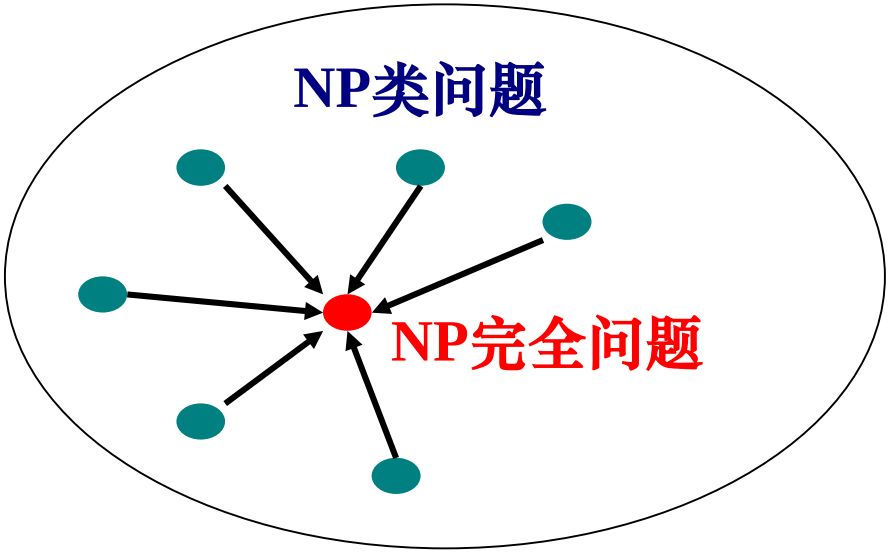
这些操作可以在多项式时间内完成。

多项式时间变换的性质

- **传递性**：设Q、Q1和Q2,是3个判定问题，若$Q \propto_{\tau(n)} Q1$，且$Q1 \propto_{\tau(n)} Q2$，则$Q \propto_{\tau(n)} Q2$。

NP完全问题

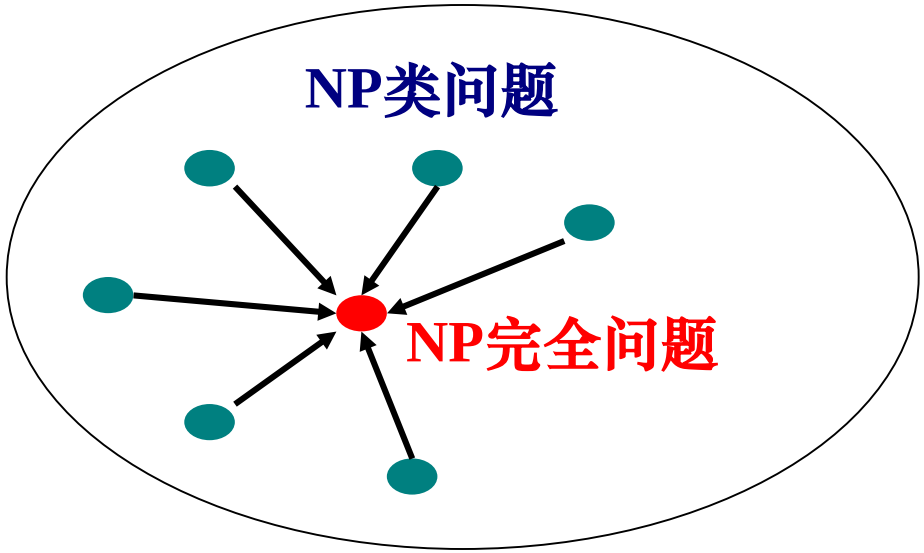
NP完全问题是一类具备如下特殊性质的NP类问题



- **该问题Q本身就是一个NP类问题**
- **每一个NP类问题都可以通过多项式时间化为Q**

NP完全问题的定义

- **NP完全问题**: 令Q是一个判定问题，如果问题Q属于NP类问题，并且对NP类问题中的每一个问题Q1，都有$Q1 \propto_{poly} Q$，则称判定问题Q是一个NP完全问题(NP complete problem, NPC)。



