数学建模

浙江大学数学系　谈之奕

tanzy@zju.edu.cn

运筹与统计

组合优化

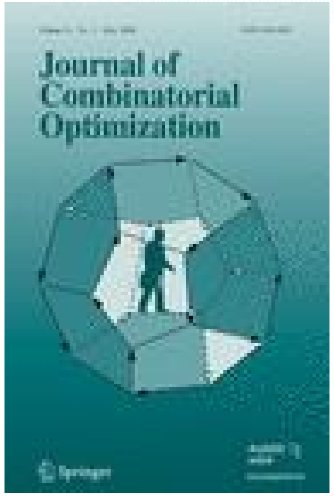
组合优化

- 组合优化（**Combinatorial Optimization)**：从有限个可行解中找出使某个目标函数达到最优的解的优化问题
- 连续优化与离散优化
  - 连续优化（**Continuous Optimization**）：决策变量在实数空间内取值的优化问题
    - **Lagrange** 乘子法
  - 离散优化（**Discrete Optimization**）：涉及离散对象的优化问题

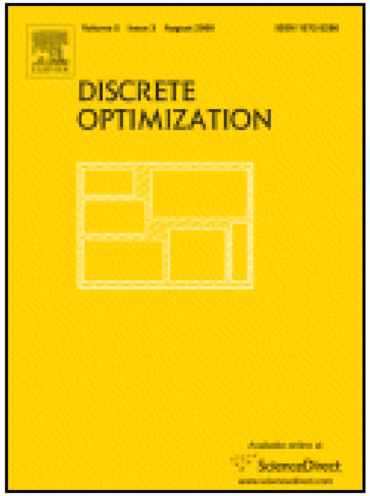
浙江大学
ZheJiang University
数学建模

组合优化

- 组合优化与组合数学（**Combinatorics**）
  - 同为研究离散对象的数学分支，但两者侧重不同。后者着重研究满足特定性质对象的存在性、计数、构造等问题，前者要求在众多可行解中按一定标准选出最优解



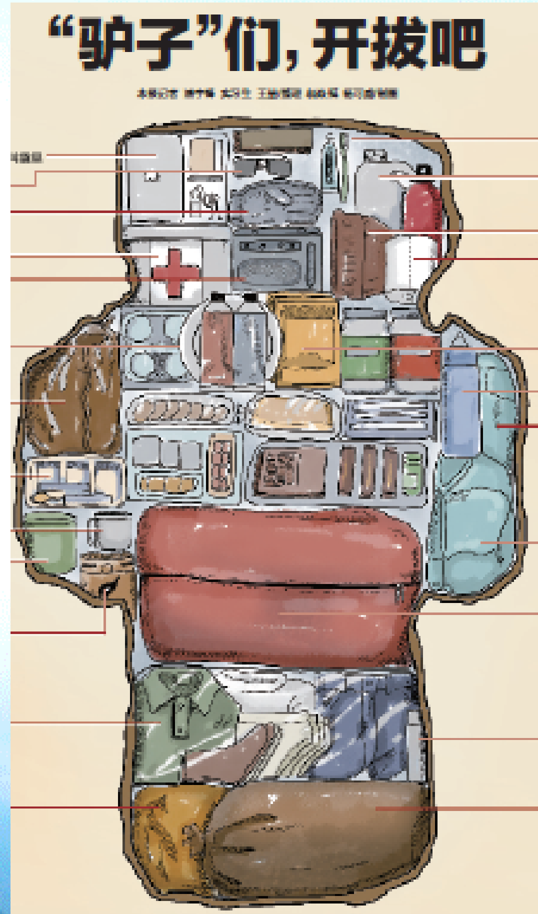
Journal of Combinatorial Optimization



Discrete Optimization

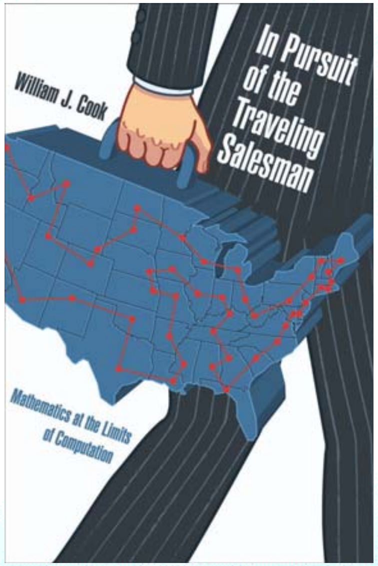
背包问题

- 一背包客准备参加自助游，想要携带的物品很多，但随身背包的容量有限，因此希望通过综合考虑，使放入背包中的物品对旅行的帮助最大
- 由于每件物品可以选择放入或不放入，因此物品数为 n 时，可行解数目不超过 2^n 个



旅行售货商问题

- 一推销商想在若干个城市中推销自己的产品。计划从某个城市出发，经过每个城市恰好一次，最后回到出发的城市。假设城市之间距离已知，推销商应如何选择环游路线，使他走的路程最短。该问题称为旅行售货商问题（**Traveling Salesman Problem**，**TSP**）

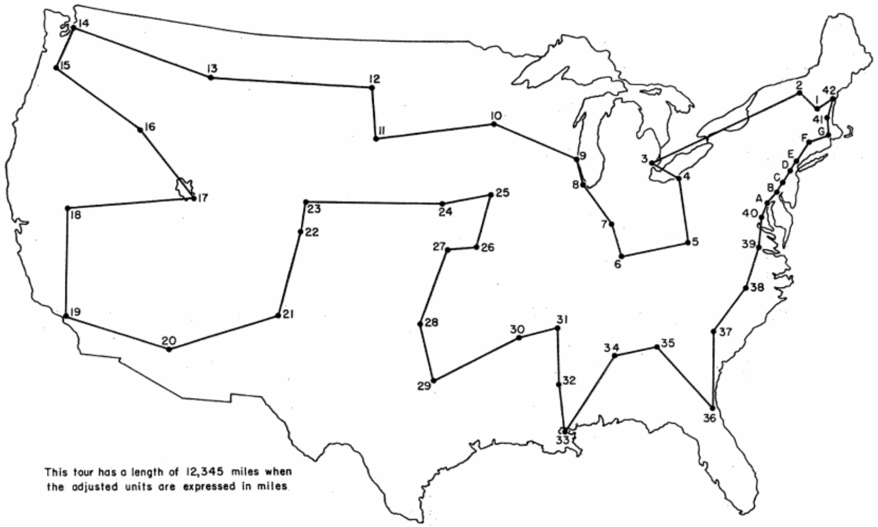


Cook, W. J., ***In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*****, Princeton University Press, 2012.**

旅行售货商问题

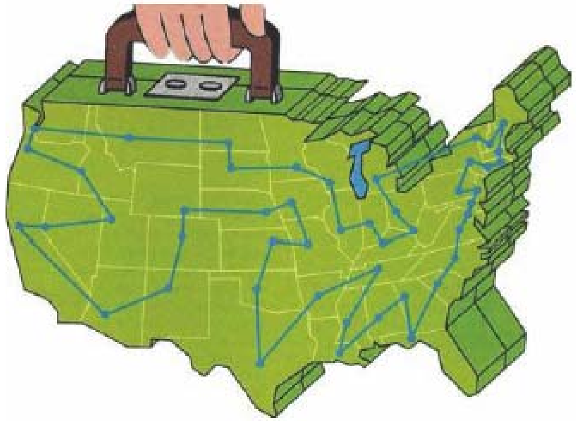
- 每一条环游路线对应于$1,2,\cdots,n$ 的一个排列。不同的排列数目共有$(n-1)!$个
  - 非对称TSP（**Asymmetric TSP**）
  - 对称TSP（**Symmetric TSP**）　$\dfrac{(n-1)!}{2}$
  - 度量TSP（**metric TSP**）
  - **Euclidean TSP**

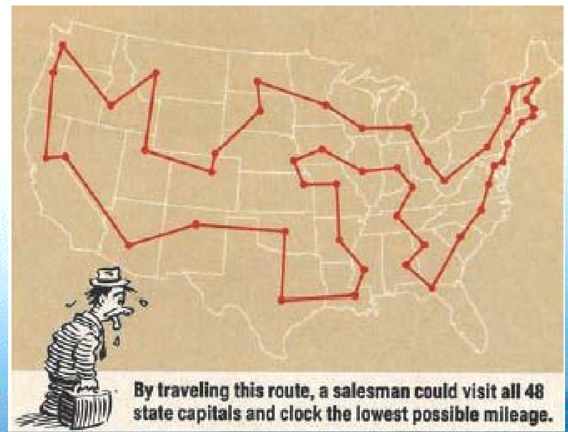
环游美国的TSP



美国49个城市的最优TSP环游

Dantzig, G., Fulkerson, R., Johnson, S., Solution of a Large-Scale Traveling-Salesman Problem, ***Journal of the Operations Research Society of America*****, 2, 393-410, 1954.**



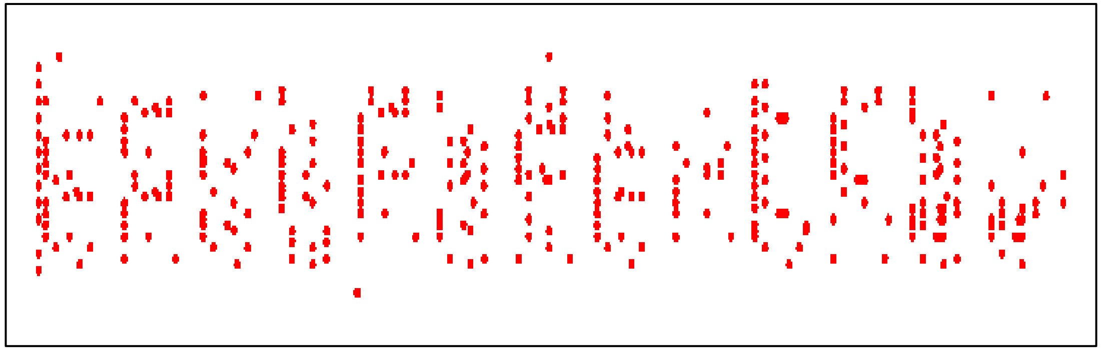


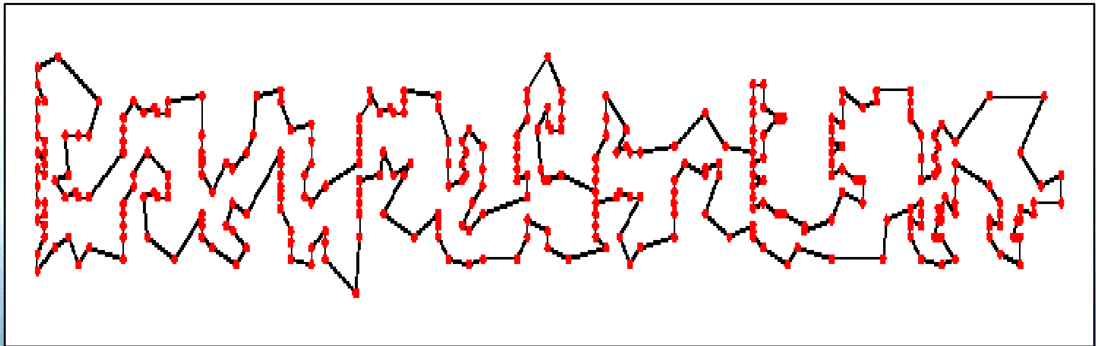
美国**48**个州首府的**TSP**环游(上图载自***Discover***的环游经过各州的顺序与原论文相同，但不再是最优环游，下图为新实例的最优环游)

