组合优化理论

主讲人：陈安龙

自我介绍

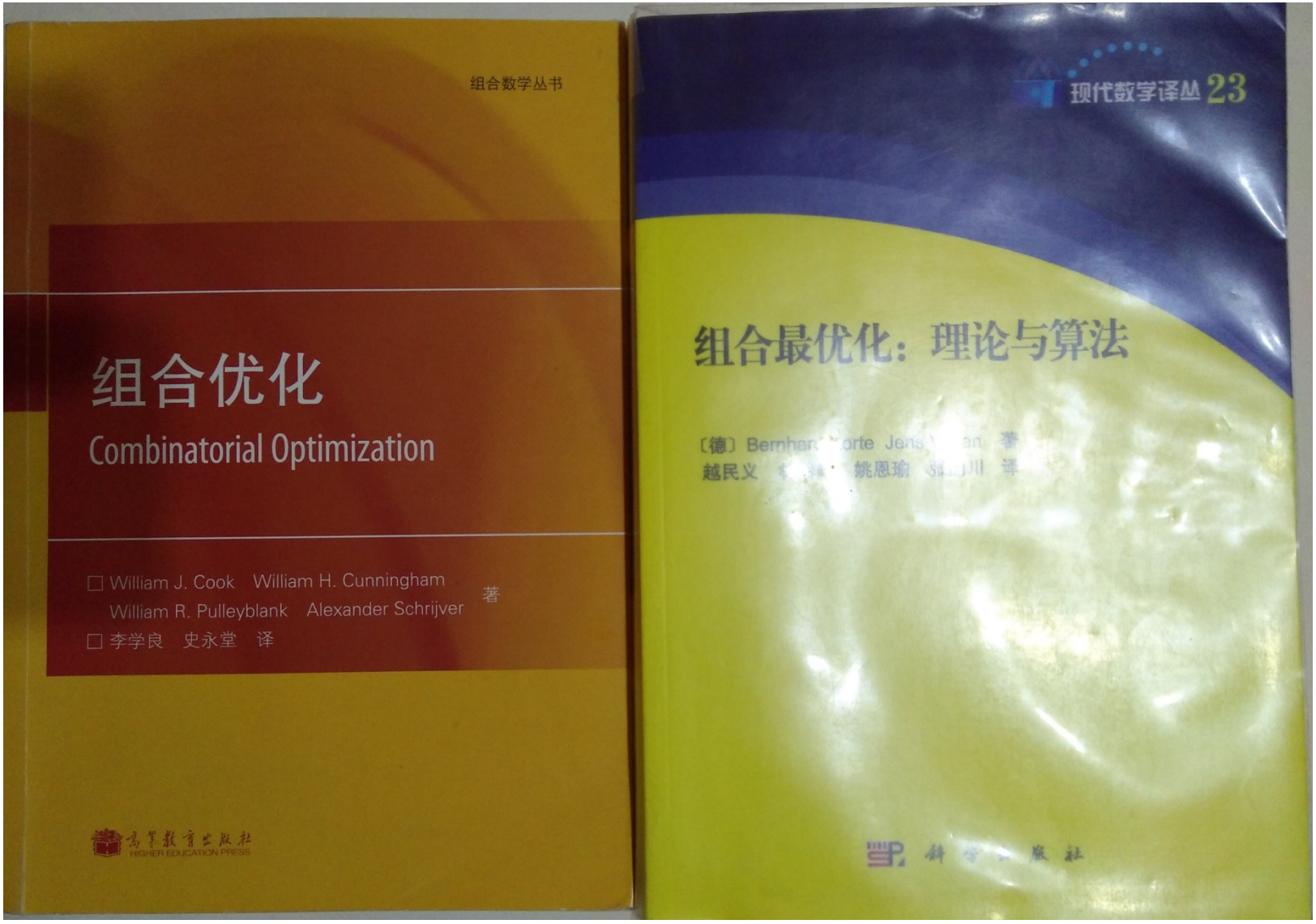
- 姓　　名：陈安龙
- 研究方向：数据挖掘及机器学习
- 电　　话：13688434459
- 课程QQ群：260472637
- e-Mail: chenanlong@uestc.com

