现代优化(计算)方法

清华大学数学科学系
邢文训
wxing@tsinghua.edu.cn
Tel: 62787945
Office：理科楼A416
答疑：周四下午4：00-5：00

课程的主要内容、要求和安排

本课程的主要内容

- **最优化问题及算法复杂性(4周)**
  - **组合最优化的复杂性概念（3.5周）**
  - **连续优化的复杂性概念（0.5周）**
- **智能(Metaheuristics)算法（4周）**
  - **禁忌搜索(tabu search)算法**
  - **模拟退火(simulated annealing)算法**
  - **遗传算法(genetic algorithms)**
  - **人工神经网络 (artificial neural networks) 算法**

- **线性锥优化理论与应用(4周）**
  - 基本概念
  - 凸优化基本理论和对偶理论
  - LP, SOCP, SDP 模型及应用
  - 线性锥优化应用案例
- **凸优化算法简介（2周+）**
  - 次梯度方法
  - 邻近点方法

基本要求

- 二次笔试（60%）
  - 第一次：算法及其复杂性，第6-7周考试。（占30%）
  - 第二次：凸优化理论和线性锥优化，第16周。（占30%）。
- 2个报告（30%）
  - 智能算法文献综述(6-8周)，算法论文（算法实现，9-12周）。
- 考勤及其他(10%)：包括随机考勤5次、答疑或课堂互动等。
- 鼓励：相关达到投稿水平论文（适当加分）。

主要参考书目

- 《现代优化计算方法》（第二版），邢文训、谢金星，清华大学出版社，2005。
- 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。
- 《线性锥优化导论》, 邢文训、方述诚, 清华出版社，2020。
- 凸优化算法，D. P. Bertsekas著，清华大学出版社，2016.

参考书目及学术网站

- Reeves C R(Ed). Modern heuristic techniques for combinatorial problems. Oxford: Blackwell Scientific Publications, 1993
- European Chapter on Metaheuristics(EU/ME)
  - http://www.euro-online.org/eume/
- Convex Optimization, Stephen Boyd, Lieven Vandenberghe, Cambridge University Press, 2004, Sixth printing with corrections 2008.
- 一些课后的阅读文献。

本课程主要检索词

- Metaheuristic, meta-heuristic
- Tabu (taboo) search
- Simulated annealing
- Genetic algorithms
- Artificial neural networks
- Linear programming, second-order conic programming, semi-definite programming, Lagrangian relaxation, quadratic programming, convex optimization， linear conic programming.
- Subgradient methods, proximal algorithms etc.

主要内容

- 介绍性课程，以思想和应用为主。
- 组合最优化
  - 复杂性理论（重点考核）
  - 一些近似算法——智能算法
- 连续优化
  - 凸优化与线性锥优化
  - 模型、思想及简单理论
  - 建模过程和应用（重点考核）

开设本课程的想法

- 人工智能的轰动效应。
- 智能算法应用比较广泛，数学知识要求少，易实现，在大数据离线算法应用中有较大的市场。
- 凸优化理论是一类应用问题的数学基础，难度较大，是锥优化、凸/非凸优化问题研究的基础。
- 锥优化及凸优化算法。大数据问题的处理。

开设本课程的想法

- 介绍性课程。
- 只讲思想、介绍理论而不系统给出理论。
- 每一个方面都可单独形成一门课程，学习者比较辛苦，课后看的材料多于课堂内容。
- 突出算法复杂性理论和锥优化建模。
- 要求比较高，需要有思想准备。

优化的名词解释及发展历史

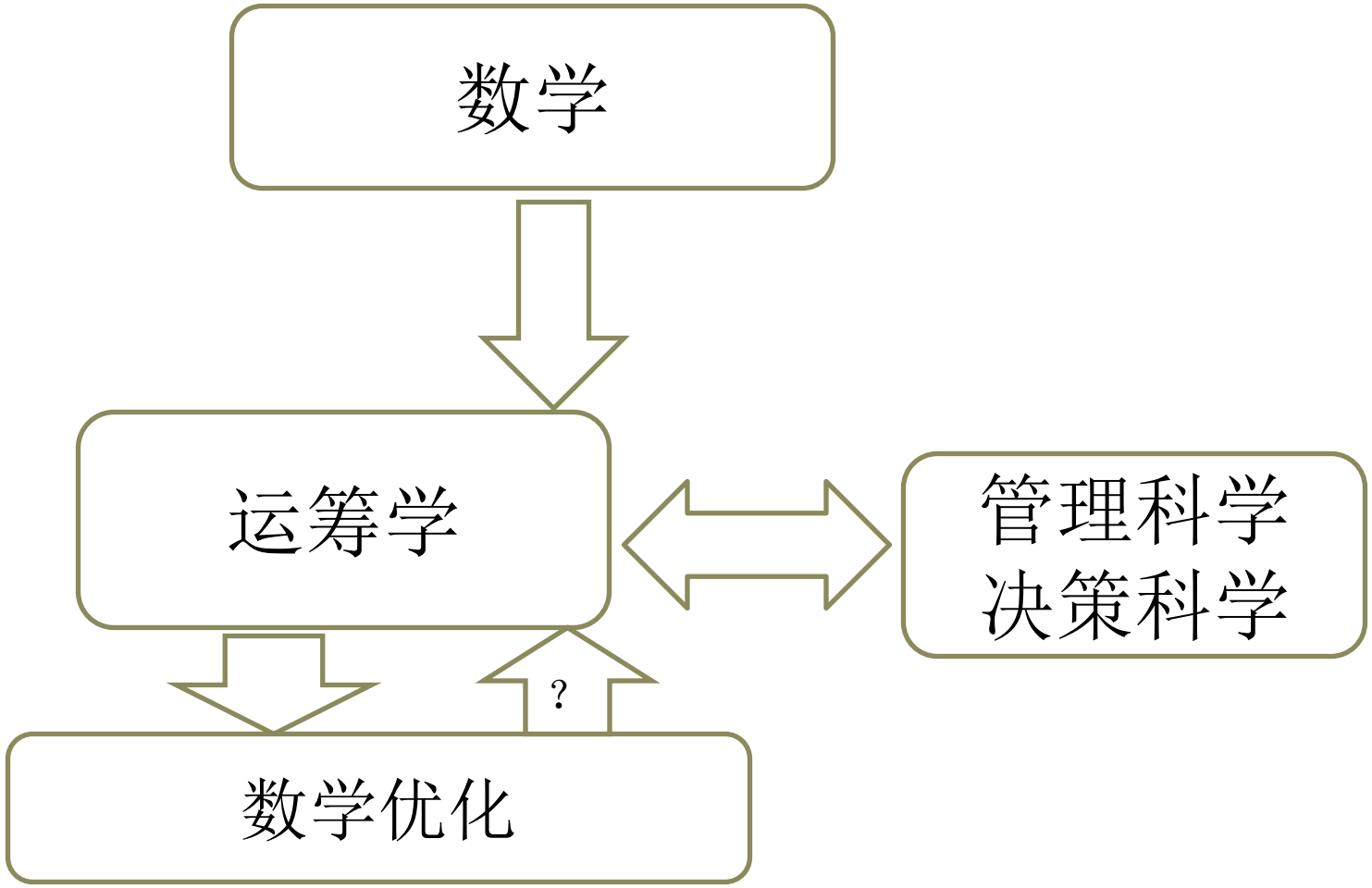
From Wikipedia, the free encyclopedia

- **Operations research**, or **operational research** in British usage, is a discipline that deals with the application of advanced analytical methods to help make better decisions.
- It is often considered to be a sub-field of mathematics.
- The terms **management science** and **decision science** are sometimes used as more modern-sounding synonyms.

- It employs techniques from other mathematical sciences, such as mathematical modeling, statistical analysis, and mathematical optimization.
- It arrives at optimal or near-optimal solutions to complex decision-making problems.
- It has overlap with other disciplines, notably industrial engineering and operations management, and draws on psychology(心理学) and organization science(组织理论学).

我的理解

- 运筹学和数学优化的等价。
  - 运筹学——问题出发。
  - 数学优化——突出数学工具的重要性。
- 它们都是用以数学为基础的工具，为决策提供方案。
- 运筹学者是决策的方案的提供人，只提供方案，但不是决策人。


数学
运筹学
管理科学
决策科学
?
数学优化

名词解释

- 数学优化(mathematical optimization) =数学规划(mathematical programming)：运筹学的基础。
- 运筹学——operations research（美国），operational research（欧洲）。 操作层面理解。
- 数学优化，数学规划——mathematical optimization。突出数学工具。

- It is often concerned with determining the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective.
- Originating in military efforts before World War II, its techniques have grown to concern problems in a variety of industries.