VLSI设计中的TSP

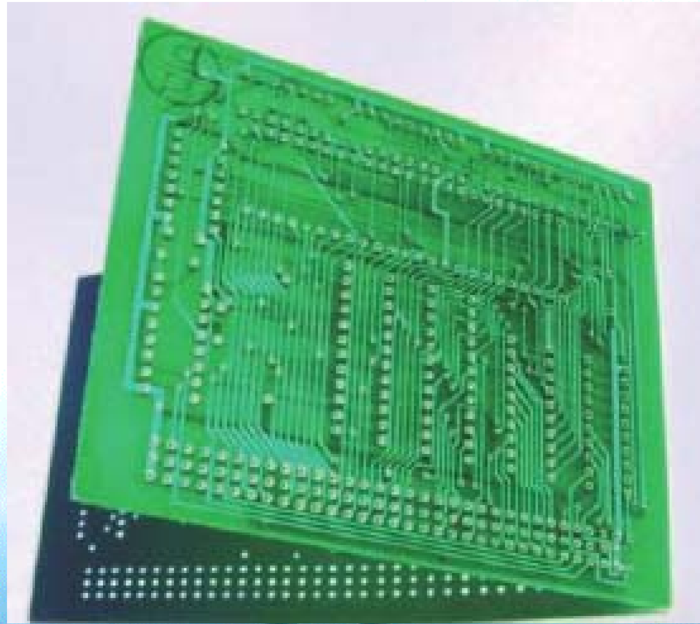
数学建模

- **PMA343**





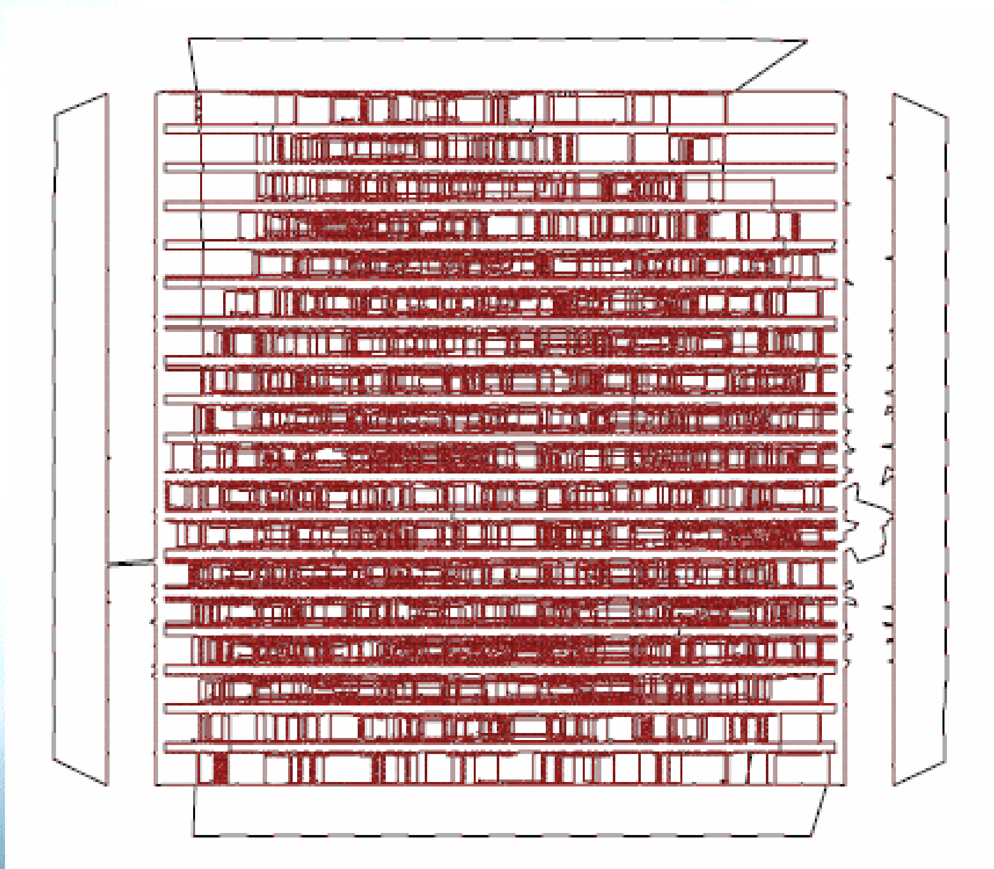
- **441个焊点的印刷电路板**

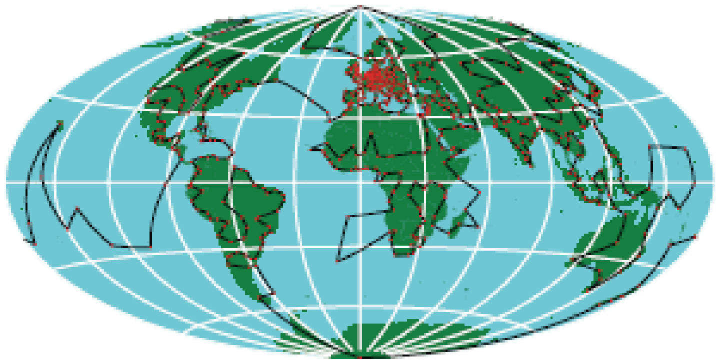


大规模TSP实例

浙江大学 ZheJiang University

数学建模





全球666个城市的最优环游（1991）

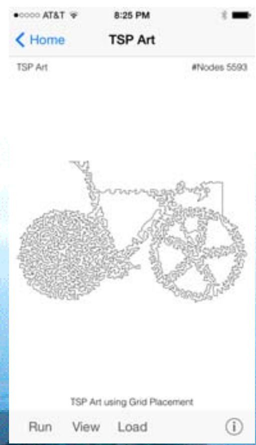
来自计算机芯片设计含85900点的TSP最优环游（2009）

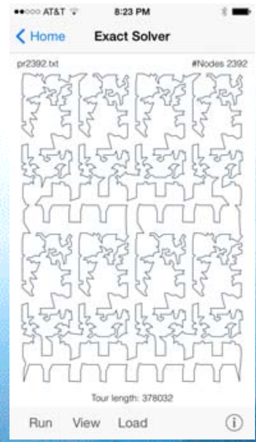
http://www.math.uwaterloo.ca/tsp/

Concorde TSP

数学建模

- The Concorde App computes exact optimal solutions for TSP. Instances of 1,000 or more cities can often be solved exactly, with all computations carried out locally on your iPhone or iPad





View in iTunes

This app is designed for both iPhone and iPad

Free

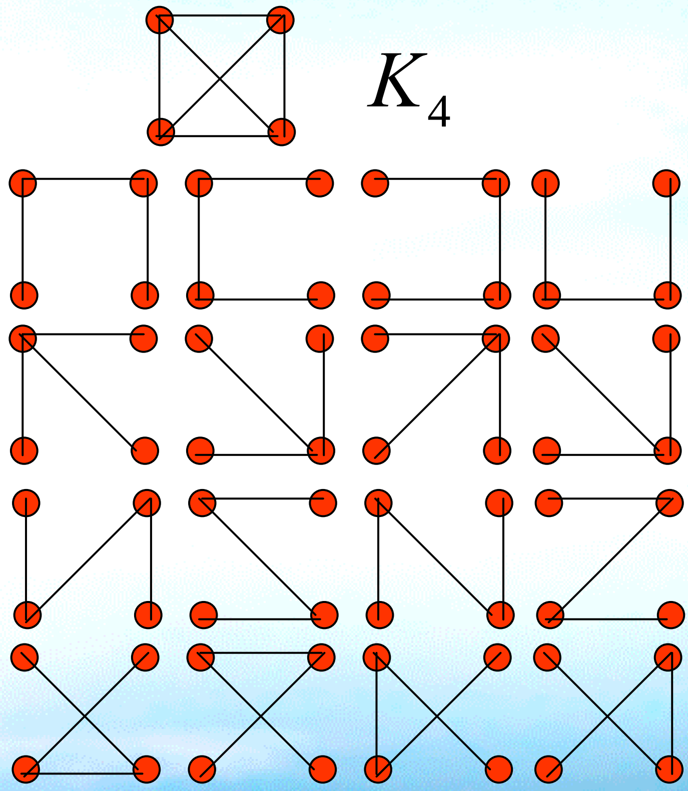
Category: Education
Updated: May 18, 2015
Version: 1.5
Size: 4.5 MB
Language: English
Seller: William Cook
© William Cook, Monika Mevenkamp

最小生成树

- 在某地区内修建连接若干城镇的公路系统，假设公路造价与长度成正比，如何设计造价最低的建造方案
- 最小生成树（**minimum spanning tree，MST**）
  - 连通的无圈图称为树（**tree**）
  - 树 T 称为图 G 的生成树（**spanning tree**），若 T 是 G 的子图、且和 G 有相同的顶点集
  - 赋权图 G 的所有生成树中总权和最少的生成树称为最小生成树

最小生成树

- 将城镇视作图的顶点，城镇之间的距离视作连接两个顶点的边的长度。生成树可以把所有城镇都连接起来；最小生成树具有最小的总长度
- 完全图 K_n 有 n^{n-2} 颗不同的生成树



指派问题

浙江大学
ZheJiang University
数学建模

- 指派问题（**Assignment Problem**）
  - 设有n项任务需分配给 n位员工，每人完成其中一项，员工 i完成任务 j 所需时间为 c_{ij}，如何分配可使完成所有任务所用总时间最少
  - 不同的分配方案共有n!种

Burkard, RE, Dell'Amico, M., Martello, S., *Assignment Problems*, SIAM, 2009.
封面题图：**Jan van Eyck, Portrait of Giovanni Arnolfini and his Wife, 1434**，现藏英国伦敦国家美术馆

穷举

- 组合优化问题通常不能通过穷举所有可能的解加以比较来求解，其原因是可行解的数目可能是一很大的数，以致于当前或相当长的一段时间内人力或计算机不能承受

FIGURE 2
Grand Vizier Sissa Ben Dahir, a skilled mathematician, asks his reward from King Shirham of India.

《One Two Three . . . Infinity: Facts and Speculations of Science》插图

穷举

$2^0=1$

King Shirham vs. Sissa Ben Dahir

按一千克小麦含25000粒计算，棋盘上的小麦总计约为7400亿吨，按目前的平均产量计算，是全世界一千多年生产的全部小麦

$$2^{63}=9223372036854775808$$
$$=9.22\times10^{18}$$

函数量阶

浙江大学 ZheJiang University
数学建模

函数	10	20	40	100
$\lg n$	1秒	1.30秒	1.60秒	2秒
n	4.34秒	8.69秒	17.37秒	43.4秒
n^5	12小时	16天	514天	138年
2^n	444秒	5.27天	151世纪	1.7×10^{20}世纪
$n!$	18.2天	3.3×10^{8}世纪	1.1×10^{38}世纪	1.2×10^{148}世纪
n^n	138年	1.4×10^{16}世纪	1.6×10^{54}世纪	1.4×10^{190}世纪

函数量阶

浙江大学 ZheJiang University

数学建模

	现在的计算机	快100倍	快10000倍	快1000000倍
$\lg n$	N	N^{100}	N^{10000}	$N^{1000000}$
n	N	$100N$	$10000N$	$1000000N$
n^5	N	$2.51N$	$6.31N$	$15.85N$
2^n	N	$N+6.64$	$N+13.28$	$N+19.93$

全球Top500超级计算机

http://www.top500.org/

数学建模

时间	公司	计算机	浮点数运算次数	提高倍数
1993.6(首届)	TMC	CM5	59.70GFlops	
1998.6(11届)	Intel	ASCI-Red	1338.00GFlops	22.4
2003.6(21届)	NEC	NEC Vector	35860.00GFlops	600.7
2010.11(36届)	国防科大	天河一号	2566.0TFlops	42981
2015.6(45届)	国防科大	天河二号	33.86PFlops	567169

kiloFLOPS=10^3, megaFLOPS=10^6, gigaFLOPS=10^9, teraFLOPS=10^12, petaFLOPS=10^15, exaFLOPS=10^18, zettaFLOPS=10^21, yottaFLOPS=10^24

穷举

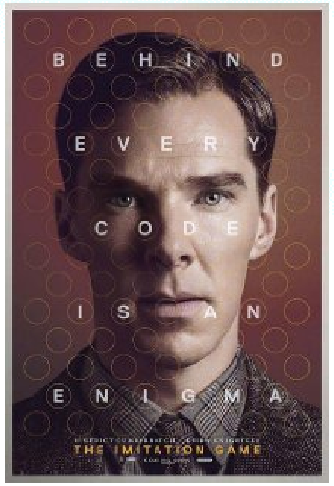
- 也有一些问题，如最小生成树、指派问题等，可以不通过穷举或类似于穷举的方法找到最优解，从而使求解时间大幅下降。而对有些问题，如背包问题、TSP等，目前还没有找到这样的方法
- 组合优化研究的一个重要方面是区分哪些问题是容易求解，哪些问题是难求解的

计算复杂性

浙江大学 ZheJiang University
数学建模

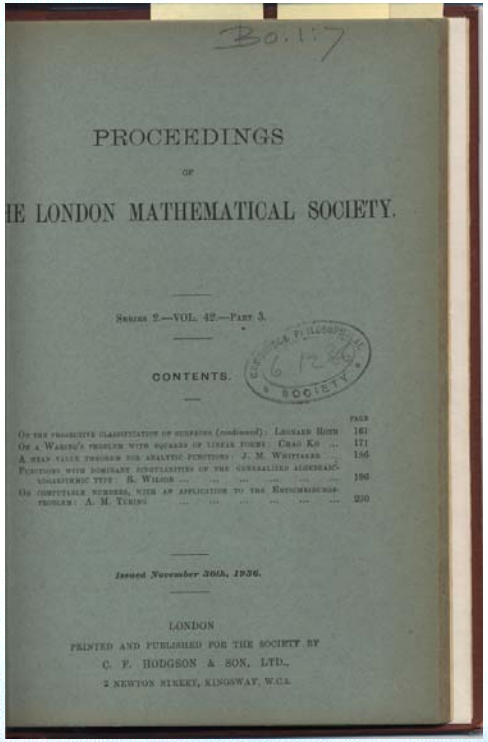
- 计算复杂性（**computational complexity**）理论在组合优化学科中的应用之一是将问题按难度分类，从而为进一步研究指明方向
- 计算复杂性理论建立在一种名为图灵机（**Turing machine**）的理论计算模型之上。该模型由**A. Turing**于**1936**年提出，它能模拟目前所有的合理计算模型