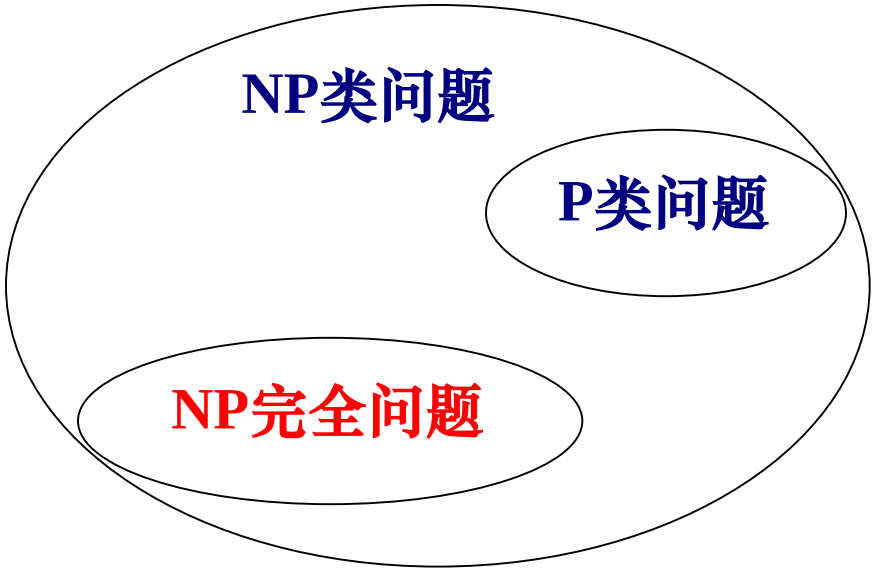
对“NP完全问题”的评述

- NP完全问题是NP类问题中最难的

  一类问题，至今已经发现了几千个，

  但一个也没有找到多项式时间算法。
- 如果某一个NP完全问题能在多项式时间内解决，则每一个NP完全问题都能在多项式时间内解决。(?)
- 这些问题也许存在多项式时间算法，因为计算机科学是相对新生的科学，肯定还有新的算法设计技术有待发现；
- 这些问题也许不存在多项式时间算法，但目前缺乏足够的依据来证明这一点。

P类问题、NP类问题、NP完全问题的关系



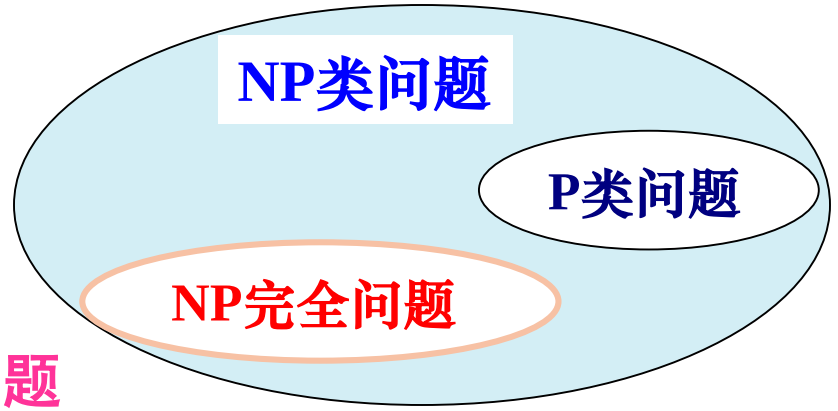
关于P与NP关系的再思考
--从深层意义

至今没有人能证明是否P ≠ NP。

- 若P=NP，说明NP类中所有问题

  （包括NP完全问题）都具有

  多项式时间算法；



- 若P ≠ NP，

  说明NP类中除P类之外的所有问题

  （包括NP完全问题）都不存在多项式时间算法。

  无论哪种答案，都将为算法设计提供重要的指导和依据。

目前人们普遍猜测：P ≠ NP

最基本的NP完全问题

- 1971年，美国的Cook证明了Cook定理：布尔表达式的可满足性(SAT)问题是NP完全的。

可满足性问题即合取范式的可满足性问题，来源于许多实际的逻辑推理的应用。合取范式形如$A_1 \wedge A_2 \wedge ... \wedge A_n$，其中子句$A_i$ $(1 \le i \le n)$形如：$a_1 \vee a_2 \vee ... \vee a_k$, 其中$a_j$称为文字，为布尔变量。SAT问题是指：是否存在一组对所有布尔变量的赋值，使得整个合取范式为真？例如

$$f = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee x_3 \vee x_4 \vee \overline{x}_5) \wedge (x_1 \vee \overline{x}_3 \vee x_4)$$