- In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

名词解释

- 最优化——Optimization
- Operational Research——英国，欧洲
- Operations Research——美国
- 运筹学——《汉书•高祖本纪》：“运筹帷幄之中，决胜千里之外”。
  - （班固（公元32-92）成书公元80年前后，记载公元前200年前后汉高祖刘邦统治历史。）
- 运筹学——作业研究（台湾、香港、新加坡）

发展历史

- 第二次世界大战期间（1940s），game theory
- 二战英国与德国空战博弈。1934德国建立空中力量，英国空中力量较弱。英国空军轰炸机具有一定规模，战斗机力量较弱。
- 希望设计dead ray（电磁波）或预警系统，负责人：Sir Henry Tizard，1935年否定了dead ray，开发雷达。

- 1938年：A.P. Rowe, 防空雷达系统与战斗机配合的作业研究 。成立Operational Research机构。
- 1940 The Battle of Britain.
  - May 14, Dowding得知：法国总理向丘吉尔要增加10个飞行中队到法国，丘吉尔基本接受，15日他将参加战争内阁会议。
  - 一个小组：Harold Larnder and E.C. Wlliams给出一些数据和建议：不派。
  - 最后没派！

参考文献：Larnder H, The origin of operational research. Operations Research, 32:465:475, 1984,

- 1940’s美国， linear programming
  - Dantzig领导一个Operations Research小组，研究军队供给问题。
  - 模型：Linear programming
  - 算法：Simplex algorithm(表上作业法)。

- 1947年
  - George B. Dantzig提出线性规划(linear programming), (U.S. Department of the Air Force)及单纯形算法
- 1970s
  - 计算复杂性理论的出现，S. A. Cook(1971)的主要工作。M.R. Garey and D. S. Johnson(1979)总结工作。

M.R. Garey and D. S. Johnson , Computers and intractability: a guide to the theory of NP-completeness, 1979.

- 猜想（假设）：P≠NP (21世纪7大数学难题之一，100万美元）

最优算法（精确算法）和启发式算法

- 精确算法：问题规模较小，计算出最优解。典型代表：单纯形算法，Branch and bound。
- 启发式算法：一些简单而不一定求到最优解的快速算法无法得到认同，’quick and dirty’。

分支定界的基本思想

- 最小化优化问题。给定一个初始可行解，上界。
- 分支：一个问题分成2个子问题。
- 定界：分别求解两个子问题松弛问题的下界。
- 删除：若某个子问题的下界，比给定的上界还大，删除这一支。
- 重复分支定界。

启发式算法

- 开学回北京，如何将包装进去的物品价值最大？
- 参数：n个物品，价值向量c，尺寸向量a,包容量b。
- 模型：$\max_{\{x\in\{0,1\}^n | a^T x <= b\}} c^T x$
- 贪婪算法：c_i 从大到小装包，直至无法装包。

- 1970s，计算机开始广泛应用，维数爆炸现象，快速反应要求，计算复杂性理论产生。

- 将问题进行分类
  - 简单问题：多项式可解
  - 复杂问题：非多项式可解, NP-complete, NP-hard
- 实际问题求解要求
  - 必须快速求解
- 启发式算法得到认可
  - 比较直观

- 1980s，智能计算方法(metaheuristics)产生。
  - tabu——人工智能，记忆与干预的应用

  - Simulated annealing——金属退火过程模拟
  - Genetic algorithms——生物进化，适者生存
  - Neural networks——大脑神经功能
  - 90s Ant colony optimization algorithms——蚂蚁觅食过程
- 得到重视，被人们广泛应用，每年相关文章过千篇。
- 万能算法——不需要很强的数学基础。

学习目的

- 更多寻求更高效求解问题的算法
- 面临实际问题采用什么样的算法？简单问题还是复杂问题？
- 接受创新的思想，培养创新的能力。
- 掌握解决问题的一些方法。

- Convex Optimization
  - SDP: Semi-Definite Programming, 1980s
  - 1984, N. Karmarkar 提出一个求解线性规划的多项式时间算法。

    N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica 4, 375-395.
- 2006数学家大会1小时报告（20人）
  - Convex programming, Arkadi Nemirovski
    - Advances in convex optimization: conic programming,

- 凸优化算法。特点：用一阶梯度（方向）信息，多次迭代，适用于大规模问题。

- 大数据时代对简单的要求：每一步计算快，每一步占内存小，整体收敛最优解！
- 凸优化的一阶、二阶结构算法、交替计算等：近期研究热点！如
  - ADMM-Alternative Direction Multiplier Methods
  - Proximal point methods.
  - Augmented Lagrangian methods.
- 随机算法.

近期的一个研究热点

- Yang等获[yan2015]2018 The Beale-Orchard-Hays Prize, the 23rd International Symposium for Mathematical Programming, France, July, 2018.
- 郦旭东[li2018]获第六届国际连续优化大会(ICCOPT2019)青年学者最佳论文奖，德国柏林，2019.

[yan2015] Liuqin Yang, Defeng Sun and Kim-Chuan Toh, SDPNAL+: a majorized semismooth Newton-CG augmented Lagarangian method for semidefinite programming with nonnegative constraints, Mathematical Programming Computation 7: 331-366, 2015.

[li2018] Xudong Li, Defeng Sun and Kim-Chuan Toh , A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems, SIAM J. OPTIM. 28:433-458, 2018.

国内状况

- 学会：
  - 中国运筹学会（一级）
  - 省市学会，专业学会（数学规划学会，排序学会等）
- 学报
  - 运筹学学报，运筹与管理，应用数学学报，高校应用数学学报，系统工程理论与实践，*Journal of the Operations Research Society of China*等。

国际刊物

- Operations research, Management Science, Mathematics of Operations Research, European Journal of Operational Research, The Society of Operational Research, Mathematical Programming, Operations Research Letters, Computers and Operations Research, Discrete Applied Mathematics, The International Journal of Game Theory, IIE Transactions, IEEE on sth, SIAM on sth etc.