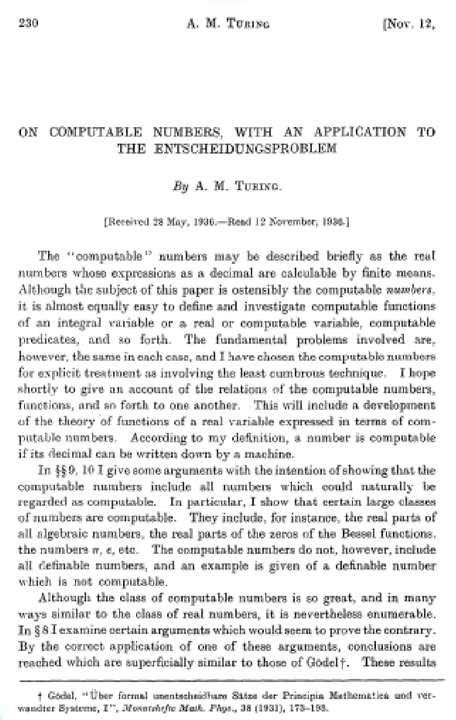
Alan Turing
英国计算机学家
(1912-1954)



The Imitation Game
（《模仿游戏》）
（**2014**年上映）

图灵机





230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

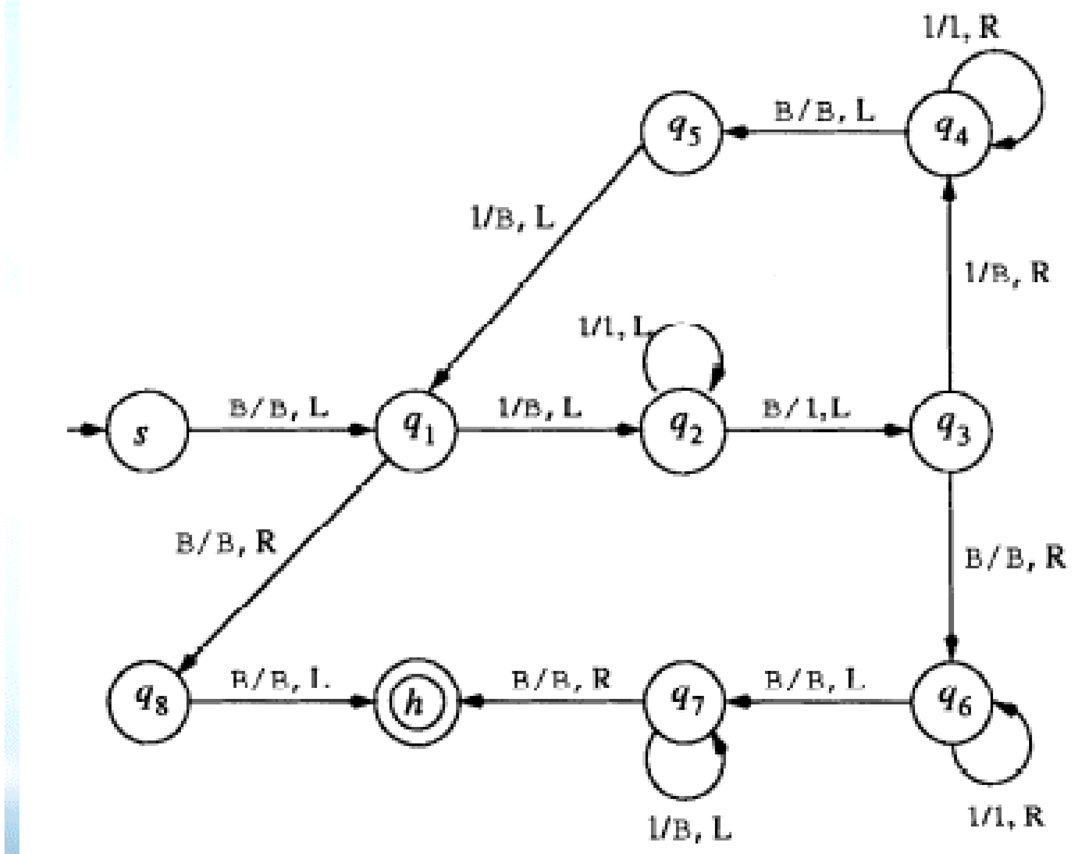
† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

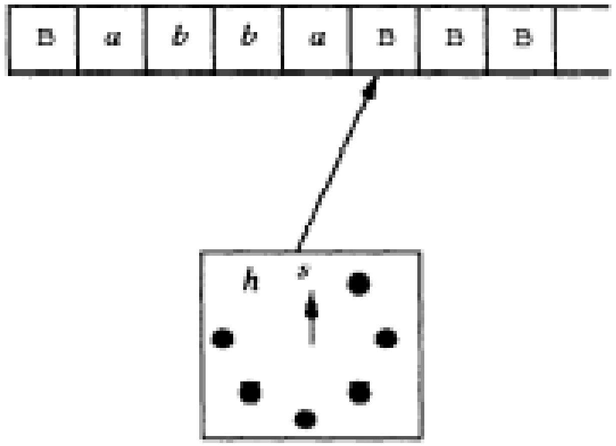
2012年6月23日Turing诞辰 100 周年当日Google发布的互动Doodle

Turing, A., On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, S2-42, 230–265

图灵机

数学建模





B	1	1	1	B	1	1	B

B	1	B

B	1	1	B	1	1	1	B

B	B

算法

浙江大学 ZheJiang University

数学建模

- 算法：在有限步骤内求解某一问题的一组含义明确的可以完全机械执行的规则
- **Algorithm is a sequence of computational steps that transform the input into the output**

Algoritmi（**al-Khwarizmi**的拉丁译名）

↓

Algorism(us)（阿拉伯数字系统，十进制）

↓

Algorithm　（仿**logarithm**所造法语单词，后引入英语，**19**世纪转为现义）

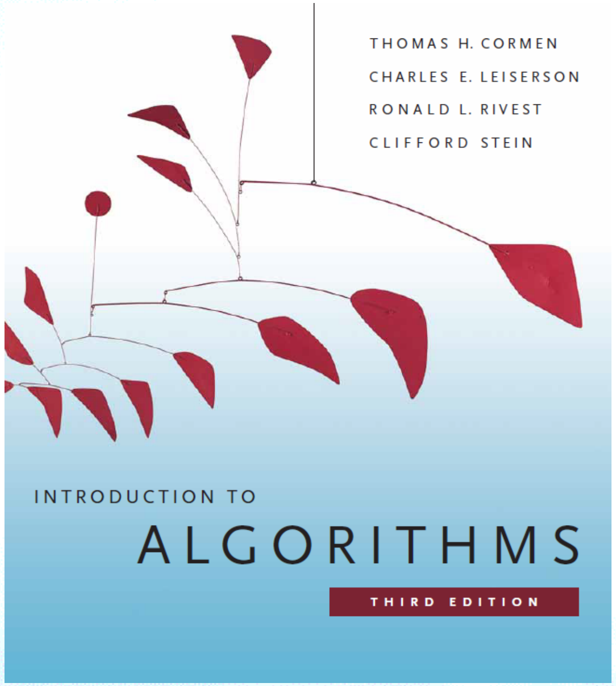
Abu Ja'far Muhammad ibn Musa al-Khwarizmi

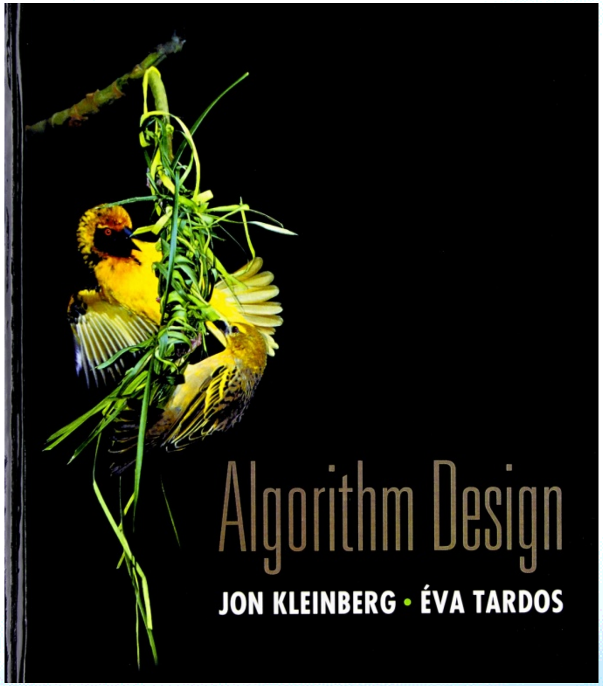
（约**780**-约**850**）

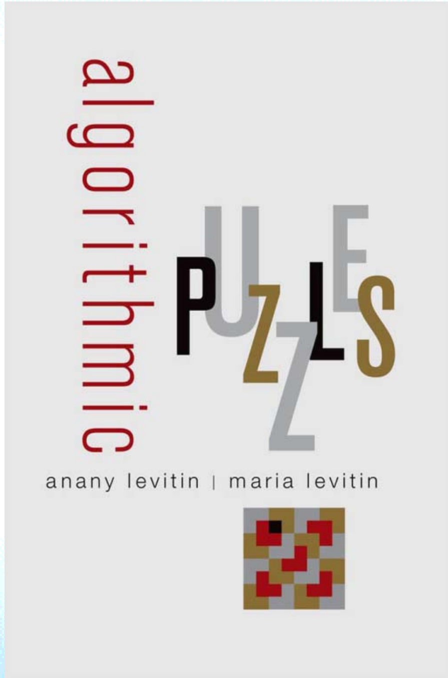
波斯数学家

23

算法







Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms*. MIT press, 2009

Kleinberg J, Tardos É. *Algorithm Design*. Pearson Education India, 2006

Levitin A, Levitin M. *Algorithmic puzzles*. Oxford University Press, 2011.

24

时间复杂性

数学建模

- 用算法执行过程中所需的加、乘、比较、赋值等基本运算次数表示算法所用的时间
- 在计算机中表示一实例所需的字节数称为实例的规模（**size**）
- 算法的时间复杂性（**time complexity**）是关于实例规模 n 的一个函数 $f(n)$，它表示用该算法求解所有规模为 n 的实例中所需基本运算次数最多的那个实例的基本运算次数

不同硬件配置所用时间不一样？

不同大小的例子所用基本运算次数不一样？

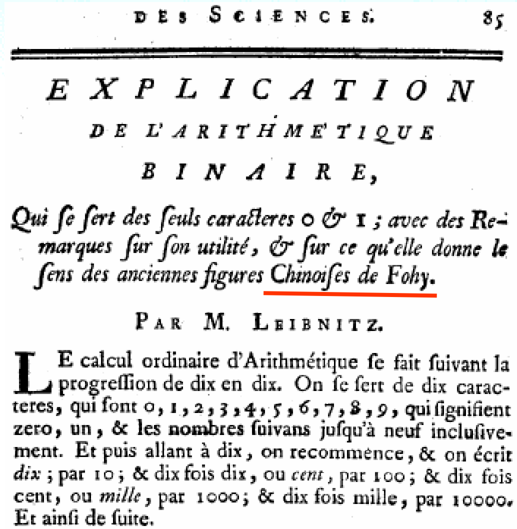
同样大小的例子所用基本运算次数也可能不一样？

ZheJiang University

数学建模

时间复杂性

- 在计算机中，常用二进制表示整数，因此存储大小为 k 的整数所需字节数为 $\lfloor \log_2 k \rfloor + 1$
- 若一算法时间复杂性 $f(n)=O(p(n))$，这里 $p(\cdot)$ 为一多项式，则称它为多项式时间算法。不能这样限制时间复杂性函数的算法称为指数时间算法


DES SCIENCES. 85

EXPLICATION
DE L'ARITHMETIQUE
BINAIRE,

Qui ſe ſert des ſeuls caracteres 0 & 1; avec des Remarques ſur ſon utilité, & ſur ce qu'elle donne le ſens des anciennes figures Chinoiſes de Fohy.

PAR M. LEIBNITZ.

LE calcul ordinaire d'Arithmétique ſe fait ſuivant la progreſſion de dix en dix. On ſe ſert de dix caracteres, qui ſont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, qui ſignifient zero, un, & les nombres ſuivans juſqu'à neuf incluſivement. Et puis allant à dix, on recommence, & on écrit *dix*; par 10; & dix fois dix, ou *cent*, par 100; & dix fois cent, ou *mille*, par 1000; & dix fois mille, par 10000. Et ainſi de ſuite.