当x_1和x_3都为真、其余文字任意赋值时，f值为真。

其他NP完全问题

- **可满足性(SAT)问题是第一个被证明的NP完全问题。**

  1972年，Karp证明了十几个问题都是NP完全的。这些NP完全问题的证明思想和技巧，以及利用他们证明的几千个NP完全问题，极大地丰富了NP完全理论。

- **已证明的NP完全问题：**

  SAT问题 、最大团问题 、图着色问题 、

  哈密顿回路问题 、旅行商问题 、背包问题 、

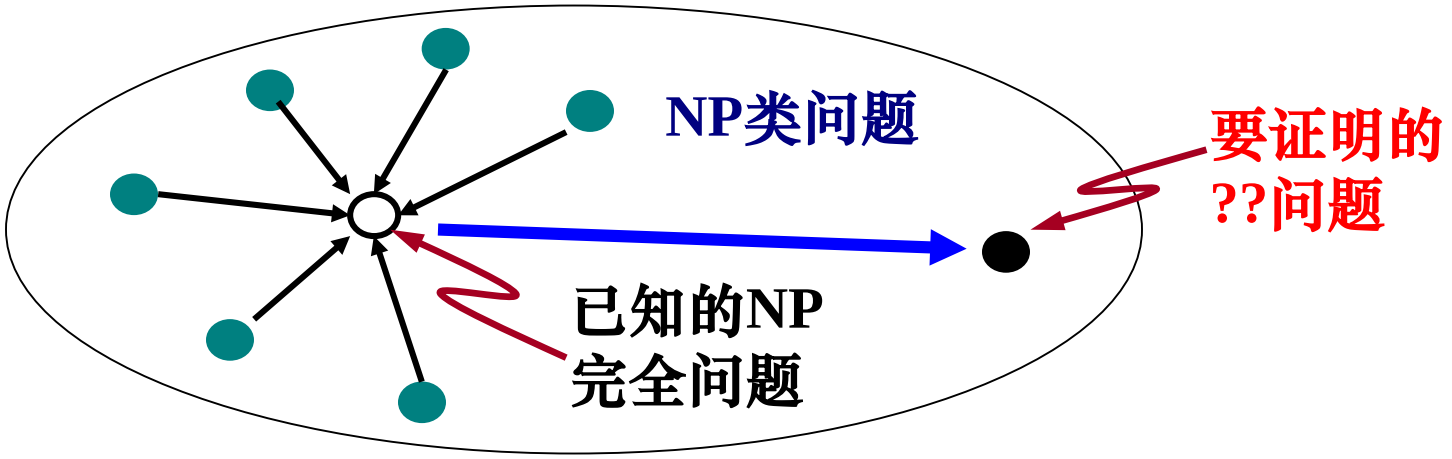
  最长路径问题、 扫雷游戏…

如何证明一个问题是NP完全的？

- 根据NP完全问题的定义（满足的两个性质），显然地，证明需要分两个步骤进行：

① 证明问题Q是NP类问题；即可以在多项式时间内以确定性算法验证一个任意生成的串，以确定它是否为问题的一个解。

② 证明NP类问题中的每一个问题都能在多项式时间内变换为问题Q。由于多项式问题变换具有传递性，所以，只需证明一个已知的NP完全问题能够在多项式时间内变换为问题Q.