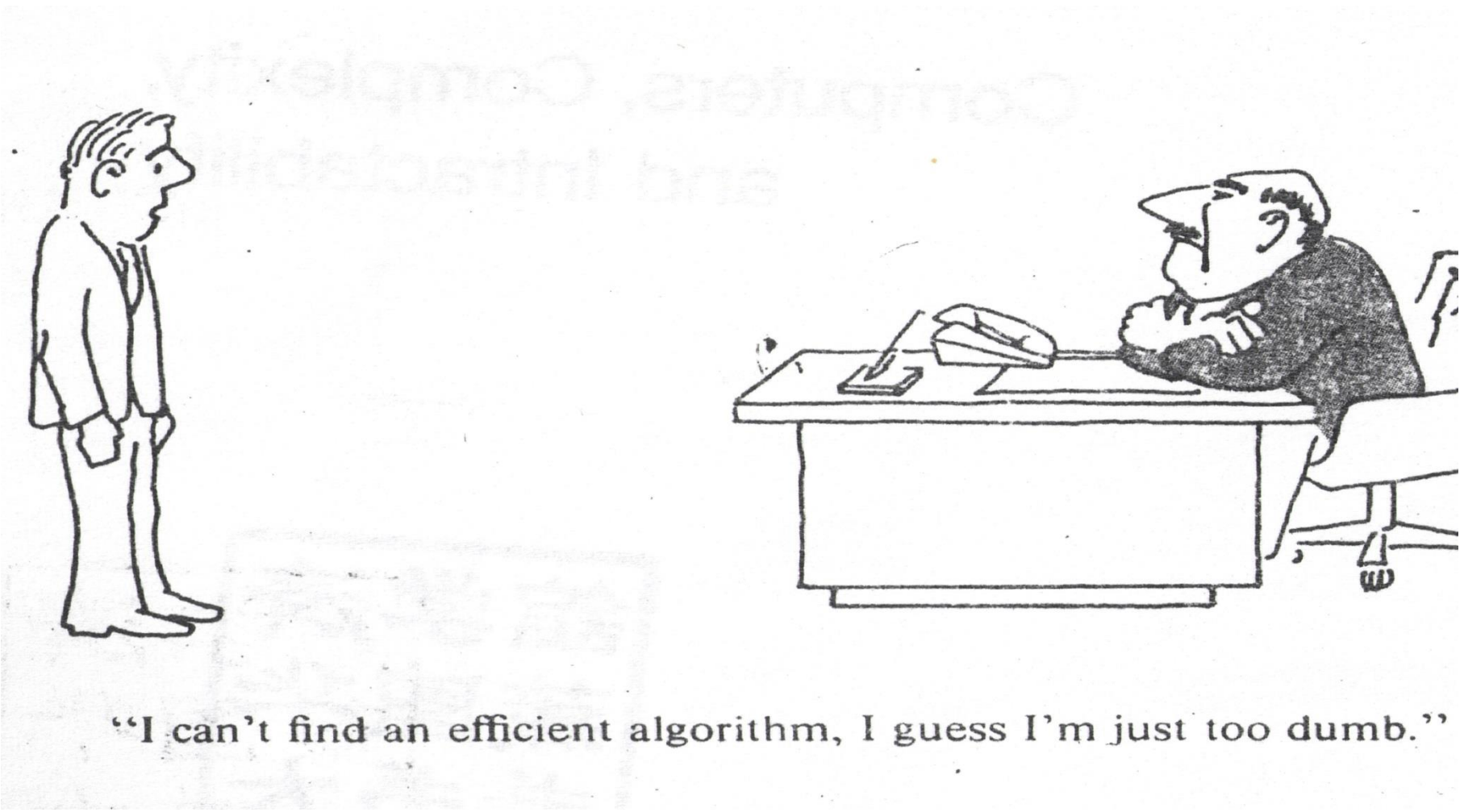
第一章 概论

第一章 概 论

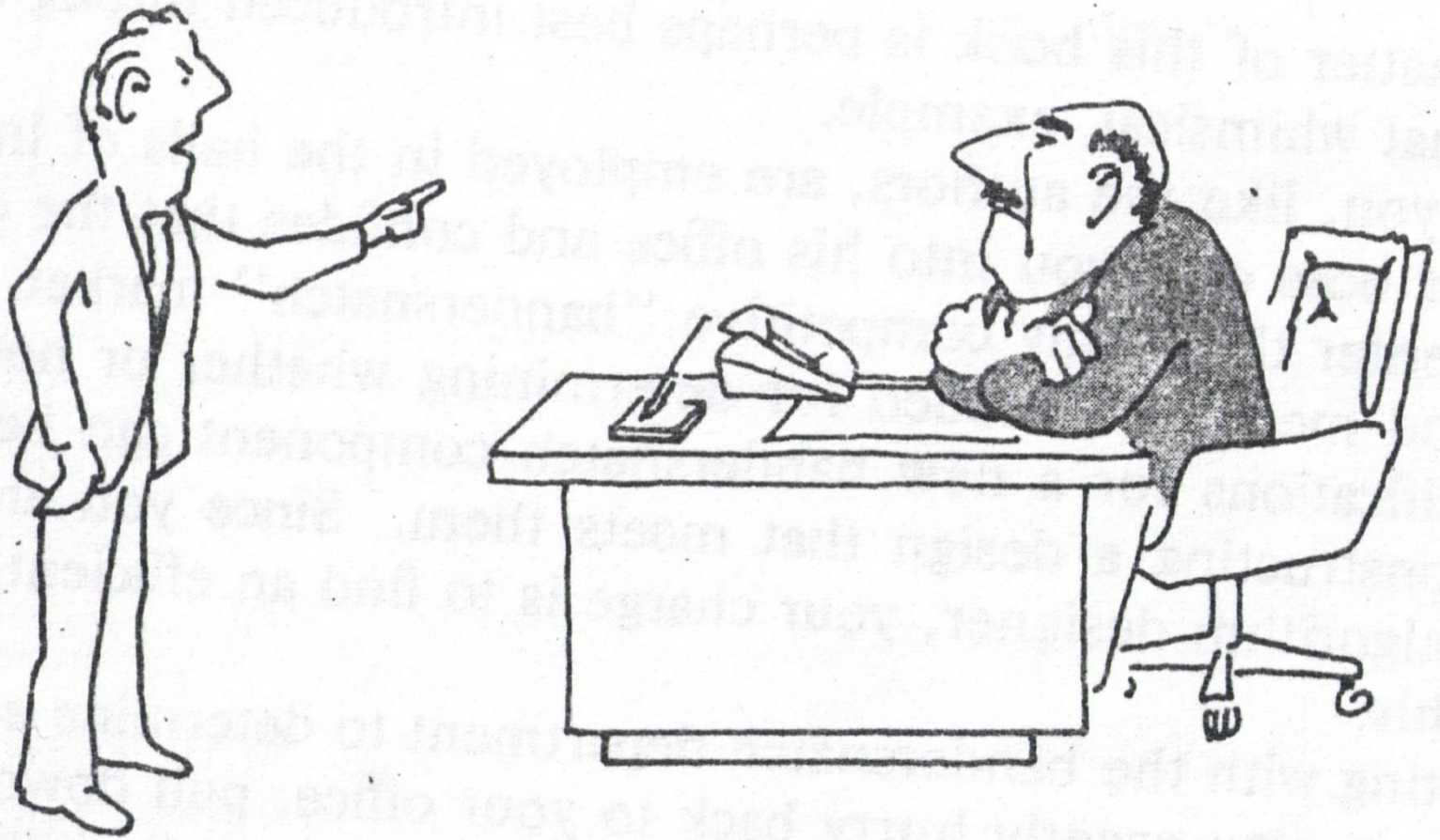
- 组合最优化问题
- 计算复杂性的概念
- 邻域的概念
- 启发式算法
- NP,NP完全和NP难
- 多项式时间迫近格式

When you get a problem, your answer is…

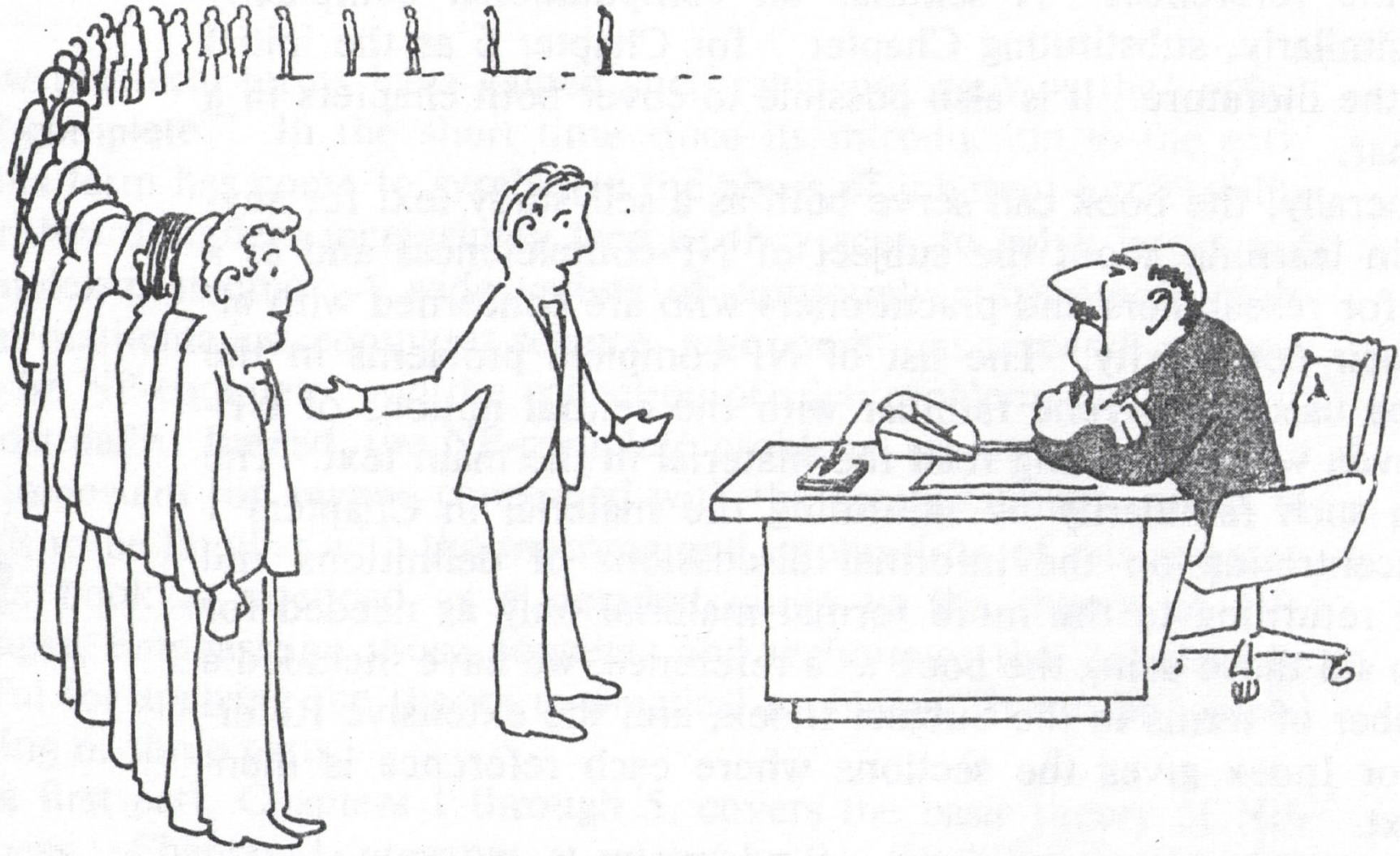
(cited from Garey et al, Computers and Intractability, 1979)