Leibniz G., Explication de l'Arithmétique Binaire, *Memoires de mathématique et de physique de l'Académie royale des sciences*, Académie royale des sciences, 1703

26

高效算法

- 结合关于函数增长速度的比较，和算法的实际运行效果，通常将多项式时间算法称为高效算法（**efficient algorithm**）

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

—— Edmonds, J. Paths, trees, and flowers.
***Canadian Journal of Mathematics*, 17, 449–467, 1965**

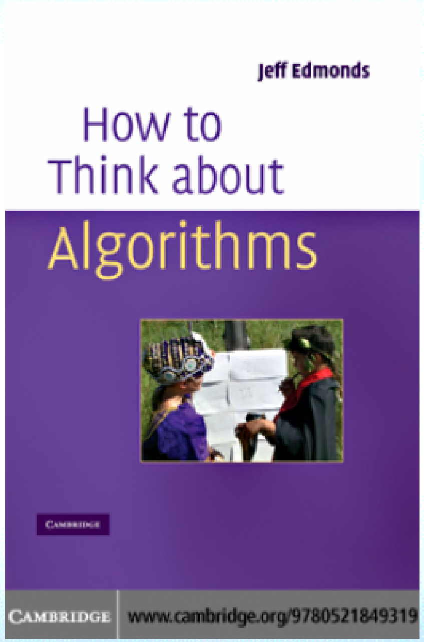
27

高效算法

Kathie Cameron
加拿大劳瑞尔大学
数学系教授

Jack Edmonds
（1934- ）
原加拿大滑铁卢
大学数学系教授
(上图摄于1957年)

Jeff Edmonds
加拿大约克大学电气工程
和计算机科学系教授

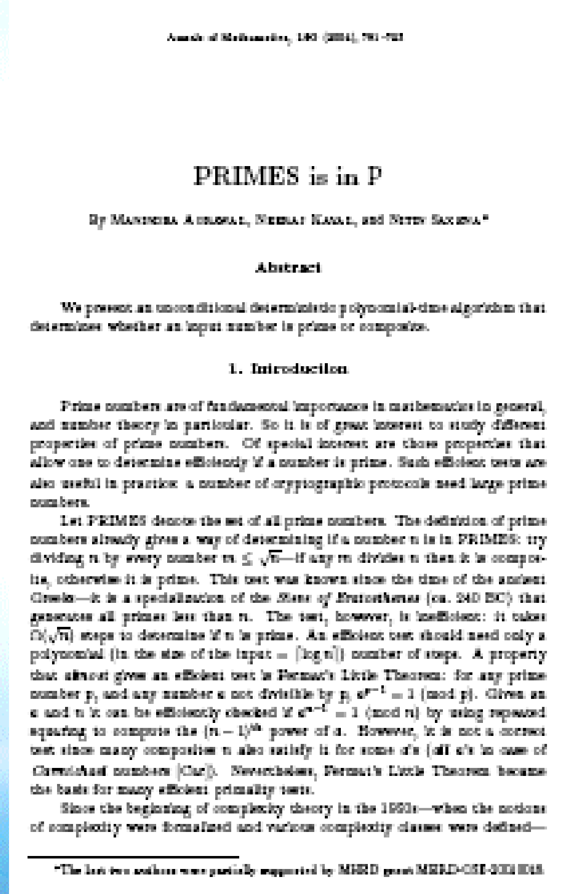


Edmonds J. *How to think about algorithms*. Cambridge University Press, 2008.

$\mathcal{P}$ 类

- 若一问题已找到多项式时间算法，称这样的问题属于多项式时间可解问题类（**polynomial solvable problem class**），记为 $\mathcal{P}$。证明一问题属于$\mathcal{P}$ 类只需设计出求解该问题的多项式时间算法
- 最小生成树、指派问题即为$\mathcal{P}$ 类中的问题；判断一个整数是否为质数也属于$\mathcal{P}$ 类

Agrawal, M., Kayal, N., Saxena, N., PRIMES is in P. *Annals of Mathematics*, 160, 2, 781-793, 2004



Annals of Mathematics, 160 (2004), 781–793

PRIMES is in P

By Manindra Agrawal, Neeraj Kayal, and Nitin Saxena*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

1. Introduction

Prime numbers are of fundamental importance in mathematics in general, and number theory in particular. So it is of great interest to study different properties of prime numbers. Of special interest are those properties that allow one to determine efficiently if a number is prime. Such efficient tests are also useful in practice: a number of cryptographic protocols need large prime numbers.

Let PRIMES denote the set of all prime numbers. The definition of prime numbers already gives a way of determining if a number n is in PRIMES: try dividing n by every number $m \le \sqrt{n}$—if any m divides n then it is composite, otherwise it is prime. This test was known since the time of the ancient Greeks—it is a specialization of the *Sieve of Eratosthenes* (ca. 240 BC) that generates all primes less than n. The test, however, is inefficient: it takes $\Omega(\sqrt{n})$ steps to determine if n is prime. An efficient test should need only a polynomial (in the size of the input $= \lceil \log n \rceil$) number of steps. A property that almost gives an efficient test is Fermat's Little Theorem: for any prime number p, and any number a not divisible by p, $a^{p-1} = 1 \pmod{p}$. Given an a and n it can be efficiently checked if $a^{n-1} = 1 \pmod{n}$ by using repeated squaring to compute the $(n-1)^{th}$ power of a. However, it is not a correct test since many composites n also satisfy it for some a's (all a's in case of Carmichael numbers [Car]). Nevertheless, Fermat's Little Theorem became the basis for many efficient primality tests.

Since the beginning of complexity theory in the 1960s—when the notions of complexity were formalized and various complexity classes were defined—

*The last two authors were partially supported by MHRD grant MHRD-CSE-20010018.

素性测试

Eratosthenes筛法（Sieve of Eratosthenes）

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Eratosthenes of Cyrene
（约公元前**276-**
约公元前**194**）
古希腊科学家

素性测试

数学建模

- 素性测试问题：给定整数 n，判断 n 是否为素数
  - 实例规模为 $\log_2 n$，**Eratosthenes**筛法是指数时间算法
  - **AKS素性测试**算法可在 $O(\log_2^{7.5} n \cdot \mathrm{poly}(\log\log n))$ 时间内完成

Agrawal, M., Kayal, N., Saxena, N., PRIMES is in P. *Annals of Mathematics*, 160, 2, 781-793, 2004

Manindra Agrawal：印度理工学院坎普尔学院（**Indian Institute of Technology Kanpur**）计算机科学与工程系教授

Neeraj Kayal与**Nitin Saxena**是**1997**年国际数学奥林匹克印度队成员，**2002**年时均为该系本科生，**"Towards a Deterministic Polynomial-Time Primality Test"**是他们的一项本科生科研项目

31

$\mathcal{NP}$ 类

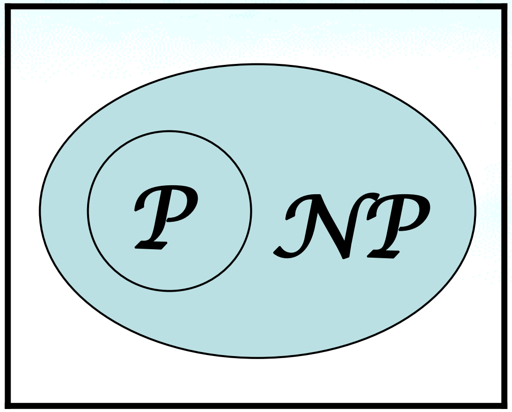
数学建模

- 非确定性算法多项式时间可解问题类（**nondeterministic polynomial solvable problem class**），记为$\mathcal{NP}$ 类
- 所谓非确定性算法在多项式时间内求解某问题，是指它能
  - 猜想出该实例的一个可行解，其规模不超过输入规模的多项式函数
  - 在输入规模的多项式时间内验证猜想是否正确

非确定性算法只是为研究而定义的一种理论算法模型，在现实生活中并不存在，非确定性算法的严格定义需要借助图灵机

$\mathcal{P}$=$\mathcal{NP}$ 猜想

数学建模



- $\mathcal{P}$ 类中的问题是多项式时间可求解问题，而$\mathcal{NP}$ 类中的问题仅是多项式时间可验证问题。因此 $\mathcal{P} \subseteq \mathcal{NP}$
- 是否有 $\mathcal{P} = \mathcal{NP}$ 成立是数学和理论计算机科学中一个重要课题

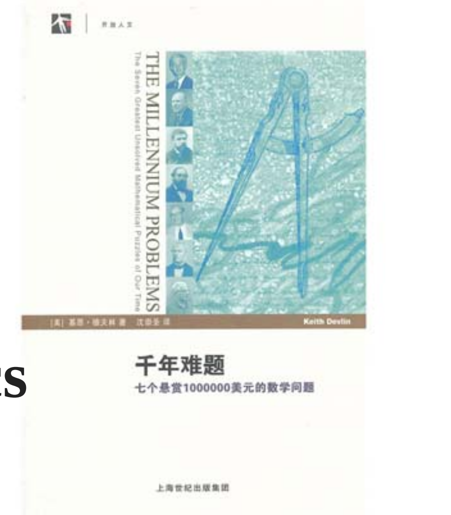
千年难题

浙江大学
ZheJiang University
数学建模

- **Millennium Problems**
  - **Yang–Mills and Mass Gap**
  - **Riemann Hypothesis**
  - **P vs NP Problem**
  - **Navier–Stokes Equation**
  - **Hodge Conjecture**
  - **Poincaré Conjecture**
  - **Birch and Swinnerton-Dyer Conjecture**

CMI

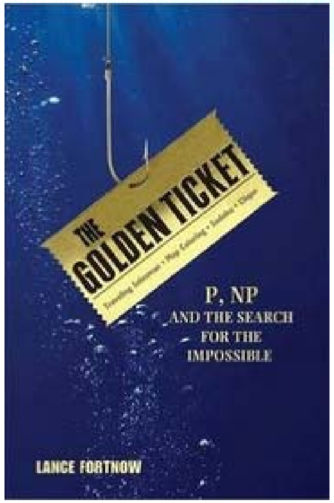
Clay Mathematics Institute（CWI）

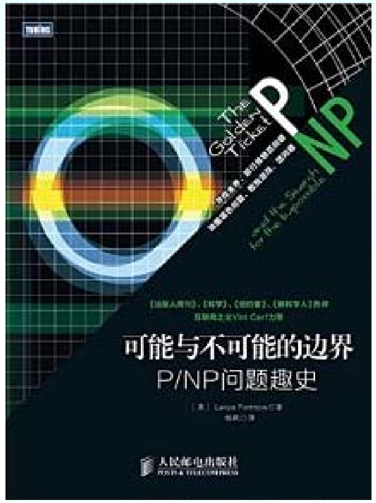


Devlin, K. J., ***The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*****, Basic Books , 2003.**（中译本：沈崇圣译，上海科技教育出版社，**2012**）

$\mathcal{P}=\mathcal{NP}$ 猜想

- 尽管 $\mathcal{P}=\mathcal{NP}$ 猜想是未决问题，但是目前普遍相信 $\mathcal{P}\neq\mathcal{NP}$ 成立，并在此假设下进一步研究 $\mathcal{NP}$ 类内部的结构





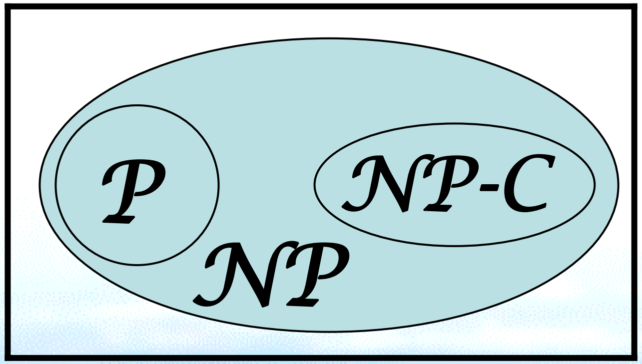
Fortnow, L., ***The Golden Ticket: P, NP, and the Search for the Impossible*****, Princeton University Press , 2013.** （中译本：可能与不可能的边界：**P/NP**问题趣史，杨帆译，人民邮电出版社，**2014.**

Fortnow L. The status of the P versus NP problem. ***Communications of the ACM*****, 2009, 52(9): 78-86.**

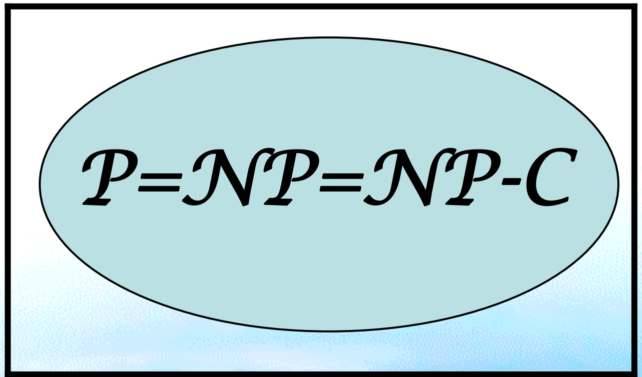
$\mathcal{NP}$–完全问题

数学建模

- $\mathcal{NP}$ 类中最“难”的问题子集称为$\mathcal{NP}$ －完全类，记为$\mathcal{NP}$-C。$\mathcal{NP}$-C 类中的问题称为$\mathcal{NP}$ －完全问题（$\mathcal{NP}$ -**complete problem**）
  - 若$\mathcal{NP}$-C 类中有一个问题有多项式时间算法，则$\mathcal{NP}$ 类中所有问题都有多项式时间算法



$\mathcal{P} \neq \mathcal{NP}\ (\mathcal{P} \cup \mathcal{NP}\text{-}C \neq \mathcal{NP})$