证明一个问题是NP完全的 --以最大团问题为例

- **团的概念**：团是图G=(V,E)的一个完全子图，该子图（有k个顶点）中任意两个顶点都有1条边相连。
- **团问题**：对于给定的无向图G=(V,E)和正整数k，是否存在具有k个顶点的团?
- 下面证明团问题属于NP完全问题，证明分两步：

（1）团问题属于NP类问题：显然，验证图G的一个子图是否构成团只需要多项式时间，所以团问题属于NP类问题。

（2）SAT问题$\propto_{poly}$团问题

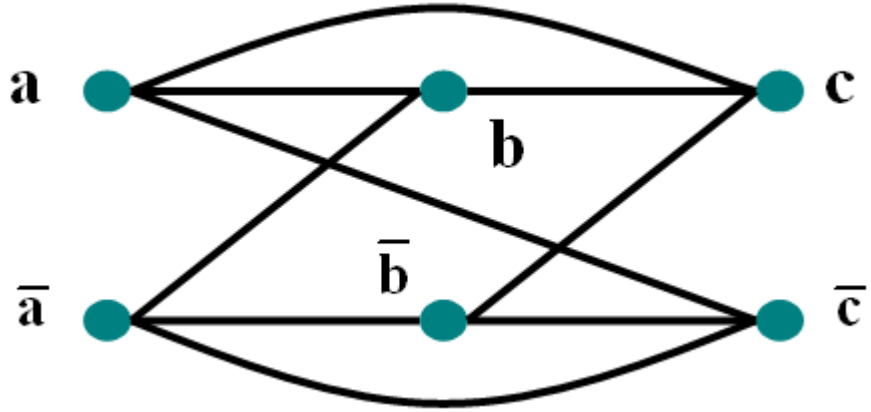
SAT问题$\propto_{poly}$团问题

对于任意一个合取范式，按照如下方式构造相应的图G：

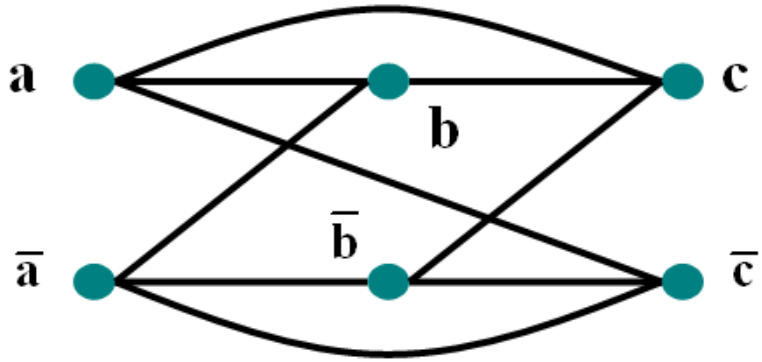
例如 $f=(a\vee\bar{b})\wedge(b\vee\bar{c})\wedge(c\vee\bar{a})$