"I can't find an efficient algorithm, I guess I'm just too dumb."



"I can't find an efficient algorithm, because no such algorithm is possil



"I can't find an efficient algorithm, but neither can all these famous people."

1.1 组合最优化问题

组合最优化(combinatorial optimization)是通过数学方法的研究去寻找离散事件的最优编排、分组、次序或筛选等，是运筹学(operations research)中的一个经典和重要的分支，所研究的问题涉及信息技术、经济管理、工业工程、交通运输、通讯网络等诸多领域。

$$\min_{x \in F} \quad f(x)$$

本课程中习惯用三参数（D，F，f）

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t. } & g(x) \geq 0, \\ & x \in D, \end{aligned}$$

D：定义域，离散集合

F：可行域，$F = \{x \mid g(x) \geq 0, x \in D\}$

f：目标函数。标准形式 $\min f(x)$.

0-1背包问题(knapsack problem)

设有一个容积为b的背包，n个尺寸分别为a_i，价值分别为c_i的物品，如何装包以使装入物品的总价值最大？

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

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

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

$$D = \{(x_1, x_2, \cdots, x_n) \mid x_i \in \{0,1\}, \ j = 1,..., n\} = \{0,1\}^n$$

$$F = \{(x_1, x_2, \cdots, x_n) \in D \mid \sum_{i=1}^{n} a_i x_i \le b\} \qquad f(x) = -\sum_{i=1}^{n} c_i x_i$$

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一遍后回到起点且所走路径最短？TSP还可以细分为对称和非对称距离两大类问题。前者称为对称距离TSP，后者为非对称TSP。

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

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

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

$$\sum_{i,j \in s} x_{ij} \leq |s| - 1, \quad 2 \leq |s| \leq n-2, \quad s \subset \{1,2,\ldots,n\},$$

$$x_{ij} \in \{0,1\}, \quad i, j = 1,\ldots,n, \; i \neq j.$$

TSP建模评价

- 所有目标函数，约束都是线性。
- 第三个约束个数：2^n-2-2n。
- 线性的目的便于商业软件求解。

装箱问题(Bin packing)

设有*n*个一维尺寸{d_i，i=1,2,…,n}不超过1的物品集合，如何以个数最少的一维尺寸为1的箱子装进这*n*个物品？该问题为（一维）装箱问题。

$$\min \sum_{j=1}^{n} y_j$$

$$s.t. \ \sum_{i=1}^{n} d_i x_{ij} \le 1, \ j = 1,2,\cdots,n$$

$$x_{ij} \le y_j, i = 1,2,\cdots,n; \ j = 1,2,\cdots,n$$

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

$$x_{ij} \in \{0,1\}, \quad y_j \in \{0,1\}, i, \ j = 1,2,\cdots,n.$$

约束机器排序问题(capacitated machine scheduling)

n个加工量为d_i的产品在一台机器上加工，机器在第t个时段的工作能力为c_t，求出所有产品得以加工的最少时段数。

$$\min T$$

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

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

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

文字描述与数学模型

- 建立数学模型的目的？
- 整数规划模型的建立有助于对问题的理解和采用已有的如分枝定界、割平面等算法求最优解，也利于松弛变量的整数约束条件得到问题的下界或上界。
- 组合优化问题通常可以用整数规划模型的形式表示，但有些组合最优化问题用整数规划模型表示比较复杂和不易被理解，不如问题的直接叙述容易理解。

误区！！！！！！！！！！！

- 由于组合最优化问题的复杂性，很多快速的算法只给出问题的可行解。
- 有大量的组合最优化问题是通过文字语言叙述的，不一定强求给出整数规划模型。
- 负面效应：为了显示数学基础好，而建立数学模型！

1.2 计算复杂性的概念

非对称距离TSP问题：*n*个城市的一个排列表示商人按这个排列顺序推销并返回起点。

算法的维数爆炸问题

固定一个城市为起终点，则需要$(n-1)!$次枚举。以计算机1秒可以完成24个城市所有路径枚举（固定起点后，实际上是余下的23个城市的所有排列）为单位

城市数	24	25	26	27	28	29	30	31
计算时间	1秒	24秒	10分	4.3小时	约4.9天	约136.5天	10.8年	约325年

问题(problem)

- 一个抽象的模型或概念，是需要回答的一般性提问，通常含有若干个满足一定条件的参数。问题通过下面的描述给定：（1）描述所有参数的特性，（2）描述答案满足的条件。
  - 如TSP是一个问题，可由文字或模型表示，该问题的参数是：城市数和城市间的距离；而答案（经过所有城市并回到出发城市的旅行路线）满足的条件是：该旅行路线在所有可能的旅行路线中是最短的。

实例(instance)

- 问题中的参数赋予了具体值的例子。
  - 一个问题通过它的所有实例表现，实例是问题的表现形式，问题是实例的抽象（集合）。
  - TSP的一个实例：n=100，d_{ij}给定，i,j=1,2,…,100。
  - 背包问题的实例：c，a，b都给定。
- 集合论的理解：问题={所有实例}.

算法

- 求解问题的一个计算机方法
- 一个算法针对一个问题来设计的。
- 通过实例而具体实现。
  - 如TSP的枚举算法可以求解任何一个TSP的最优解。若用计算机求解，则必须对问题中的参数赋予具体值，如TSP中的城市数和城市间的距离。

一个数在计算机中存储的长度

- **二进制编码**

正整数：　　$x \in [2^s, 2^{s+1})$

$$x = a_s 2^s + a_{s-1} 2^{s-1} + \cdots + a_1 \times 2 + a_0$$

$$a_s = 1, a_i \in \{0,1\}, i = 0,1,..., s-1$$

编码（s+1位）：　　$(a_s a_{s-1} \cdots a_1 a_0)$

2进制存储的长度的理论估计

由于：
$$\log_2(x+1) \le s+1 \le 1+\log_2 x$$

$$1+\log_2 x-\log_2(x+1)<1, \forall x \ge 1$$

任何一个整数x的输入规模(size)为(包含一个符号位和一个数据分隔位)：

$$l(x)=\lceil \log_2(|x|+1) \rceil+2$$

注：2的位置可以用一个正整数替代！

实例的规模(instance size)

- 实例的规模：存储实例系数所占的计算机空间的大小。
- 在给定区分和符号位的前提下，每个实例的规模唯一确定。
- 为了便于算法复杂性分析，对实例规模进行上下界估计。
- 例：线性规划问题

线性规划实例规模的精确计算

- 线性规划

$$\min \{c^{\mathrm{T}}x: Ax=b, x\geq 0\}$$

- 输入：m, n, c, A, b

$$l(I)=\lceil \log_2(|m|+1)\rceil+\lceil \log_2(|n|+1)\rceil+\sum_{j=1}^{n}\lceil \log_2(|c_j|+1)\rceil$$