$\mathcal{P} = \mathcal{NP}$

$\mathcal{P}$，$\mathcal{NP}$，与$\mathcal{NP}$—完全

浙江大学 ZheJiang University

数学建模

- $\mathcal{NP}$ 类是没有多项式时间算法的问题组成的集合 ×
  - $\mathcal{P}$ 问题也属于$\mathcal{NP}$ 类，它们都有多项式时间算法
  - $\mathcal{NP}$ 类并未包含所有问题，没有多项式时间算法的问题并不都属于$\mathcal{NP}$ 类
  - $\mathcal{P} \neq \mathcal{NP}$仅是猜想，若 $\mathcal{P} = \mathcal{NP}$，$\mathcal{NP}$ 类中所有问题均有多项式时间算法
- $\mathcal{NP}$ 类中的问题是最难的问题 ×
  - 混淆$\mathcal{NP}$ 类和$\mathcal{NP}$-C 类的概念
  - $\mathcal{NP}$-C 类仅是$\mathcal{NP}$ 类中最难的问题，在$\mathcal{NP}$ 类之外可能存在着更“难”的问题

$\mathcal{P}$，$\mathcal{NP}$，与$\mathcal{NP}$—完全

TEACHER'S BOOK　普通高级中学实验教科书　数学3（必修）教师教学用书

4. 计算的复杂性

计算的复杂性测度函数有三类：一类是指数型的，常写为 c^n 的形式（c 是常数）；
的，常写为 n^k 的形式（k 为非负整数）；另一类是对数型的，常写为 $\log n$ 的形式．具
问题分别称为指数复杂性，多项式复杂性和对数复杂性．

人们习惯于把理论上可计算的问题类称为能行可计算的，而把具有多项式复杂性
效可计算的，通常称为 P 问题．NP 问题是指还未找到多项式复杂性算法的问题．

研究和实验表明，单纯靠提高计算机速度并不能解决 NP 问题．例如，根据某些
录，对于复杂性为 2^n 的问题，即使计算机速度提高 1 000 倍，也只能是多算约 10 道题
为，解决 NP 问题的关键是要从数学上找出好的算法．事实上，数学家们也找到了各种各样的好办法，
大大简化了计算．例如，用计算机对卫星照片进行处理，如果在一张 10 cm² 的照片上以一微米为间隙
打上格子，则处理一张照片需要进行 10^{16} 次运算，即使用每秒百亿次的计算机也要连续算上十多个昼夜．
后来，有人发明了一种好的算法，大大降低了计算的复杂性，使得用同样的计算机计算只需要 $\frac{1}{3}$ 秒．

数理逻辑

数学建模

- 数理逻辑（**mathematical logic**）：用数学的方法研究逻辑推理和数学计算，将推理论证、数学计算的过程符号化、形式化、公理化的学科

1956年**Gödel**致**von Neumann**信，信中对若干数理逻辑问题算法和复杂性的讨论被认为是计算复杂性研究的开端

Princeton 20./III.1956.

Lieber Herr v. Neumann!

Ich habe mit grösstem Bedauern von Ihrer Erkrankung gehört. Die Nachricht kam mir ganz unerwartet. Morgenstern hatte mir zwar schon im Sommer von einem Schwächeanfall erzählt, den Sie einmal hatten, aber er meinte damals, dass dem keine grössere Bedeutung beizumessen sei. Wie ich höre, haben Sie sich in den letzten Monaten einer radikalen Behandlung unterzogen u. ich freue mich, dass diese den gewünschten Erfolg hatte u. es Ihnen jetzt besser geht. Ich hoffe u. wünsche Ihnen, dass

Kurt Friedrich Gödel
（1906－1978 ）
奥地利哲学家、数学家

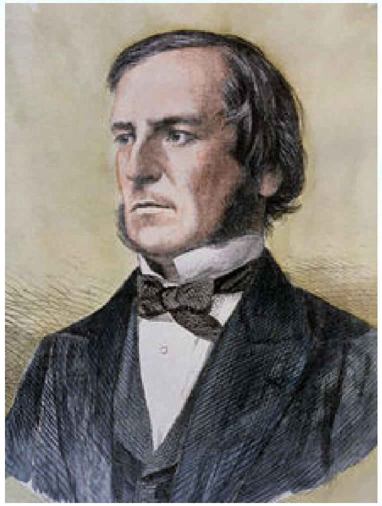
John von Neumann
（1903－1957）
匈牙利裔美国科学家

39

Boolean变量

浙江大学 ZheJiang University
数学建模

- 仅可取“真” 和“假”（**True(T)和False(F)，1和0**）两种值的变量称为**Boolean变量**
- **Boolean变量的运算**
  - 非（**negation**）$\neg$： $\neg x = 1 \Leftrightarrow x = 0$
  - 析取（**disjunction**）$\vee$
    $$x_1 \vee x_2 = 1 \Leftrightarrow x_1 = 1 \text{或} x_2 = 1$$
  - 合取（**conjunction**）$\wedge$
    $$x_1 \wedge x_2 = 1 \Leftrightarrow x_1 = 1 \text{且} x_2 = 1$$
- **Boolean变量的运算遵从**双重否定律、 **De Morgan律**、析取对合取的分配率、合取对析取的分配率等运算定律



George Boole
（1815-1864）
英国数学家、哲学家、逻辑学家

Boolean表达式

数学建模

- 若干**Boolean**变量用运算符和括号按一定的逻辑关系联结起来的表达式称为**Boolean**表达式（**Boolean expression**）
- 对出现在**Boolean**表达式中的所有**Boolean**变量各指定一个值，可按**Boolean**变量的运算法则确定表达式的值**1**或**0**
- 任一**Boolean**表达式都存在与之等价的合取范式（**conjunctive normal form, CNF**）
  - 文字（ **literal** ）：变量或变量的非
  - 子句（ **clause** ）：若干个文字的析取
  - **CNF**：若干个子句的合取

$$\neg(\neg x_1 \vee x_2) \vee x_3 \Leftrightarrow (\neg\neg x_1 \wedge \neg x_2) \vee x_3 \Leftrightarrow (x_1 \wedge \neg x_2) \vee x_3$$

$$\Leftrightarrow (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3)$$

SAT

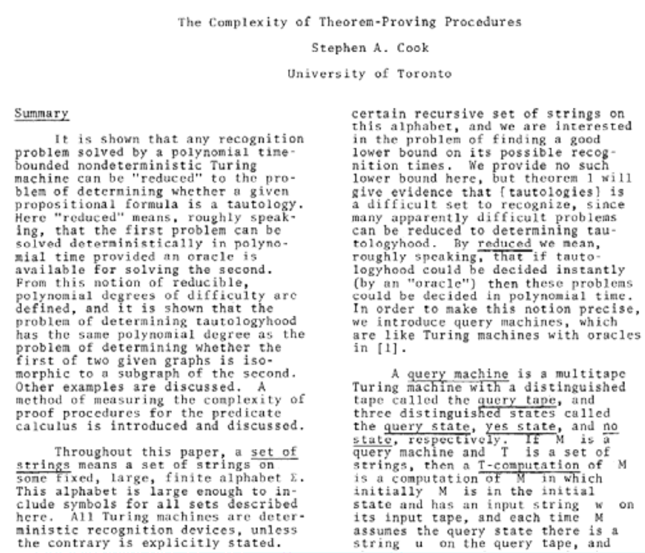
- 可满足性问题（ **Satisfiability**，**SAT**）
  - 给定一合取范式，问是否存在其变量的一种赋值，使得该表达式值为真
    - $(x_1 \vee \neg x_2) \wedge x_2$ ： x_1 取**1**，x_2 取**1** 可使表达式值为**1**
    - $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \neg x_1$ ：不论 x_1，x_2 取值为何值，表达式值均为 **0**
  - 猜想所有变量的一种赋值，在多项式时间内可验证表达式值确为**1**，因此 $\mathrm{SAT} \in \mathcal{NP}$

42

SAT

数学建模

- **1971年，Cook运用图灵机语言，通过$\mathcal{NP}$问题的一种等价定义，用$\mathcal{NP}$－完全问题的定义证明了SAT问题的$\mathcal{NP}$－完全性**
- **SAT问题被认为是第一个$\mathcal{NP}$－完全问题**



The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ. This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and

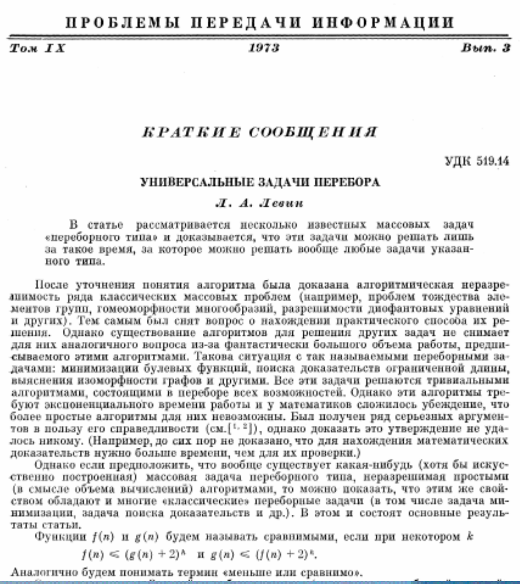
Cook SA. The complexity of theorem-proving procedures, *Proceedings of the 3rd annual ACM Symposium on Theory of Computing*, 151-158, 1971.

43

Cook-Levin定理

数学建模

- 与**Cook**同时，**Levin**独立地给出了若干$\mathcal{NP}$－完全问题，SAT $\in \mathcal{NP}\text{-}C$ 因此被称作**Cook-Levin**定理



ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Том IX *1973* *Вып. 3*

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов групп, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тем самым был снят вопрос о нахождении практического способа их решения. Однако существование алгоритмов для решения других задач не снимает для них аналогичного вопроса из-за фантастически большого объема работы, предписываемого этими алгоритмами. Такова ситуация с так называемыми переборными задачами: минимизации булевых функций, поиска доказательств ограниченной длины, выяснения изоморфности графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что более простые алгоритмы для них невозможны. Был получен ряд серьезных аргументов в пользу его справедливости (см.[1, 2]), однако доказать это утверждение не удалось никому. (Например, до сих пор не доказано, что для нахождения математических доказательств нужно больше времени, чем для их проверки.)

Однако если предположить, что вообще существует какая-нибудь (хотя бы искусственно построенная) массовая задача переборного типа, неразрешимая простыми (в смысле объема вычислений) алгоритмами, то можно показать, что этим же свойством обладают и многие «классические» переборные задачи (в том числе задача минимизации, задача поиска доказательств и др.). В этом и состоят основные результаты статьи.

Функции $f(n)$ и $g(n)$ будем называть сравнимыми, если при некотором k

$$f(n) \leqslant (g(n)+2)^k \quad \text{и} \quad g(n) \leqslant (f(n)+2)^k.$$

Аналогично будем понимать термин «меньше или сравнимо».

Levin LA, Universal Sequential Search Problems, *Problems of Information Transmission*, 9, 115–116, 1973

Trakhtenbrot BA, A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6, 384-400, 1984

Leonid Anatolievich Levin

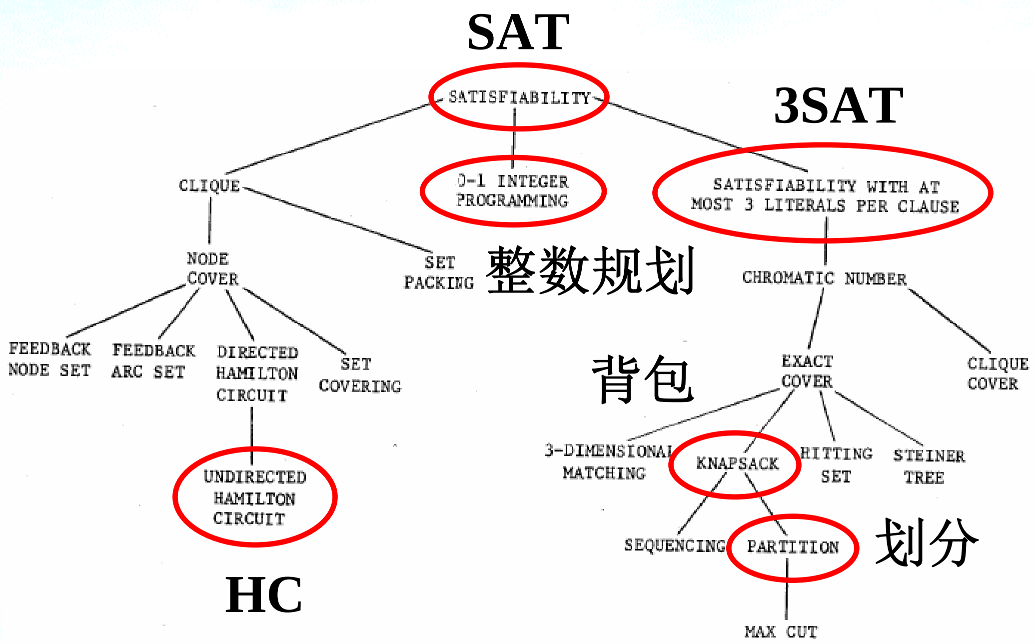
（1948－ ）

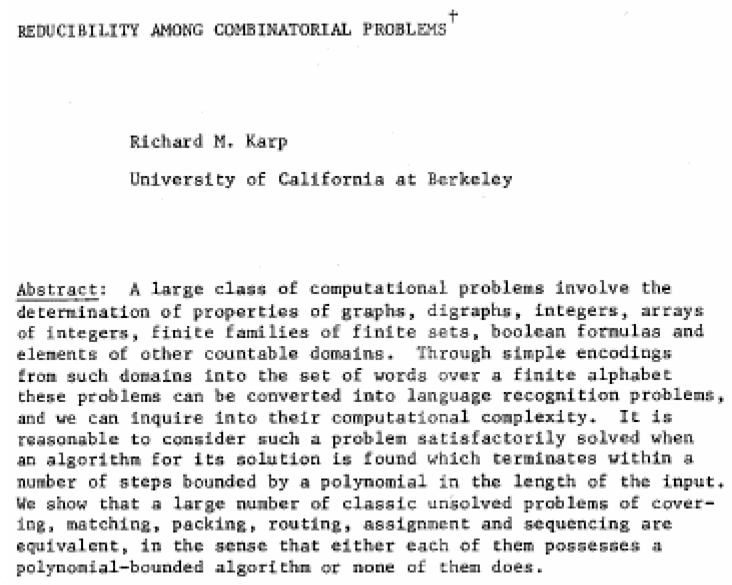
苏联计算机学家

44

Karp归约

浙江大学
ZheJiang University
数学建模





REDUCIBILITY AMONG COMBINATORIAL PROBLEMS†

Richard M. Karp

University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

Karp RM. Reducibility Among Combinatorial Problems, *Proceedings of a Symposium on the Complexity of Computer Computations*, 85-103, 1972