群名称：2017组合优化课程班
群　号：260472637

教材及参考书

- 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
- 最优化理论与算法 （第2版） 陈宝林，清华大学出版社
- 网络最优化，谢政，科学出版社，2015.05

教材及参考书



教材及参考书

该课程的相关课程

- 组合数学
- 离散数学
- 计算复杂性理论
- 算法分析与设计

课时数及考试

- 讲授40学时
- 考核方式：开卷
- 成绩评定：平时成绩（作业及出勤）占30%，期末考试成绩占70%。

课程的主要讲授内容

- 线性规划---单纯形法、对偶理论、整数规划
- 网络流---最大流、最小生成树、最短路径
- 最优指派问题
- 装箱问题
- 背包问题
- 作业排序问题
- 旅行售货商问题
- 启发式算法---遗传算法、退火算法

重点讲授上述问题的数学描述和算法的理论基础.

组合优化 – 定义

所谓组合(最)优化(Combinatorial Optimization)又称离散优化（Discrete Optimization），它是通过数学方法去寻找离散事件的最优编排、分组、次序或筛选等. 这类问题可用数学模型描述为：

$$\min \quad f(x)$$

$$s.t.\ g(x) \geq 0, \text{或者} g(x) \leq 0$$

$$x \in D$$

其中，$f(x)$为目标函数， $g(x)$为约束函数，x为决策变量， D表示有限个点组成的集合.

组合优化－定义（续）

一个组合最优化问题可用三参数(D, F, f)表示，其中D表示决策变量的定义域，F 表示可行解集合$F=\{x \mid x\in D, g(x)\geq 0\}$，$F$ 中的任何一个元素称为该问题的可行解，f 表示目标函数．满足

$$f(x^*)=\min\{f(x)\mid x\in F\}\text{或}f(x^*)=\max\{f(x)\mid x\in F\}$$

的可行解x^* 称为该问题的 最优解．组合最优化的特点是可行解集合为有限 点集.

可行解是满足某线性规划所有的约束条件的任意一组决策变量的取值

组合优化 – 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 – 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 – 示例2(续)

$$\min \sum_{i \neq j} d_{ij} x_{ij} \tag{1.4}$$

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

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

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = n \tag{1.7}$$

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

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 – 示例2(续)

(1.5)和(1.6)表示每个城市经过一次．仅有(1.5)和(1.6)的约束无法避免回路的产生，(1.7)约束商人在任何一个城市子集中不形成回路．此时 $D=\{0,1\}^{n\times(n-1)}$.

TSP 问题解的另一种表示法：给定图 G=(V,E)和费用 $c_{ij}>0$ 且$(V_i,V_j)\in E$，求解包含每一个点仅一次的圈，并使得圈上所有边的费用之和最小。

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

数学模型为：$\min L=\sum_{j=1}^{n} d_{i_j i_{j+1}}$ (记 $i_{n+1}=i_1$)

组合优化问题的特点

$$\min \quad f(x)$$

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

$$x \in D$$

- 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。
- 然而，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
- 各种组合优化问题拥有什么性质？数学规律、定理
- 为了构造快速解法，什么样的性质是有用的？筛选有效的数学规律、定理

求解方法分类

- 从求解方法的大的分类上，可以分为：
- 最优化的求解方法
  - 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，通常采用的是近似求解方法。
- 近似的求解方法
  - ① 以**确保解的精度**为目标的方法
  - ② 以**尽可能获得更好解**为目标的方法

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

$$f(I, \sigma^*) \leq f(I, \sigma) \text{ 或者 } f(I, \sigma^*) \geq f(I, \sigma)$$