$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil \log_2(|a_{ij}|+1)\rceil+\sum_{i=1}^{m}\lceil \log_2(|b_i|+1)\rceil$$

$$+2(mn+m+n+2)$$

上界估计

$$l(I)=\lceil\log_2(|m|+1)\rceil+\lceil\log_2(|n|+1)\rceil+\sum_{j=1}^{n}\lceil\log_2(|c_j|+1)\rceil$$
$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil\log_2(|a_{ij}|+1)\rceil+\sum_{i=1}^{m}\lceil\log_2(|b_i|+1)\rceil$$
$$+2(mn+m+n+2)$$

- 关系式

$$\lceil\log_2(x+1)\rceil\le 1+\log_2 x,\qquad x\ge 1$$

- 上界估计1： $l(I)\le 3(mn+n+m+2)+\log_2|P|,$

其中P为A,b,c,m,n的非零数乘积。

- 上界估计2： $l(I)\le 12mn+\log_2|P|.$
- 上界估计3：

$$l(I)\le 3(mn+n+m+2)+(mn+n+m+2)\log_2 M,$$

其中M为A,b,c,m,n中绝对值最大数。

下界估计

$$l(I)=\lceil\log_2(|m|+1)\rceil+\lceil\log_2(|n|+1)\rceil+\sum_{j=1}^{n}\lceil\log_2(|c_j|+1)\rceil$$
$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil\log_2(|a_{ij}|+1)\rceil+\sum_{i=1}^{m}\lceil\log_2(|b_i|+1)\rceil$$
$$+2(mn+m+n+2)$$

- 关系式

$$\lceil\log_2(x+1)\rceil\geq\log_2(x+1)\geq\log_2 x,\qquad x\geq 1$$

- 下界估计1： $l(I)\geq 2(mn+n+m+2)+\log_2|P|,$

  其中P为A,b,c,m,n的非零数乘积。

- 下界估计2： $l(I)\geq 2(mn+n+m).$
- 下界估计3： m, n, 0.

计算量

- 计算量（复杂性）：算法求解中的加、减、乘、除、比较、读和写磁盘等基本运算的总次数。
- 同样需要计算量的上下界估计。

计算次数与实例规模之间的关系

- $O(x)$的含义：一个实例为I，实例规模为$l(I)$, 算法A在求解I时的计算量（基本计算总次数,算法中的所有加、减、乘、除、读、写、比较等的和）为$C_A(I)$。当存在一个函数$g(x)$和一个正常数，使得对于该问题任意的实例I均满足

$$C_A(I) \le \alpha g(l(I)), \forall I$$

则记为

$$C_A(I) = O(g(l(I)))$$

注：只对多项式g(x)才有意义。

复杂性分析的概念

- 当给定一个算法，其存在多项式函数$g(x)$和一个与I无关的实数α, 使得下式成立，则称该算法为多项式时间算法。

$$C_A(I) \le \alpha g(l(I)), \forall I$$

非多项式时间算法

- 否则，称为非多项式时间算法。
  - 等价表述：对任意给定的最高次幂为正系数的多项式函数g(x)和正实数α，都存在一个实例I，使得

$$C_A(I) > \alpha g(l(I)).$$

  - 充分性表述（指数算法）：当存在一个单调增指数函数g(x)和一个正实数α，都有

$$C_A(I) > \alpha g(l(I)), \forall I$$

  - 充分性表述（指数算法）：当存在一个单调增指数函数g(x)，一个正实数α和无穷多个实例$\{I_k\ k=1,2,\ldots,\}$，都有

$$C_A(I_k) > \alpha g(l(I_k)), k = 1, 2, \ldots$$

多项式时间算法是一个好算法

- $n^{10000000000\ldots}$比1.00001^n好！！！

算法复杂性与实例规模的计算

- 精确估计
  - 实例中每一个系数按如下公式计算并相加

$$l(x) = \lceil \log_2(|x|+1) \rceil + 2$$

  - 算法复杂性，所有加、减、乘、除、读、写、比较等的和。

- 粗略估计，以假设达到目标为准则，以上下界来估计。

粗略估计原则

- 目标：多项式时间算法
- 原则：实例下界估计，算法计算量上界估计
  - 实例规模：下界$lb(I)$
  - 计算复杂性：选计算量最大的实例作为估计（上界）$UC_{\mathrm{A}}(I)$。

结论：若存在多项式函数$g(x)$（在x>M单调递增），满足

$$UC_A(I) \le \alpha g(lb(I))$$

则有

$$C_A(I) \le \alpha g(l(I))$$

上下界的使用原理

- 预期：多项式时间算法,当$x \geq K$(一个充分大的数)时，$g(x)$关于x单调升。
- 满足的关系式： $C_A(I) \leq \alpha g\big(l(I)\big)$
- 当满足 $C_A(I) \leq \mathrm{UP}(I)$ （上界）和$l(I) \geq lb(I)$（下界）时,
- 若对任意实例I, 存在多项式函数$g(x)$满足$UP(I) \leq \beta(lb(I))$, 其中β是一个与I无关的正数，则有

$C_A(I) \leq UP(I) \leq \beta g(lb(I)) \leq \alpha g\big(l(I)\big)$，其中

$\alpha = \max\{\beta, \beta g(j), j=1,2,\ldots,K\}$.

指数时间算法估计原理

- 目标：指数时间算法
- 原则：实例规模上界，算法计算量下界
  - 实例规模：上界$ub(I)$
  - 计算复杂性：选可以得到指数算法结论的估计量$LC_{\mathrm{A}}(I)$。

指数时间算法：若存在指数函数$g(x)$, $\alpha>0$（与I无关）和一个实例I, 满足

$$LC_A(I) \geq \alpha g(ub(I))$$

则有 $$C_A(I) \geq \alpha g(l(I))$$

界估计与算法复杂性的平衡

- 目标：多项式时间算法
  - 字长下界，计算时间上界，但放大或缩小不能破坏多项式的关系，可产生多项式阶的差异。阶数越低算法越好。
- 目标：指数算法
  - 字长上界，计算时间下界。
- 算法中与实例中某些常数有关？
  - 以后再讨论。

复杂性理论的功能I
——算法复杂性归类

- 给定问题P，设计一个算法A，A是否为多项式算法？
  - 当存在一个多项式函数$g(x)$和一个与I无关的正常数α，使得

$$C_A(I) \le \alpha g(l(I)), \qquad \forall I \in P$$

- 没有求解质量的要求！
  - A为多项式时间算法。
- 指数时间算法？

如何简单处理复杂性分析？——例1

- 问题：求n个数中的最大数。
- 实例：给定n=100及100个数。
- 算法：从第一个数开始，读出（赋值）、比较（大小）、更新（大则更新，否则保持不变），给出最大的数。
- 直观结论：多项式时间算法。
  - 简单处理：实例规模：n个数$a_1,a_2,\ldots,a_n$规模

$$l(I)=\sum_{i=1}^{n}(\lceil\log_2(|a_i|+1)\rceil+2)+\lceil\log_2(n+1)\rceil+2\geq 2(n+1)\geq n$$

- 计算次数：赋初值。对每个数比较，赋值，计算量上界$C_A(I)<=3n+1$。
- $C_A(I)=O(n)$, $g(x)=x$.

- 注意$l(I)$下界估计。
  - 下界$2(n+1)$,n都不影响多项式结论。
  - 但下界1,0等则影响结果。
- 同理注意$C_A(I)$上界估计。

如何简单处理复杂性分析？——例2

- 问题：TSP问题
- 实例：给定n个城市及城市间距离，假定最大距离不超过一个常数K。
- 算法：全排列，选出最短路径。
- 直观结论：非多项式时间算法。
  - 实例规模：$l(\mathrm{I})=\mathrm{O}(n^2)$。（上界控制）
  - 算法计算次数：$n!$ (关于n指数)（下界控制）
  - 不存在多项式函数$g(x)$, 使得$\mathrm{C_A}(I)=\mathrm{O}(\mathrm{g}(l(I)))$

TSP的“短视”算法——例3

- 记录一个行走城市路径和长度。初始行走路径为出发城市长度为0。
- 在行走路径是全部n个城市或无法再加入下一步行走城市，算法终止。否则循环计算一个城市到可以直接到达且没有到过城市的最短距离，加入行走路径。
- 终止第一种情况，若路径中最后一个与第一个城市有直接连接，加入连接，得到一个可行解。否则或第二种情况没得到可行解。