45

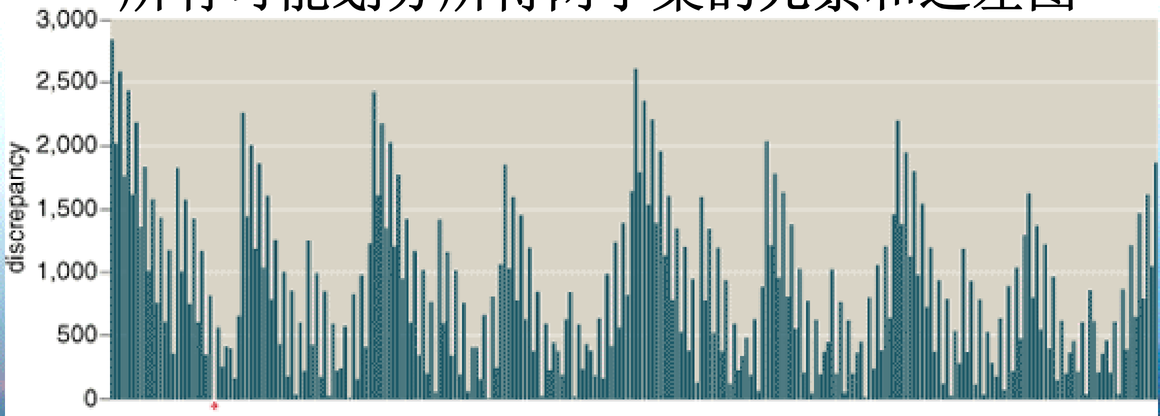
划分问题

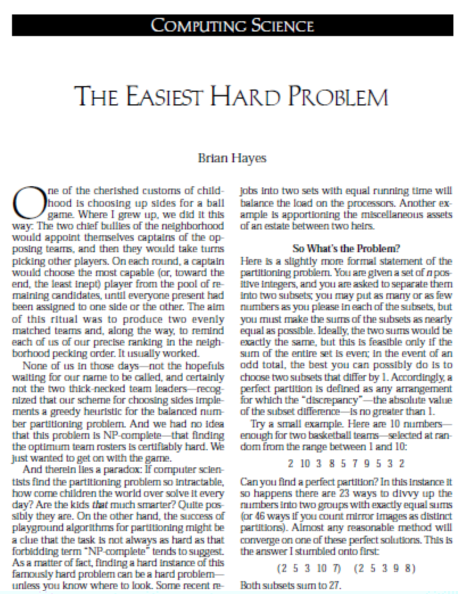
- 划分问题（**Partition**）
  - 给定一正整数集 $A=\{a_1,a_2,\cdots,a_n\}$，问是否存在子集 A_1,A_2，使得 $A=A_1\cup A_2$，$A_1\cap A_2=\phi$，且满足 $\sum_{a_i\in A_1}a_i=\sum_{a_i\in A_2}a_i=\frac{1}{2}\sum_{j=1}^{n}a_j$

$$A=\{484,114,205,288,506,503,201,127,410\}$$

$$A_1=\{410,506,503\},\ A_2=\{484,114,205,288,201,127\}$$

所有可能划分所得两子集的元素和之差图





COMPUTING SCIENCE

THE EASIEST HARD PROBLEM

Brian Hayes

One of the cherished customs of childhood is choosing up sides for a ball game. Where I grew up, we did it this way: The two chief bullies of the neighborhood would appoint themselves captains of the opposing teams, and then they would take turns picking other players. On each round, a captain would choose the most capable (or, toward the end, the least inept) player from the pool of remaining candidates, until everyone present had been assigned to one side or the other. The aim of this ritual was to produce two evenly matched teams and, along the way, to remind each of us of our precise ranking in the neighborhood pecking order. It usually worked.

None of us in those days—not the hopefuls waiting for our name to be called, and certainly not the two thick-necked team leaders—recognized that our scheme for choosing sides implements a greedy heuristic for the balanced number partitioning problem. And we had no idea that this problem is NP-complete—that finding the optimum team rosters is certifiably hard. We just wanted to get on with the game.

And therein lies a paradox: If computer scientists find the partitioning problem so intractable, how come children the world over solve it every day? Are the kids *that* much smarter? Quite possibly they are. On the other hand, the success of playground algorithms for partitioning might be a clue that the task is not always as hard as that forbidding term "NP-complete" tends to suggest. As a matter of fact, finding a hard instance of this famously hard problem can be a hard problem—unless you know where to look. Some recent re-

jobs into two sets with equal running time will balance the load on the processors. Another example is apportioning the miscellaneous assets of an estate between two heirs.

So What's the Problem?

Here is a slightly more formal statement of the partitioning problem. You are given a set of *n* positive integers, and you are asked to separate them into two subsets; you may put as many or as few numbers as you please in each of the subsets, but you must make the sums of the subsets as nearly equal as possible. Ideally, the two sums would be exactly the same, but this is feasible only if the sum of the entire set is even; in the event of an odd total, the best you can possibly do is to choose two subsets that differ by 1. Accordingly, a perfect partition is defined as any arrangement for which the "discrepancy"—the absolute value of the subset difference—is no greater than 1.

Try a small example. Here are 10 numbers—enough for two basketball teams—selected at random from the range between 1 and 10:

2 10 3 8 5 7 9 5 3 2

Can you find a perfect partition? In this instance it so happens there are 23 ways to divvy up the numbers into two groups with exactly equal sums (or 46 ways if you count mirror images as distinct partitions). Almost any reasonable method will converge on one of these perfect solutions. This is the answer I stumbled onto first:

(2 5 3 10 7) (2 5 3 9 8)

Both subsets sum to 27.

Hayes B. The easiest hard problem. ***American Scientist*****, 90(2), 113-117, 2002.**

子问题

- 在某问题 Π 的实例构成上增加一些限制就得到的一个子问题（**subproblem**）Π'
  - Π'的实例集包含在 Π 的实例集中，Π'的答案为“是”（“否”）的实例集恰为 Π 的答案为“是”（“否”）的实例集与 Π'的实例集之交
- 子问题的计算复杂性
  - 若 $\Pi \in \mathcal{P}$，则 $\Pi' \in \mathcal{P}$，但反之不然
  - 若 Π'是$\mathcal{NP}$-完全的，则 Π 也是$\mathcal{NP}$-完全的，但反之不然
  - 希望寻找“最特殊”的$\mathcal{NP}$–完全子问题和“最一般”的$\mathcal{P}$子问题

47

子集和问题

浙江大学 ZheJiang University

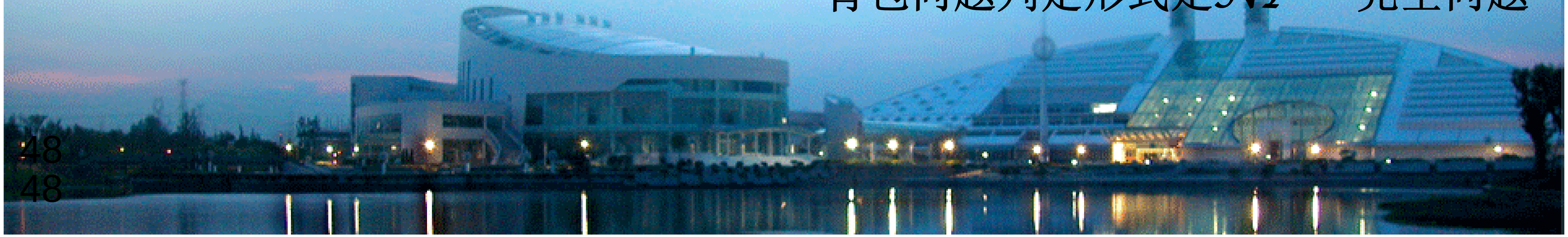
数学建模

- 子集和问题（**Subset Sum**）
  - 给定正整数集 $A=\{a_1,a_2,\cdots,a_n\}$和数 B，问是否存在子集$A_1\subseteq A$，使得 $\sum_{a_i\in A_1} a_i = B$
  - 取 $B=\frac{1}{2}\sum_{j=1}^{n} a_j$，划分问题成为子集和问题的子问题
- 子集和问题的优化形式

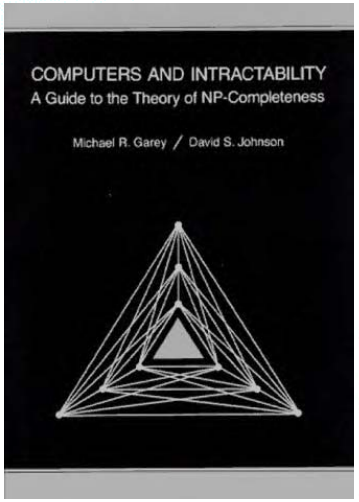
子集和是$\mathcal{NP}$ −完全问题

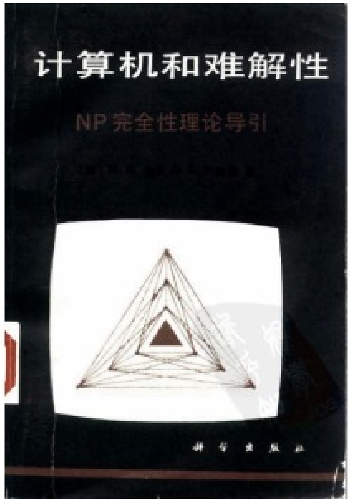
  - 求子集 $A_1\subseteq A$，使得 $\sum_{a_i\in A_1} a_i \le B$ 且 $\sum_{a_i\in A_1} a_i$ 尽可能大
  - 若背包问题物品 j 的价值与大小均为 a_j，容量为 B，则子集和问题优化形式成为背包问题优化形式的子问题

背包问题判定形式是$\mathcal{NP}$ −完全问题

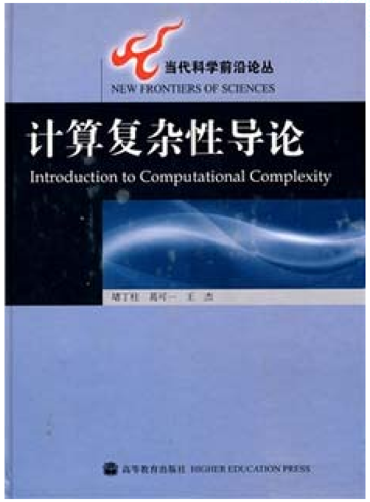


参考资料

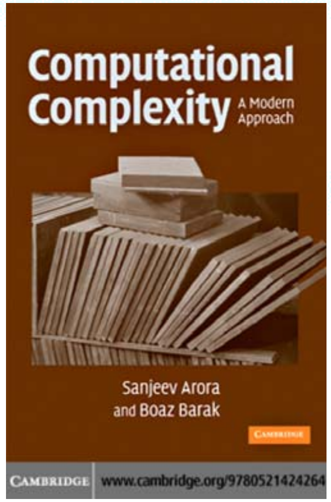




Garey MR, Johnson DS. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979.（中译本：计算机和难解性： NP 完全性理论导引. 张立昂，沈泓译，科学出版社，1990.）



堵丁柱，葛可一，王杰，计算复杂性导论，高等教育出版社，2002.



Arora S, Barak B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

$\mathcal{NP}$ –难问题汇编

浙江大学
ZheJiang University
数学建模

A compendium of NP optimization problems

Editors:
[Pierluigi Crescenzi], and [Viggo Kann]
Subeditors:
Magnús Halldórsson (retired)
Graph Theory: Covering and Partitioning, Subgraphs and Supergraphs, *Sets and Partitions*.
Marek Karpinski
Graph Theory: Vertex Ordering, *Network Design*: Cuts and Connectivity.
Gerhard Woeginger
Sequencing and Scheduling.