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是一个问题，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

因为一些组合最优化问题还**没有找到求最优解的多项式时间算法**，而这些组合最优化问题又有非常强的实际应用背景，人们才不得不尝试用一些并不一定可以求解到最优解的算法，在此称为**启发式算法**，来求解组合最优化问题。

启发式算法是相对于最优算法提出的．一个问题的**最优算法**求得该问题每个实例的**最优解**．启发式算法可以这样定义：

一个基于直观或经验构造的算法，在可接受的花费（指计算时间，占用空间等）下给出待解决组合优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度不一定事先可以预计．

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

对于组合优化问题(D, F, f)，D上的一个映射：$N: S \in D \rightarrow N(S) \in 2^D$称为一个**邻域映射**，其中$2^D$表示$D$的所有子集组成的集合，$N(S)$称为$S$的邻域，$S' \in N(S)$称为$S$的一个**邻居**.

邻域例子

例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：

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

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问题解的另一种表示法为

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

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）

第一循环： $N(x^{best})$={(ABCDE),(ACBDE) (ADCBE), (AECDB),(ABDCE),(ABEDC), (ABCED)}，对应目标函数值为：$f(x)$={45,43, 45,60, 60,59,44}. $x^{best} := x^{now}$=(ACBDE）

第二循环： $N(x^{best})$={(ACBDE),(ABCDE) (ADBCE), (AEBDC),(ACDBE),(ACEDB), (ACBED)},对应目标函数值为: $f(x)$={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**.

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：

蒙特卡罗算法---TSP求解示例

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

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

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.

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）$R_H = a$

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 $R_H \geq a$;

界

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中央处理器（*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一个界.

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 $Z_H(I)-Z_{\mathrm{UB}}(I)$

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 $D=\sum_{I\in S}(Z_H(I)-\overline{\mathrm{H}})^2$

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

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： $O(n!)$、$O(3^{n})$

计算复杂性问题

多项式时间算法 —— 好算法 有效算法

指数时间算法 —— 坏算法 恶劣算法

Note: *A* 算法的计算复杂性为 $O(n^{80})$，*A* 是好算法吗？ 与 $O(2^n)$ 比较

问题 *D* 是否有多项式时间算法是问题的固有性质.

伪多项式时间算法：它的时间复杂性不超过关于实例输入长度和实例中最大数的某个二元多项式 .

如：$O(m\,n^{2})$.

P 类、*NP* 类、*NP*完全问题

复杂性分析的一个研究方向：对组合优化问题归类

P 类问题：（*Polynomial*）

对组合优化问题π，如果存在一个求解该问题的多项式时间算法，则称π是多项式时间可解问题．所有多项式时间可解问题构成的集合，称为 *P* 类问题

记为 P．$\pi \in P$

判定问题： 如果一个问题π的输出（即答案）为 *Yes or No*，则称问题π为判定问题 .

Example 8 可满足性问题（*SATisfiability*）

已知一组逻辑变量的一个布尔表达式，其中每个逻辑变量的取值为 1 (真) 或 0 (假). 问是否存在该组逻辑变量的取值，使得布尔表达式取值为真？

$$(x_1 + \overline{x_2}) \cdot (x_2 + \overline{x_3}) \cdot (\overline{x_2} + \overline{x_3})$$

Example 9 （*TSP*）已知完全图 G 上每一条边（v_i,v_j）的权w_{ij}，及常数 h 。问是否存在一个 H - 圈 C，使得

$$\sum_{(v_i,v_j)\in C} w_{ij} \le h$$

Example 10 （*LP*）给定常数 z ，是否有

$$\{x \mid c^T x \le z, Ax = b, x \ge 0\} \ne \Phi$$

显然，判定问题不比原优化问题更难。可以证明两个问题在计算复杂性意义下是等价的。

NP 类问题：（*Nondeterministic Polynomial*）

给定一个判定问题 *D*，如果存在一个算法，对 *D* 的任何一个答案为“是”的实例，它给出的回答均可经多项式时间界的运算加以检验，则称 *D* 为一个 *NP* 问题.