计算复杂性分析

- 直觉：多项式时间算法
- 实例规模下界：n^2
- 算法计算量上界估计。假设算法目前已得到：行走路径中已有k个城市，对行走路径的最后一个城市(i)，从第一个城市(j)开始的每一个城市首先判断是否有直接路径(d_{ij}，计算量1)，再判断是否已在k个城市中(计算量k-1)，再决定是否加入行走路径中(赋值1，比较值计算量1、或更新计算量1)，最后更新行目前走列表(计算量1)和行走的路长(计算量1)。这一过程的计算量上界：$n(k+3)+2$。总体计算量上界：$n^2(k+3)+2n \le n^3+5n^2 \le 6n^3$
- $g(x)=x^{1.5}$

复杂性理论的功能II
——问题复杂性归类

- 给定问题，能否设计一个多项式算法A？
- 多项式问题
  - 对于给定的一个问题，若存在一个求解该问题精确解的多项式算法，则称给定的问题是多项式时间可解问题。所有多项式问题的集合记为**P**(Polynomial)。
- 对算法解质量要求！

复杂性估计例子——线性规划

- 线性规划

$$\min \{c^{\mathrm{T}}x: Ax=b, x\geq 0\}$$

- 输入：m, n, c, A, b

$$l(I)=\lceil \log_2(|m|+1)\rceil+\lceil \log_2(|n|+1)\rceil+\sum_{j=1}^{n}\lceil \log_2(|c_j|+1)\rceil$$

$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil \log_2(|a_{ij}|+1)\rceil+\sum_{i=1}^{m}\lceil \log_2(|b_i|+1)\rceil$$

$$+2(mn+m+n+2)$$

下界估计

- 下界 $\log_2 |P| + 2(mn + m + n + 2)$

或 $\log_2 |P| + mn + m + n + 2$

$$mn + m + n + 2$$

n

P为A, b, c, m, n中非零数的乘积。

多项式时间估计——内点算法

- 原始-对偶内点算法的计算量，$\varepsilon>0$为给定精度，标准形式，$m\le n$, x^0 ,s^0分别为原始和对偶问题的初始内点（参考方、邢《线性锥优化》）。 x^0 ,s^0是基本可行解的严格凸组合。

$$O(\sqrt{n}\log_2\frac{(x^0)^T s^0}{\varepsilon})$$

- 实例规模估计——下界

$$l_b=\log_2|P|+mn+m+n+2+(2n+1+\log_2|M|),$$

M为x^0, s^0 ,$1/\varepsilon$中非零数的乘积。

- 线性规划问题存在多项式时间算法
- $g(x)=x^{5/2}$,
- 线性规划问题是多项式时间问题。

$$\sqrt{n}\log_2\frac{(x^0)^T s^0}{\varepsilon}=\sqrt{n}\log_2(x^0)^T s^0+\sqrt{n}\log_2\frac{1}{\varepsilon}\le\sqrt{n}(2n+1)\log_2 M$$

$$\le\sqrt{l_b}(l_b)l_b=(l_b)^{5/2}$$

单纯形算法——指数算法

- 给定一个实例(Klee and Minty, 1972)

$$\min -x_n$$

$$s.t. 4x_1 - 4r_1 = 1$$

$$x_1 + s_1 = 1$$

$$4x_j - x_{j-1} - 4r_j = 0,\ j = 2,3,\ldots,n$$

$$4x_j + x_{j-1} + 4s_j = 4,\ j = 2,3,\ldots,n$$

$$x_j, r_j, s_j \geq 0,\ j = 1,2,\ldots,n.$$

- $3n$个变量，$2n$个约束。

- 单纯形算法的计算量（选取一个实例）

$$2^n - 1$$ （指数型）

- 实例规模估计——上界（线性规划规模估计）

$$\min c^T x$$

$$s.t. Ax = b$$

$$x \geq 0$$

参数：$m, n, c(n维), A(mn维), b(m维)$

- 理论规模

$$l(I)=\lceil\log_2(|m|+1)\rceil+\lceil\log_2(|n|+1)\rceil+\sum_{j=1}^{n}\lceil\log_2(|c_j|+1)\rceil$$

$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil\log_2(|a_{ij}|+1)\rceil+\sum_{i=1}^{m}\lceil\log_2(|b_i|+1)\rceil$$

$$+2(mn+m+n+2)$$

- 放大公式 $\lceil\log_2(|x|+1)\rceil\le 1+\log_2|x|$

$$l(I)\le\log_2|P|+3(mn+m+n+2)$$

一般模型估计：

$$l(I) \le \log_2 |P| + 3(mn + m + n + 2)$$

- 回到给定的例，对3n个变量，2n个约束的模型
  - 最大数为4，P中最多有不超过6nn+5n+2个数

$$l(I) \le (\log_2 |4| + 3)(6n^2 + 2n + 3n + 2) \le 65n^2$$

- 结论：单纯形算法为指数型算法。

1.3 邻域的概念

对于组合最优化问题(D,F,f)，D上的一点到D的子集的一个映射

$$N: \quad x \in D \to N(x) \in 2^D$$

且 $x \in N(x)$. $N(x)$ 称为x的邻域，$y \in N(x)$ 称为x的一个邻居。

若 $x^* \in F$ 满足

$$f(x^*) \le (\ge) f(x), \text{ 其中 } x \in N(x^*) \cap F,$$

则称x^*为f在F上的局部(local)最小(最大)解（点）。若

$$f(x^*) \le (\ge) f(x), \ x \in F,$$

则称x^*为f在F上的全局(global)最小(最大)解（点）。

例子

- 背包问题
  - $N(x)=\{y\in\{0,1\}^n \mid \|y-x\|_1 \le 1\}$
  - 邻居个数：n+1个。
  - $N(x)=\{y\in\{0,1\}^n \mid \|y-x\|_1 \le 2\}$
  - 邻居个数：n+n(n-1)/2+1个
- TSP问题
  - N(x): x是{1,2,…,n}的一个排列, 2-opt邻居为交换两个城市的位置或保持不变。
  - 邻居个数：n(n-1)/2+1
  - k-opt邻域

邻域构造要点

- 邻居数量为成员个数的多项式时间长度。
  - 为多项式时间算法准备。
- 好的结构连通性：能从一个解用邻域迭代到任意其他解。
  - 可以从任意初始解出发，达到全局最优解。
- 邻居中尽量多的可行解。
  - 提高搜索效率，减少不可行解的判别。

1.4 启发式算法

- 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）下给出待解决组合最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。
- 另一种定义为：启发式算法是一种技术。这种技术使得在可接受的计算费用内去寻找最好的解，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，无法阐述所得解同最优解的近似程度。

背包问题的贪婪算法(greedy algorithm)

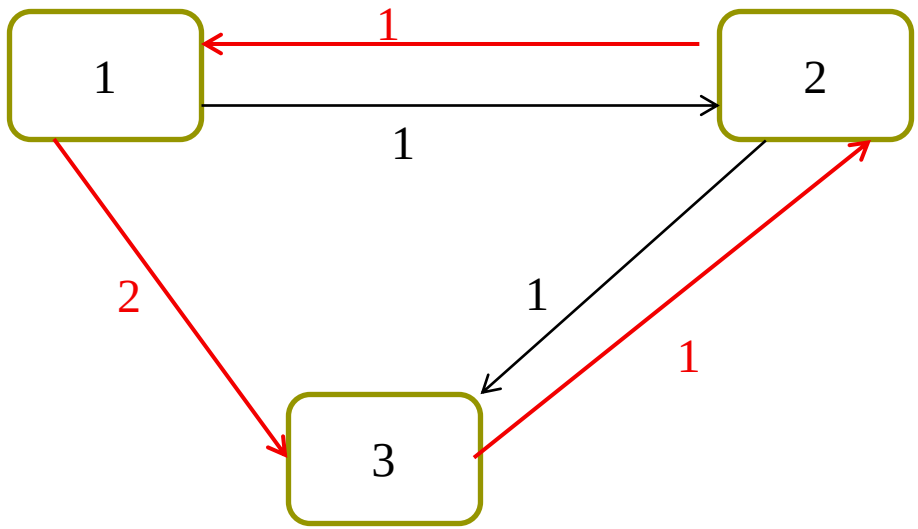
STEP1 对物品以 $\frac{c_i}{a_i}$ 从大到小排列，不妨把排列记成{1,2,...,n}, k:=1；

STEP2 若 $\sum_{i=1}^{k-1} a_i x_i + a_k \le b$，则 $x_k = 1$；否则，$x_k = 0$。k:=k+1；当 k=n+1 时，停止；否则，重复 STEP2。

计算量： $O(n \log n)$

TSP的短视算法

- 从一个城市开始，在一步直接可达且没有到过的城市中选距离最短的城市为下一站，直到走遍所有城市.
- 计算量$O(n^2)$
- 可能得不到可行解.