① 图G的每个顶点对应于*f*中的每个文字，多次出现的重复表示；

② 若图G中两个顶点对应的文字不互补，且不出现在同一子句中，则将其连线。（a-b连线意味着a和b同时为真）



SAT问题$\propto_{poly}$团问题



设 f 有n个子句，则 f 可满足

- ⇔ n个子句同时为真
- ⇔每个子句至少1个文字为真 ⟸ 同时为真，则相连
- ⇔G中有n个顶点之间彼此相连
- ⇔G中有n个顶点的团

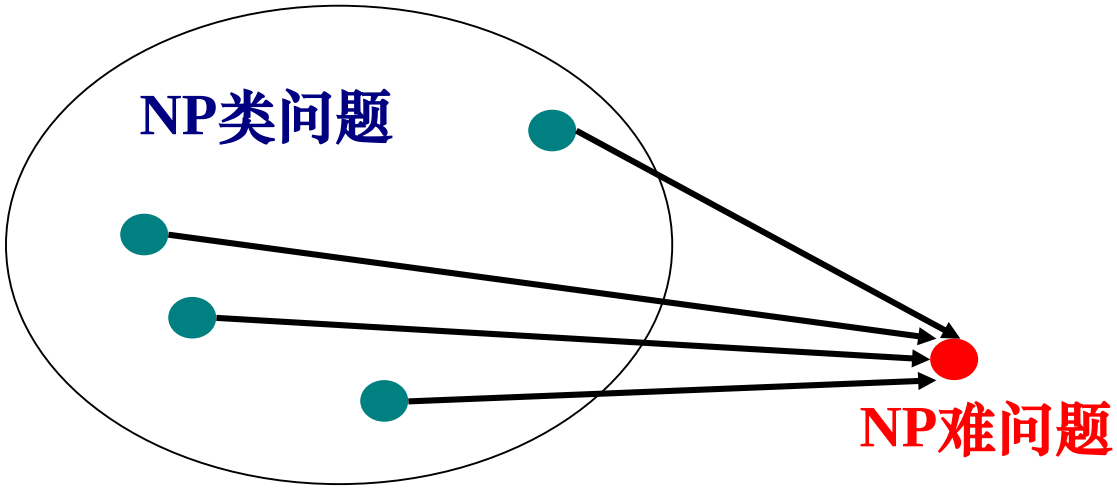
显然，上述构造图G的方法可在多项式时间内完成，

故有：SAT$\propto_{poly}$团问题。

由以上证明可知，团问题是NP完全问题。

NP难问题

难解问题中还有一类问题，虽然也能证明所有的NP类问题可以在多项式时间内变换到问题Q，但并不能证明Q也是NP类问题，所以Q不是NP完全的。但问题Q至少与任意NP类问题有同样的难度，这样的问题称为NP难问题。



co-NP类问题

co-NP类由它们的补属于NP类的那些问题组成。例如：

- 旅行商问题的补：给出n个城市和它们之间的距离，不存在长度为k或更少的任何旅程，情况是那样吗?
- 可满足性问题(SAT)的补：给出一个公式f，不存在使得f为真的布尔变量指派，是吗? 换言之，f是不可满足的吗?