This is a continuously updated catalog of approximability results for NP optimization problems. The compendium is also a part of the book Complexity and Approximation. The compendium has not been updated for a while, so there might exist recent results that are not mentioned in the compendium. If you happen to notice such a missing result, please report it to us using the web forms.

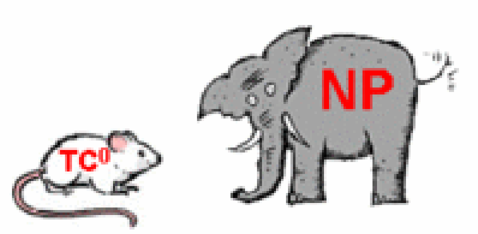
You can use web forms to report new problems, new results on existing problems, updates of references or errors.

http://www.nada.kth.se/~viggo/problemlist/compendium.html

Complexity Zoo

数学建模

- **There are now 496 classes and counting**



All Classes

Complexity classes by letter: Symbols - A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Lists of related classes: Communication Complexity - Hierarchies - Nonuniform

Symbols

0-1-NP_C - $1NAuxPDA^p$ - 2-EXP - 3SUM-hard - $\#AC^0$ - #L - #L/poly - #GA - #P - #W[t] - ⊕EXP - ⊕L - ⊕L/poly - ⊕P - $\oplus SAC^0$ - $\oplus SAC^1$

A

A_0PP - AC - AC^0 - $AC^0[m]$ - AC^1 - ACC^0 - AH - AL - ALL - ALOGTIME - AlgP/poly - Almost-NP - Almost-P - Almost-PSPACE - AM - AM_{EXP} - AM ∩ coAM - AM[polylog] - AmpMP - AmpP-BQP - AP - APP - APX - ATIME - AUC-SPACE(f(n)) - AuxPDA - AVBPP - AvgE - AvgP - AW[P] - AWPP - AW[SAT] - AW[*] - AW[t] - AxP - AxPP

B

βP - BH - $BP_d(P)$ - BPE - BPEE - $BP_HSPACE(f(n))$ - BPL - BP•NP - BPP - BPP^{cc} - BPP_k^{cc} - BPP^{KT} - BPP/log - BPP/mlog - BPP//log - BPP/rlog - BPP-OBDD - BPP_{path} - BPQP - BPSPACE(f(n)) - BPTIME(f(n)) - BQNC - BQNP - BQP - BQP/log - BQP/poly - BQP/mlog - BQP/mpoly - BQP/qlog - BQP/qpoly - BQP-OBDD - BQPSPACE - BQP_{CTC} - BQP_{tt}/poly - BQTIME(f(n)) - k-BWBP

NL: Nondeterministic Logarithmic-Space

Has the same relation to L as NP does to P.

In a breakthrough result, was shown to equal coNL [Imm88] [Sze87]. (Though contrast to mNL.)

Is contained in LOGCFL [Sud78], as well as NC^2.

Is contained in UL/poly [RA00].

Deciding whether a bipartite graph has a perfect matching is hard for NL [KUW86].

NL can be defined in a logical formalism as SO(krom) and also as FO(tc), reachability in directed graph is NL-Complete under FO-reduction.

https://complexityzoo.uwaterloo.ca/

组合优化问题的求解方法

数学建模

- 组合优化问题
  - 设计多项式时间算法（$\mathcal{P}$ 问题）
    - 进一步改进算法性能
  - 复杂性未决问题
  - 证明为$\mathcal{NP}$ -难
    - 证明难近似性
  - 复杂性未决问题、证明为$\mathcal{NP}$ -难
    - 研究特殊可解性
    - 在指数时间内求最优解
    - 在多项式时间内求近似解

$\mathcal{NP}$-难问题的求解方法

数学建模

- 在指数时间内求最优解
  - 整数规划法
  - 分支定界法
  - 动态规划法
  - ···
- 在多项式时间内求近似解
  - 近似算法
  - 近似方案
  - 启发式算法
    - 贪婪法
    - 线性规划松弛
    - 局部搜索法
    - · · ·
    - Meta-heuristic

背包问题

数学建模

- 现有n件物品，物品j的价值为p_j，大小为w_j，背包容量为C。要求选择若干物品放入背包，在放入背包物品大小之和不超过背包容量前提下使放入背包物品价值之和尽可能大
  - 决策变量 $x_j=\begin{cases}1 & \text{放入第} j \text{种物品}\\ 0 & \text{其他}\end{cases} \quad j=1,2,\cdots,n$
  - 整数规划

$$\max \sum_{j=1}^{n} p_j x_j \qquad \text{放入背包物品价值之和}$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C \qquad \text{放入背包物品大小之和不超过} C$$

$$x_j = 0,1,\ \ j=1,\cdots,n$$

指派问题

数学建模

- 设有 n 项任务需分配给 n 位员工，每人完成其中一项，员工 i 完成任务 j 所需时间为 c_{ij}，如何分配可使完成所有任务所用总时间最少
  - 决策变量 $x_{ij}=\begin{cases}1, & \text{员工 } i \text{ 被分配完成工作 } j \\ 0, & \text{其他}\end{cases}$ $\quad i,j=1,2,\cdots,n$
  - 整数规划

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$ 完成所有任务所用总时间

$$s.t. \quad \sum_{j=1}^{n} x_{ij}=1,\ i=1,\cdots,n$$ 每位员工完成一项工作

$$\sum_{i=1}^{n} x_{ij}=1,\ j=1,\cdots,n$$ 每项工作由一位员工完成

$$x_{ij}=0 \text{ 或 } 1,\ i,\ j=1,\cdots,n$$

整数规划 $\in \mathcal{NP}\text{-}C$

指派问题 $\in \mathcal{P}$?

指派问题

- 记决策变量向量为 $\mathbf{x}=(x_{11},x_{12},\cdots,x_{1n},x_{21},\cdots,x_{2n},\cdots,x_{n1},\cdots,x_{nn})^T$ ，整数规划 $\min\{\mathbf{cx}\mid\mathbf{Ax}=\mathbf{b},\mathrm{x}_{ij}\in\{0,1\}\}$ 的系数矩阵为

$$\mathbf{A}=\begin{pmatrix} 1\ 1\ \cdots\ 1 & 0\ 0\ \cdots\ 0 & & 0\ 0\ \cdots\ 0 \\ 0\ 0\ \cdots\ 0 & 1\ 1\ \cdots\ 1 & \cdots & 0\ 0\ \cdots\ 0 \\ \vdots\ \vdots\ \ \vdots\ \ \vdots & \vdots\ \vdots\ \ \vdots\ \ \vdots & & \vdots\ \vdots\ \ \vdots\ \ \vdots \\ 0\ 0\ \cdots\ 0 & 0\ 0\ \cdots\ 0 & & 1\ 1\ \cdots\ 1 \\ & & & \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ & & & \end{pmatrix}$$

$$\begin{aligned} \min & \sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij} \\ s.t. & \sum_{j=1}^{n}x_{ij}=1,\ i=1,\cdots,n \\ & \sum_{i=1}^{n}x_{ij}=1,\ j=1,\cdots,n \\ & 0\le x_{ij}\le 1\quad i,j=1,\cdots,n \end{aligned}$$

- 若矩阵的所有子式均为**0**或 ± 1，则称该矩阵为全幺模（**totally unimodular**）矩阵
- 系数矩阵为全幺模矩阵的整数规划的松弛线性规划的最优解必为整数解

TSP问题

浙江大学 ZheJiang University

数学建模

- 决策变量 $x_{ij}=\begin{cases}1, & 离开城市\ i\ 后到达城市\ j\\ 0, & 其他\end{cases}\quad i,j=1,2,\cdots,n$
- 整数规划

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

决策变量能表示环游，同时能计算环游长度

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

离开城市 i 后到达另一个城市

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

从一个城市来到城市 j

$$x_{ij}=0,1,\ i,j=1,\cdots,n$$

指派问题 $\in \mathcal{P}$

TSP $\in \mathcal{NP}\text{-}C$?

问题可行解与其整数规划可行解之间未一一对应

TSP问题

- **TSP**环游不允许出现经过城市数小于 n 的子环游
- 若存在一相继经过城市 $i_1, i_2, \cdots, i_k$ 的子环游，则

$$x_{i_1 i_2} = x_{i_2 i_3} = \cdots = x_{i_{k-1} i_k} = x_{i_k i_1} = 1$$

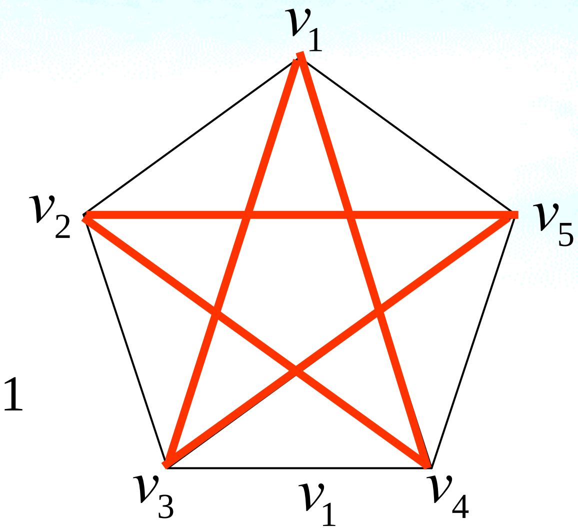
$$\sum_{i,j \in \{i_1, i_2, \cdots, i_k\}} x_{ij} \geq k$$

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

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

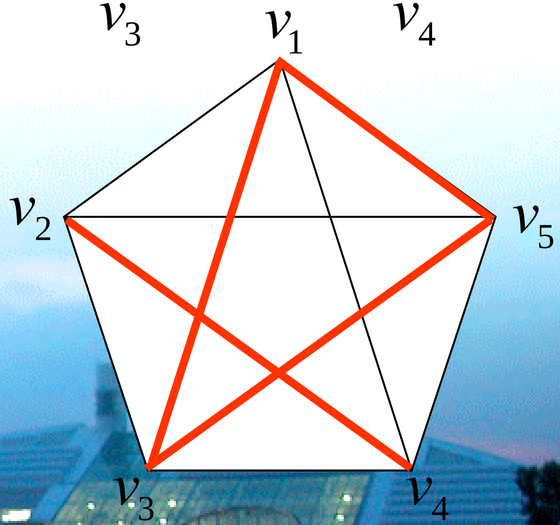
$$x_{13} = x_{35} = x_{52} = x_{24} = x_{41} = 1$$

$$x_{ij} = 0, (i,j) \notin \{(1,3),(3,5),(5,2),(2,4),(4,1)\}$$



$$x_{13} = x_{35} = x_{51} = x_{24} = x_{42} = 1$$

$$x_{ij} = 0, (i,j) \notin \{(1,3),(3,5),(5,1),(2,4),(4,2)\}$$



TSP问题的整数规划

浙江大学 ZheJiang University 数学建模

- 为使整数规划的解不存在任意子环游，需增加约束以消除子环游

$$\sum_{i,j\in S} x_{ij} \le |S|-1,\ \ \forall \varnothing \ne S \subset \{2,3,\cdots,n\}$$

  - 增加约束后的整数规划约束个数为 $O(2^n)$

- 增加 $n-1$ 个实决策变量 $u_2,u_3,\cdots,u_n$ 和 $(n-1)^2$ 个约束

$$(n-1)x_{ij} + u_i - u_j \le n-2,\ \ i,j = 2,3,\cdots,n$$

也可消除子环游

Miller CE, Tucker AW, Zemlin RA. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7, 326-329, 1960

MTZ整数规划

浙江大学
ZheJiang University
数学建模

- 若解含有子环游，则约束不被满足
  - 若含有子环游，则必有一子环游 $(i_1\, i_2 \cdots i_k)$不经过城市 1
  - 若该子环游仅含一个城市 i ，则 $x_{ii}=1$

$$(n-1)x_{ii}+u_i-u_i=n-1>n-2$$

矛盾

  - 若该子环游至少含有两个城市，则 $x_{i_j i_{j+1}}=1\left(i_{k+1}=i_1\right)$,

$$(n-1)x_{i_j i_{j+1}}+u_{i_j}-u_{i_{j+1}}=n-1+u_{i_j}-u_{i_{j+1}},\ j=1,2,\cdots,k$$

$$\sum_{j=1}^{k}\left((n-1)x_{i_j i_{j+1}}+u_{i_j}-u_{i_{j+1}}\right)=\sum_{j=1}^{k}\left(n-1+u_{i_j}-u_{i_{j+1}}\right)=k(n-1)>k(n-2)$$