从城市1出发一个短视解（不可行）: 1->2

从城市1出发一个可行解解: 1->3->2->1

近似算法

设A是一个问题，一个实例I的最优解和启发式算法H的目标函数值分别为$z_{OPT}(I)$和$z_H(I)$。当$z_{OPT}(I) \neq 0$时，$\varepsilon \geq 0$，称H是A的ε近似算法(ε-approximation algorithm)当且仅当

$$\left|z_H(I) - z_{OPT}(I)\right| \leq \varepsilon \left|z_{OPT}(I)\right|, \qquad \forall I \in A。$$

启发式算法定义的算法集合包含近似算法定义的算法集合。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。

性能比(approximation ratio)或最坏情形比(the worst case ratio)

- 考虑最小化目标函数的优化问题，设启发式算法得到优化问题的一个可行解，则有

$$z_{OPT}(I) \le z_H(I)$$

- 性能比

$$R = \sup_{I \in A} \frac{z_H(I)}{z_{opt}(I)}$$

- 当存在ε近似算法H时，则有性能比R≤1+ε。
- 当R=1+ε时，则H的最坏情形比是1+ε。

- 近似算法的基本思想是用最坏情形评价算法，这样的方法被称为最坏情形分析（the worst case analysis）。
- 为什么不用绝对误差？

$$\left|z_H(I) - z_{OPT}(I)\right| \le \varepsilon, \qquad \forall I \in A$$

绝对误差：在假设系数有理数的前提下，ε足够小等于无误差！

性能比讨论

极小化目标，启发式算法求的可行解。

$$z_{OPT}(I) \le z_H(I) \le Rz_{OPT}(I), \forall I, R \ge 1.$$

$$R = \sup_{I \in A} \frac{z_H(I)}{z_{opt}(I)}$$

极大化目标，启发式算法求的可行解。

$$Rz_{OPT}(I) \le z_H(I) \le z_{OPT}(I), \forall I, R \le 1.$$

$$R = \inf_{I \in A} \frac{z_H(I)}{z_{opt}(I)}$$

- 极小化目标函数

$$z_{OPT}(I) \le z_H(I) \le Rz_{OPT}(I) + c, \forall I$$

- 渐近性能比

$$AR(H) = \lim_{k\to\infty} \sup_I \{\frac{z_H(I)}{z_{OPT}(I)} | z_{OPT}(I) = k\}$$

启发式算法的长处

- 数学模型本身是实际问题的简化，或多或少地忽略了一些因素；而且数据采集具有不精确性；参数估计具有不准确性；这些因素**可能造成**启发式算法所得到的解不比最优算法所得到的解差。
- 有些难的组合最优化问题可能还**没有找到最优算法**，即使存在，由算法复杂性理论，它们的计算时间是无法接受或不实际的。
- 一些启发式算法**可以用**在最优算法中。如在分枝定界算法中，可以用启发式算法估计下（上）界。
- 简单易行；比较直观；易被使用者接受。
- 速度快，这在实时管理中非常重要。
- 多数情况下，程序简单，因此易于在计算机上实现和修改。

短处

- 不能保证求得最优解（有时甚至不能保证求到可行解）。
- 表现不稳定。启发式算法在同一问题的不同实例计算中会有不同的效果，有些很好，而有些则很差。在实际应用中，这种不稳定性造成计算结果不可信，可能造成管理的困难。
- 算法的好坏依赖于实际问题、算法设计者的经验和技术，这一点很难总结规律，同时使不同算法之间难以比较。

启发式算法的分类

- **一步算法**
  - 该算法的特点是：不在两个可行解之间比较，在未终止的迭代过程中，得到的中间解有可能不是一个可行解。

TSP 的路线搜索算法为：

STEP1 P={1}; N={1,2,...,n};i=1;

STEP2 从$Q=\{d_{ij}\mid i$可达j，但$j\notin P\}$选最小的值d_{ii_0}，对应城市为i_0,置P=P$\cup\{i_0\}$，若P={1,2,...,n}或$Q=\varnothing$，停止，否则$i=i_0$，重复STEP2。

- **改进算法**
  - 迭代过程是从一个可行解到另一个可行解变换，通过两个解的比较而选择好的解，进而作为新的起点进行新的迭代，直到满足一定的要求为止。
- **数学规划算法**
  - 数学规划算法主要指用连续优化（如线性规划）的方法求解组合最优化问题（如整数线性规划模型），其中包括一些启发式规则。

整数规划问题 IP 松弛为线性规划 LP：

$$(\text{IP}) \quad \begin{aligned} & z = \min c^T x \\ & s.t. Ax \geq b \\ & x \geq 0, x \in Z^n \end{aligned} \qquad (\text{LP}) \quad \begin{aligned} & z_L = \min c^T x \\ & s.t. Ax \geq b \\ & x \geq 0 \end{aligned}$$

将LP的解转化为IP的可行解，如四舍五入法，取上整数、下整数等。

- **智能算法**
  - 禁忌搜索算法，模拟退火算法，遗传算法，蚁群优化算法，人工神经网络算法等。
  - 这些算法的目标是希望能够求解NP难问题的全局最优解，有一定的普适性，可用于解决大量的实际应用问题。
- **其他方法**

启发式算法的性能分析

- 常用三个主要评价指标
  - 算法的复杂性（计算效率）、解的偏离程度（计算效果），和算法的稳健性。
- 三种不同的分析手段。
  - 最坏情形分析，概率分析的方法，计算模拟分析。
    - 理论方法：最坏情形分析，概率分析的方法
    - 计算方法：计算模拟分析

最坏情形分析(the worst case analysis)

- 算法的计算复杂性
  - 例：线性规划的单纯形算法
- 算法的计算效果
  - H为α近似，当α不可改进，即为最小的上界，称为最坏比。

$$\alpha z_{OPT}(I) \geq z_H(I) \geq z_{OPT}(I)$$

$$\alpha z_{OPT}(I) + d \geq z_H(I) \geq z_{OPT}(I)$$

  - 渐近最坏比率

$$AR(H) = \lim_{k\to\infty} \sup_I \{\frac{z_H(I)}{z_{OPT}(I)} | z_{OPT}(I) = k\}$$

背包问题的贪婪算法的最坏情形分析

STEP1 对物品以 $\frac{c_i}{a_i}$ 从大到小排列，不妨把排列记成{1,2,...,n}, k:=1；

STEP2 若 $\sum_{i=1}^{k-1} a_i x_i + a_k \le b$，则 $x_k = 1$；否则，$x_k = 0$。k:=k+1；当 k=n+1 时，停止；否则，重复 STEP2。

渐近最坏比率分析：α=0为渐近紧界

实例说明：有两个物品，c_1=1+ε, $a_1 = 1$, $c_2 = K$, $a_2 = K$, $b = K$, ε为充分小的正数

$$\frac{z_G(I)}{z_{OPT}(I)} = \frac{1+\varepsilon}{K} \to 0, K \to +\infty$$

一般背包问题

$$z_{OPT}(I) = \max \sum_{j=1}^{n} c_j x_j$$

$$s.t. \sum_{j=1}^{n} a_j x_j \le b$$

$x_j \ge 0$整数, $j = 1,2,...,n.$

$$z_G(I) \ge c_1 \left\lfloor \frac{b}{a_1} \right\rfloor$$

$$z_{OPT}(I) = \max \sum_{j=1}^{n} \frac{c_j}{a_j} a_j x_j \le \frac{c_1}{a_1} b = \max\{c_j \frac{b}{a_j} \mid j = 1,2,...,n\}$$

$$\frac{z_G(I)}{z_{OPT}(I)} \ge \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{\frac{b}{a_1}} \ge \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{1 + \left\lfloor \frac{b}{a_1} \right\rfloor} \ge \frac{1}{2}$$

$$c_1 = K + 3\varepsilon, \quad c_2 = K, \quad a_1 = \frac{K}{2} + \varepsilon, \quad a_2 = \frac{K}{2}, \quad \varepsilon > 0, \quad b = K > 2$$

$$\alpha = \lim_{z_{OPT}(I) \to \infty} \frac{z_H(I)}{z_{OPT}(I)} = \lim_{K \to \infty} \frac{K + 3\varepsilon}{2K} = \frac{1}{2}$$