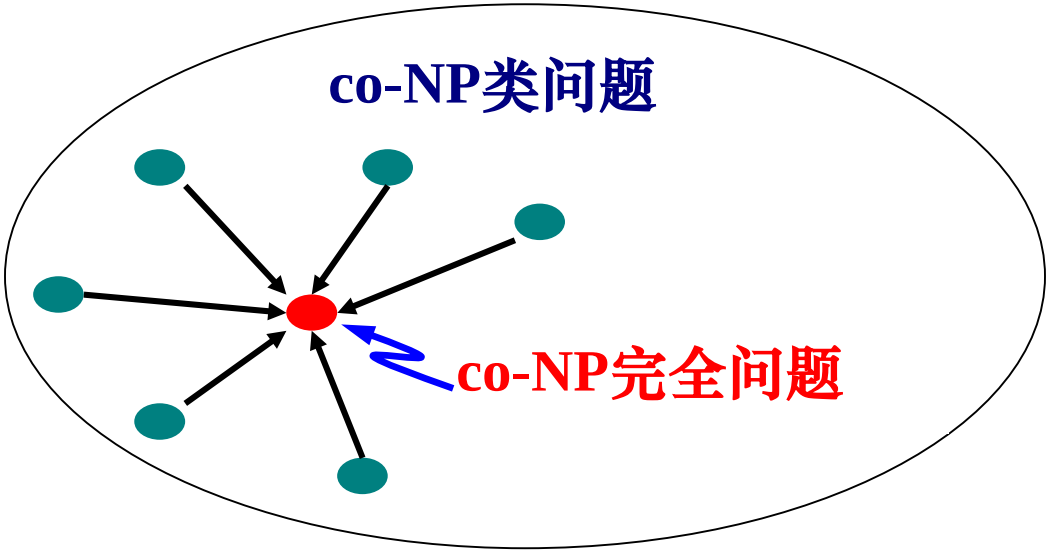
换个角度来看， co-NP类问题也是NP完全问题。

co-NP类问题的定义

co-NP完全问题: 问题Q对于co-NP类是完全的，若

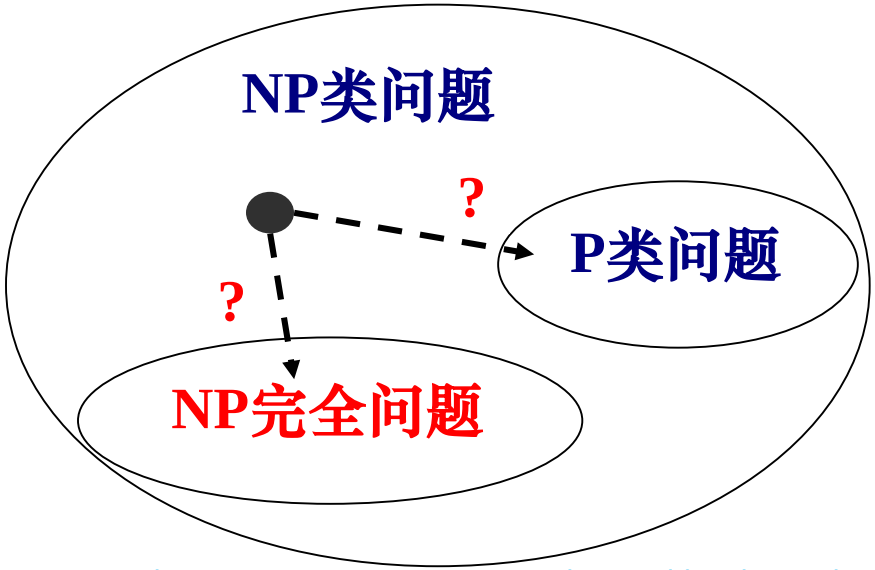
① Q在co-NP中；

② 对于co-NP中的每个问题Q1，有$Q1 \propto_{poly} Q$。



NPI类问题

- NP类问题中还有一些问题，人们不知道是属于P类还是属于NP完全问题，还有待于证明其归属。
- 这些问题是NP完全问题的可能性非常小，也因为不相信他们在P中，我们人为地增加另一问题类来接纳这类问题，这个类称为NPI(NP-Intermediate)类。

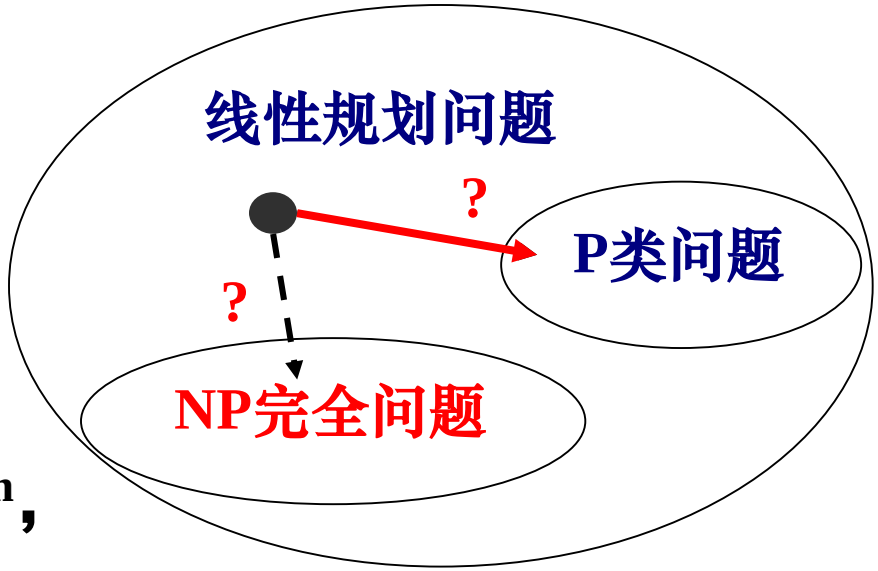


关于NPI类问题的评述

- NPI类是一个人为定义的、动态的概念，随着人们对问题研究的深入，许多NPI类问题逐渐被明白无误地证明他们原本属于P类问题或NP完全问题。

- 例如：线性规划问题、素数判定问题等，在二者没有被证明他们均属于P类问题之前，人们一直将他们归于NPI类问题。

一个例子



- **线性规划问题**

  设$A \in R^{m \times n}$, $x \in R^n$, $b \in R^m$,

  在满足$Ax=b$，$x \geq 0$约束下，

  使目标函数$c^T x$达到最大值，其中$c \in R^n$.