由全体 *NP* 问题构成的集合，称为 *NP* 类问题.

记为 *NP*

$$SAT、\ TSP、\ LP \in NP$$

显然 $\mathbf{P \subseteq NP}$

但是，多项式时间可验证性不能保证多项式时间可解性

$TSP \notin P$

21世纪7大数学难题之首

问： $P \supsetneq NP$

Theorem 1 如果问题 $\pi \in NP$ ，那么存在一个多项式 p 使得 π 能用时间复杂性为 $O(2^{p(n)})$ 的确定型算法求解，这里 n 表示 π 的实例的输入长度 .

在对 $P \stackrel{?}{=} NP$ 的研究中，1972年 加拿大数学家 *Cook* 提供了一个漂亮的理论，把前人的失败归结为一个深刻的数学猜想，提出了存在一类称之为 *NP – complete* 类的问题。

Definition 3 设 D_1，D_2 为两个判定问题，若求解 D_2 的算法 A_2 可多项式次地调用求解 D_1 的算法 A_1 而实现，则称问题 D_2 可以多项式时间归约为 D_1。

简记：$D1 \propto D2$

Theorem 2 如果 $D_2 \propto D_1, D_1 \in P$，则 $D_2 \in P$。

定理 1 说明如果 $D_2 \propto D_1$，则 D_1 不会比 D_2 易解（在计算复杂性意义下）.

多项式的四则运算及复合，仍是多项式

Definition 4 一个判定问题 Q 被称为是 ***NP – complete*** 问题，如果

1、$Q \in NP$

2、任何 *NP* 问题均可多项式时间归约为 Q .

SAT $\in$ ***NP-c***

已证有4000余个问题

这是*Cook*首先的证明结果

TSP、***IP*** $\in$ ***NP-c***

NP-c 类问题有如下十分有趣的性质：

1、任何 *NP-c* 问题都不能用任何已知的多项式时间算法求解；

2、任何一个 *NP-c* 问题有多项式时间算法，则一切 *NP-c* 问题都有多项式时间算法 .

Theorem 3 如果 π_1 是 $NP-c$ 问题, $\pi_2 \in NP$，且 π_2 可以多项式时间归约到 π_1 ，则 π_2 是 $NP-c$ 问题 .

Theorem 4 $P = NP$ 当且仅当存在一个 $NP-c$ 问题有多项式时间算法 .

对一个新的组合优化问题 E，在找不到解它的有效算法时，可去证明 $E \in NP\text{-}c$.

主要证明 *NP-c* 某一问题可多项式时间归约为 *E*

如果成立，则有如下常用方法：

1、寻找尽可能快的算法求解;(如：分枝定界法、单纯形法)

2、在给定假设下，对具体问题提出解的有效算法;

3、提出启发式算法（最好是近似算法），快速地得到满意解。（如：贪婪算法、遗传算法）

Definition 5 如果某优化问题 π 的判定问题是 $NP-c$，则称 π 是 $NP-$难 *(hard)* 问题 .

单纯形法是否是有效算法？

$LP \overset{?}{\in} P$

答案：是否定的

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.

$$Max\ z = x_n$$

$$s.t.\quad \varepsilon \le x_1 \le 1$$

$$\varepsilon x_{j-1} \le x_j \le 1 - \varepsilon x_{j-1}$$

$$(j = 2, 3, \ldots, n)$$

其中ε是充分小的正数.

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.

***Note*：** 计算复杂性是以最坏的实例来评价一个算法

通过概率分析方法，发现单纯形法的平均迭代次数为$O(n^{1.5}\sim n^2)$.所以，它的实用和有效在*LP*的计算和应用中，几乎占据了绝对优势.