概率分析

- 概率分析研究算法的平均计算量
  - 例：单纯形算法的评价（Borgwardt，1982）

$$\max c^T x$$

$$s.t.\ Ax \le e \qquad A \in R^{m\times n},\ e=1 \in R^m,\ m \ge n$$

$c, A_1, A_2, \ldots, A_m$ 在 R^n 中具有球面对称测度

上下界分别：$n^2 m^{1/n-1}$　　$n^{1.5} m^{1/n-1}$

- 概率分析研究算法的计算效果
  - 期望值模型，排队的平均等待时间等

大规模计算分析

- 算法的计算复杂性
  - 通过计算耗费的计算机中央处理器（CPU）的计算时间表现
- 计算得到的解的性能
  - 通过计算停止时的输出结果的好坏表现，评价的是算法的计算效果
- 解的稳定性
  - 解的平均值，方差等值

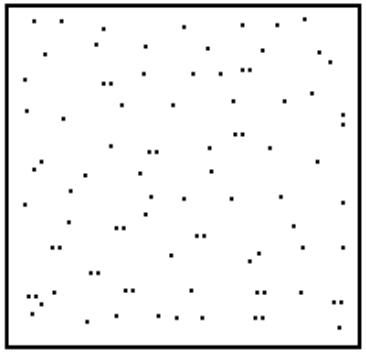
要点

- 数据产生
  - 用数值实验的方法产生一些有代表性的数据
- 评价指标
  - 需要尽量少的计算和其他信息
- 一个算法进行评价
  - 必须有比较的对象，用上下界替代最优目标值
- 多个算法进行比较
  - 算法间竞争（矮子里拔高个）

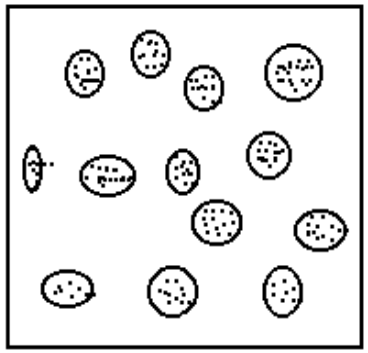
数据产生(http://web.cba.neu.edu/~msolomon/)

Vehicle Routing Problems with Time Windows (VRPTW)：*m*个车辆服务于*n*个客户，客户与客户间有路径费用；同时，每个客户有特殊的服务时间称为时间窗口，只有在服务时间内的服务才是有效的；如何安排*m*个车辆使得满足服务时间的要求且行驶的费用最小。

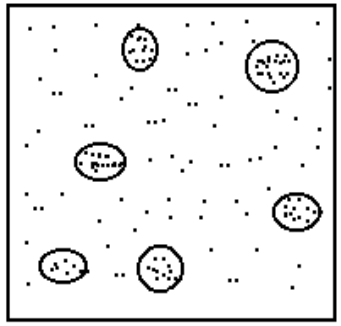
客户的位置分布



uniform



Cluster



Semi-cluster

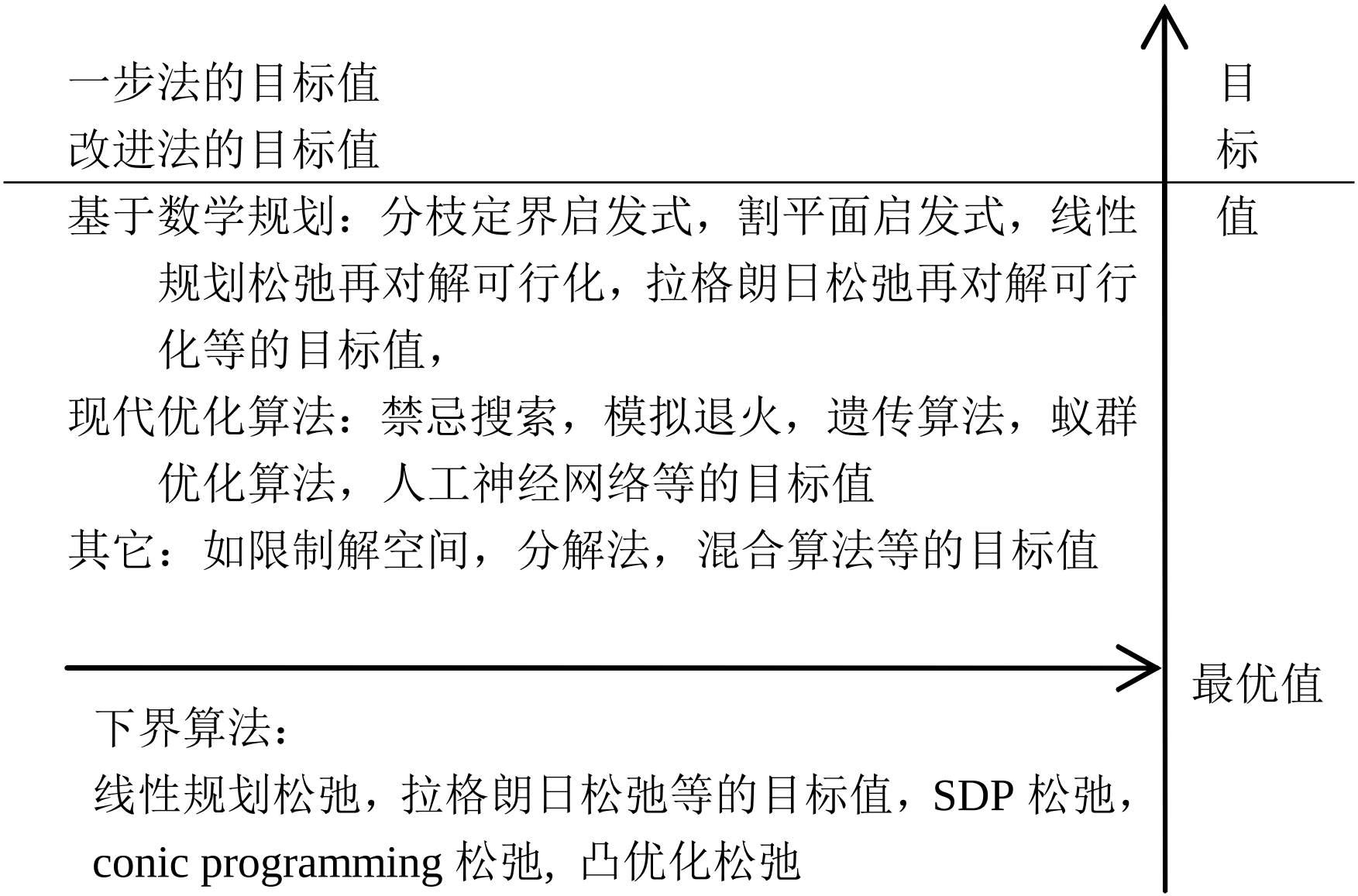
时间窗口　客户时间窗口： 75%和100%；时间跨度：大和小。数据均匀分布产生。

评价指标

- 算法的性能：CPU统计
- 解的性能：
  - 绝对差 $z_H(I)-z_{OPT}(I)$ $z_H(I)-z_{LB}(I)$
  - 相对差

$$\frac{z_H(I)-z_{OPT}(I)}{z_{OPT}(I)} \quad \frac{z_H(I)-z_{LB}(I)}{z_{LB}(I)} \quad \frac{z_H(I)-z_{OPT}(I)}{z_H(I)} \quad \frac{z_H(I)-z_{LB}(I)}{z_H(I)}$$

- 解的稳定性
  - 统计计算方差


一步法的目标值
改进法的目标值
基于数学规划：分枝定界启发式，割平面启发式，线性规划松弛再对解可行化，拉格朗日松弛再对解可行化等的目标值，
现代优化算法：禁忌搜索，模拟退火，遗传算法，蚁群优化算法，人工神经网络等的目标值
其它：如限制解空间，分解法，混合算法等的目标值
目标值
最优值
下界算法：
线性规划松弛，拉格朗日松弛等的目标值，SDP 松弛，conic programming 松弛，凸优化松弛