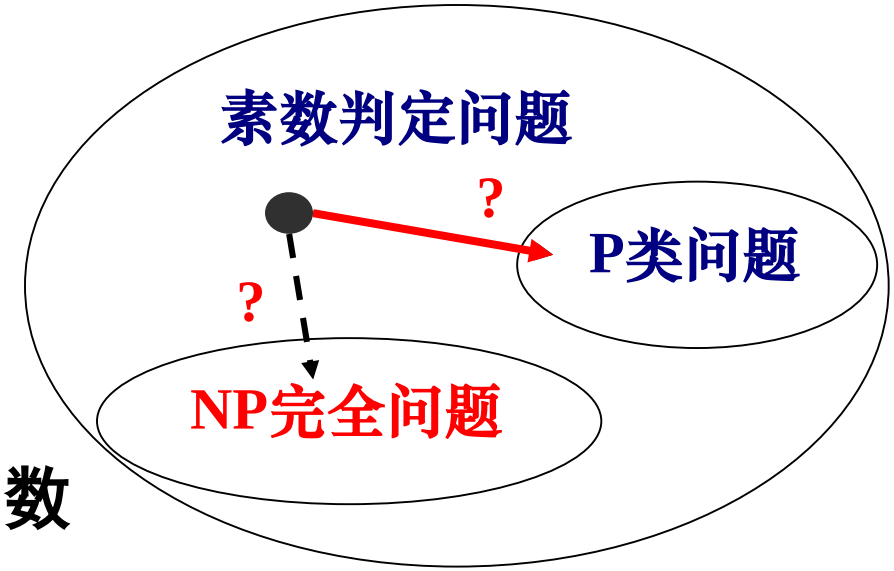
- 长期以来，线性规划问题没有多项式时间解法，也无法证明它是NP完全问题。直到20世纪80年代，这个问题得到解决，发现了多项式时间算法。

1972年，美国数学家 *Klee 和 Minty* 构造了一个著名例子，证明了单纯形算法不是多项式时间算法，而是指数时间算法.

1979年，苏联数学家*Khachiyan*提出了一个多项式时间算法——椭球算法. 证明了$LP \in P$.

1984年，美国贝尔实验室28岁的印度人提出了以他自己名字命名的 *Karmarkar* 算法.

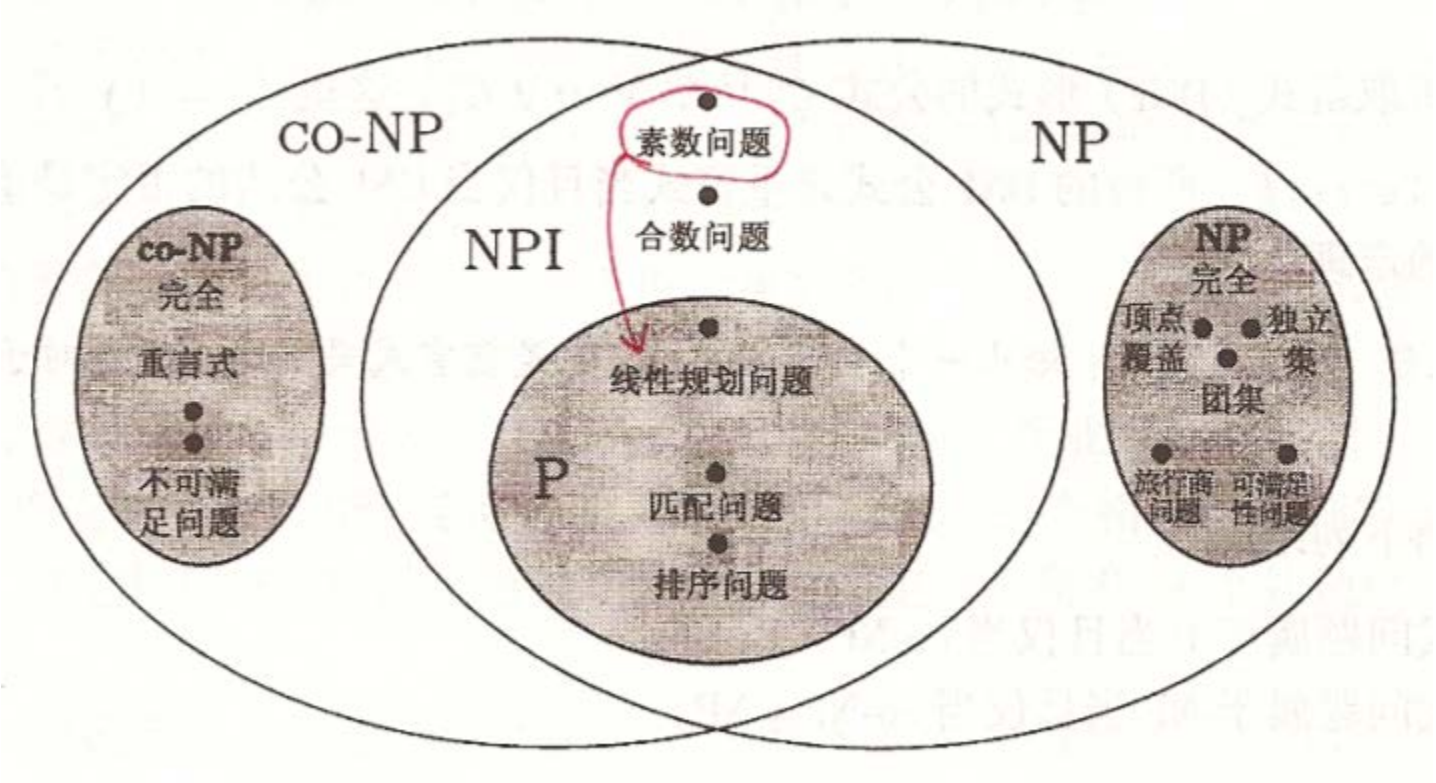
另一个例子



素数判定问题

- 给一个整数，判定其是素数还是合数。
- 经过一个重要的理论突破，印度M. Agrawal 教授和他的学生N. Kayal 和N. Saxena于2002年宣布发现一个多项式时间算法。 (PRIMES is in P, *Annals of Mathematics*, 160 (2004), 781–793)

P、NP、co-NP、NPI类之间的区别与联系



NP完全问题的计算机处理技术简介

NP完全问题是计算机难以处理的，但是实际中经常遇 到，我们无法回避这些问题。因此，人们提出了解决NP完全问题的各种方法：

- **采用先进的算法设计技术：**问题规模不是很大时，采用动态规划法、回溯法、分枝限界法等。
- **近似算法**：很多问题的解允许有一定的误差，只要给出的解是合理的、可接受的。

NP完全问题的计算机处理技术简介

- **随机算法**

① 例如随机采样算法，穷举+挑一 vs. 随机化采样，Θ(n)

② 或以较小的错误概率为代价，获得计算时间的大幅减少。

- **并行计算**

利用多台CPU共同完成一项计算，虽然从原理上说，增强计算性能并不能从根本上解决NP完全问题，但这也是解决NP完全问题的措施之一。近年来的成就如129位(bit)大整数的分解、“深蓝”下棋程序等。

- **智能算法**

① 遗传算法

② 人工神经网络

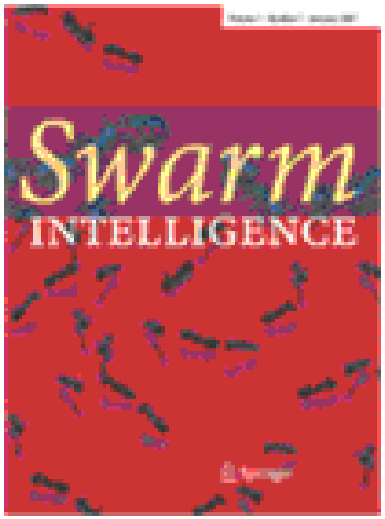
③ 模拟退火算法

④ 禁忌搜索算法

⑤ DNA计算

⑥ 人工免疫算法

⑦ **蚁群算法**



Swarm Intelligence: a new journal dedicated to reporting on developments in the discipline of swarm intelligence. Published by Springer. **Editor-in-Chief: Marco Dorigo.**

（蚁群算法小游戏）http://www.atlas-zone.com/complex/temp/ant/ant.htm