矛盾

$$\boxed{(n-1)x_{ij}+u_i-u_j\le n-2,\ \ i,j=2,3,\cdots,n}$$

MTZ整数规划

浙江大学 ZheJiang University 数学建模

- 对任意可行环游，存在一组可行解满足约束
  - 对任意可行环游，选定城市 1 为起点，若城市 i 是环游中的第 k 个经过的城市，取 $u_i = k$，$2 \le u_i \le n$，x_{ij} 的取值同前定义
  - 若对某个 i, j 组合，$x_{ij} = 0$，则

$$(n-1)x_{ij} + u_i - u_j = u_i - u_j \le n-2$$

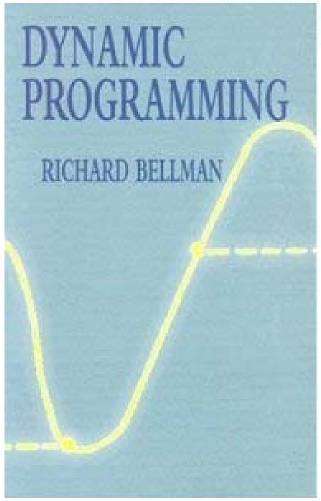
  - 若对某个 i, j 组合，$x_{ij} = 1$，则 $u_i - u_j = -1$，

$$(n-1)x_{ij} + u_i - u_j = (n-1) - 1 = n-2$$

$$\boxed{(n-1)x_{ij} + u_i - u_j \le n-2,\ \ i, j = 2, 3, \cdots, n}$$

动态规划

- 动态规划（**dynamic programming，DP**）是求解多阶段决策优化问题的一种数学方法和算法思想
- 动态规划的基本思路是将需求解的实例转化为规模较小的实例，并利用递推关系导出两者最优解之间的关系，从而可由初始条件出发逐步求得最优解



Bellman RE，*Dynamic Programming*, Princeton University Press, 1957

Richard Ernest Bellman
(1920-1984)
美国运筹学家

最优化原理

数学建模

- 最优化原理（**Principle of Optimality**）
  - 一个过程的最优决策 $(x_1^*, x_2^*, \cdots, x_n^*)$具有如下的性质：无论初始状态 s_0 及初始决策 x_1如何确定，之后的决策$(x_2^*, \cdots, x_n^*)$对于以 x_1^*所造成的状态 s_1^*为初始状态的后部过程而言，必为最优策略

$$s_0 \xrightarrow{x_1} \boxed{s_1 \xrightarrow{x_2} s_2 \longrightarrow \cdots \xrightarrow{x_n} s_n} \qquad f_n(s_0; x_1, x_2, \cdots, x_n)$$

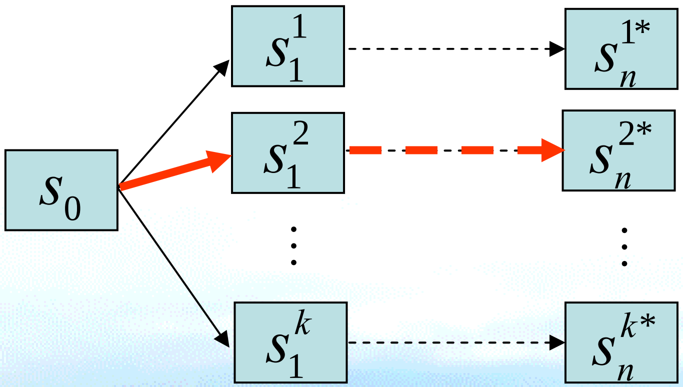
$$f_n(s_0; x_1^*, x_2^*, \cdots, x_n^*) = f_1(s_0; x_1^*) + f(s_1^*; x_2^*, \cdots, x_n^*)$$

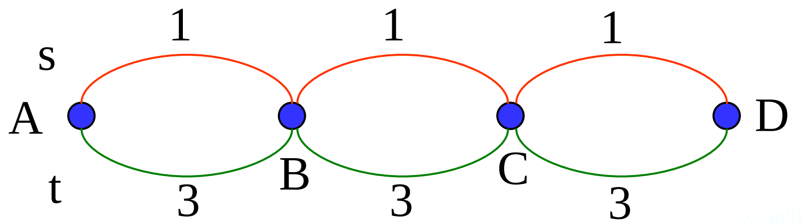
$$f_n(s_0; x_1^*, \hat{x}_2, \cdots, x_n) = f_1(s_0; x_1^*) + f(s_1^*; \hat{x}_2, \cdots, x_n)$$

PRINCIPLE OF OPTIMALITY. *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

最优化原理

- 最优化原理（**Principle of Optimality**）：一个 n 阶段过程的最优策略可以这样构成：首先求出以初始决策 x_1 造成的状态 s_1 为初始状态的 $n-1$ 阶段子过程的最优策略，然后在附加第一阶段效益（或费用）的情形下，从所有可能初始决策得到的解中选择最优者





最短路的子路也是最短路

路长定义为边长的模4加法时，t为A到D的最短路，但C到D的最短路仍为s，最优化原理不再成立

背包问题的动态规划

浙江大学 ZheJiang University
数学建模

- 依次考虑所有物品，每个物品对应动态规划的一个阶段，可能决策为该物品放入与不放入两类
- 构造一系列背包问题实例 $I(k,w), 0 \le k \le n, 0 \le w \le C$，其中背包容量为 w，物品数为 k，包含原实例中的前 k 个物品
- 记 $C^*(k,w)$ 为实例 $I(k,w)$ 的最优解，原实例的最优值即为 $C^*(n,C)$

递推关系

浙江大学 ZheJiang University

数学建模

- $C^*(k,w)$ 满足递推关系

$$C^*(k,w)=\begin{cases} C^*(k-1,w) & 若 w_k > w \\ \max\left\{C^*(k-1,w),C^*(k-1,w-w_k)+p_k\right\} & 若 w_k \le w \end{cases}$$

  - 若 $w_k > w$，则物品 k 不能放入容量为 w 的背包中，因此 $I(k,w)$ 的最优值与 $I(k-1,w)$ 的最优值必相同
  - 若 $w_k \le w$，则物品 k 可以放入容量为 w 的背包中，$I(k,w)$ 的最优解有两种可能
    - 若物品 k 未放入背包， $I(k,w)$ 的最优值必与 $I(k-1,w)$ 的最优值相同
    - 若物品 k 放入背包，背包剩余容量为 $w-w_k$，可放入的物品必为 $I(k-1,w-w_k)$ 的最优解中放入的物品

动态规划DPS

数学建模

- $C^*(k,w)$ 满足初始条件

$$C^*(0,w)=0,\ w=0,\cdots,C,\ C^*(k,0)=0,\ k=0,\cdots,n$$

- 由初始条件和递推关系可逐步求得$I(n,C)$，外层循环为k自0到n，内层循环为w自0到C
- 动态规划**DPS**的时间复杂性为$O(nC)$，背包问题的实例规模为$n+\lceil \log_2 L \rceil$，其中$L=\max\left\{\max\limits_{j=1,\cdots,n} p_j,\max\limits_{j=1,\cdots,n} w_j, C\right\}$，因此**DPS**是伪多项式时间算法，背包问题是普通意义下的$\mathcal{NP}$-完全问题

动态规划DPS

浙江大学 ZheJiang University

数学建模

- 背包问题实例

$C=5, n=4$

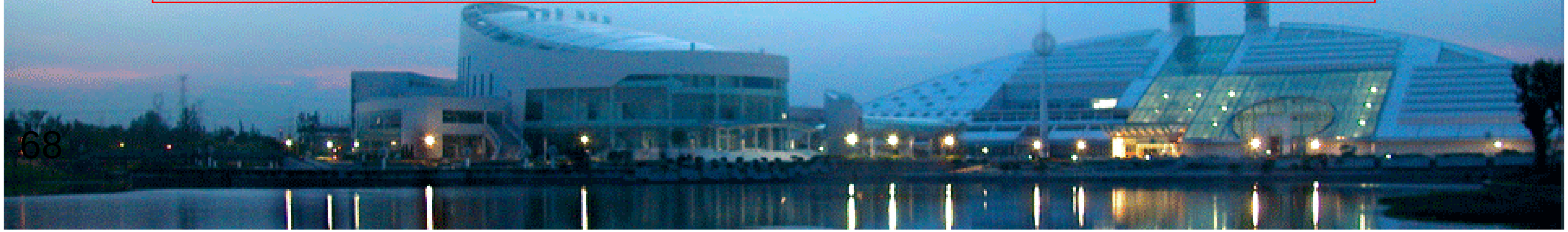
$p_1=3, p_2=4, \ p_3=5, p_4=6$

$w_1=2, w_2=3, w_3=4, w_4=5$

- 最优值为 7，最优解为物品1，2 放入背包

$w \backslash k$	0	1 +3	2 +4	3 +5	4 +6
0	0	0	0	0	0
1	0	0	0	0	0
2 w_1	0	3	3	3	3
3 w_2	0	3	4	4	4
4 w_3	0	3	4	5	5
5 w_4	0	3	7	7	7

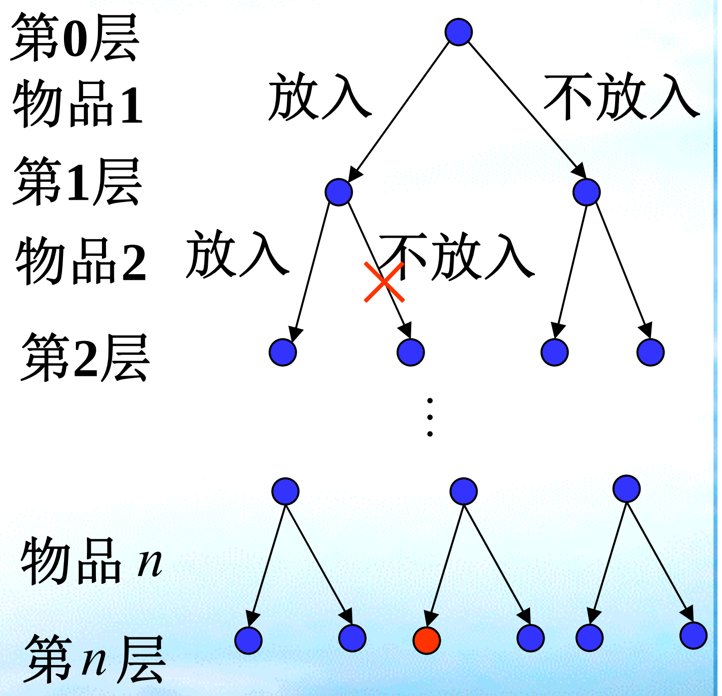
$$C^*(k,w)=\begin{cases} C^*(k-1,w) & 若 w_k > w \\ \max\left\{C^*(k-1,w), C^*(k-1,w-w_k)+p_k\right\} & 若 w_k \le w \end{cases}$$



分枝定界法

数学建模

- 背包问题的分枝定界法
  - 实例求解过程可用一颗高度为 n 的二叉树表示。各层依次对应一个物品，最后一层的每个节点对应一个解
  - 分枝：每个节点有两条出边连结两个子节点，分别代表该节点下一层对应物品放入背包与不放入背包
  - 剪枝
    - 该枝不存在可行解
      - 按该枝含义确定放入背包的物品大小之和超过背包容量
    - 该枝不存在最优解　　如何提前判断



分枝定界法

数学建模

- 定界　　下界对所有节点均有效，上界仅对该枝有效
  - 任一个已获得的可行解的目标值均是最优值的下界
  - 对每一枝，求该枝所有可行解目标值的上界，若该上界不大于下界，则该枝不存在更好的可行解
    - 按该枝含义，所有确定已放入物品和所有未确定物品价值之和

      上界越小越好，下界越大越好

      上界和下界的计算尽量简单
    - 该枝对应的整数规划的松弛线性规划的最优值

      只考虑未确定物品

松弛线性规划

- 松弛线性规划最优解 $\mathbf{x}=(x_1^*,x_2^*,\cdots,x_n^*)^T$ 的性质
  - $\sum_{j=1}^{n} w_j x_j^* = C$
    - 若不然，必存在 $i, x_i^* < 1$。令 $x_i' = x_i^* + \varepsilon$ 可得一更好的解
  - 若 $\frac{p_i}{w_i} > \frac{p_k}{w_k}$，则若 $x_i^* < 1$，则 $x_k^* = 0$
    - 若不然，令 $x_i' = x_i^* + \frac{w_k}{w_i}\varepsilon, x_k' = x_k^* - \varepsilon$，

$$\sum_{j=1}^{n} p_j x_j' - \sum_{j=1}^{n} p_j x_j^* = p_i \frac{w_k}{w_i}\varepsilon - p_k \varepsilon > 0$$ 矛盾

  - $\frac{p_j}{w_j}$ 称为物品 j 的价值密度，最优解应优先放入价值密度大的物品

$$\max \sum_{j=1}^{n} p_j x_j$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C$$

$$x_j = 0,1, \quad j = 1,\cdots,n$$

$$\max \sum_{j=1}^{n} p_j x_j$$

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C$$

$$0 \le x_j \le 1, \quad j = 1,\cdots,n$$

$$p_j > 0, j = 1,\cdots,n, \sum_{j=1}^{n} w_j > C$$

松弛线性规划

- 假设物品按价值密度非增顺序排列，即 $\frac{p_1}{w_1} \ge \frac{p_2}{w_2} \ge \cdots \ge \frac{p_n}{w_n}$
  - 若 $\frac{p_i}{w_i} = \frac{p_k}{w_k}$，可将物品 i 和物品 k 合并为一个物品
- 记 $j = \min\left\{ k \,\middle|\, \sum_{i=1}^{k} w_i > C \right\}$ 为按价值密度非增顺序第一个不能全部放入背包的物品
- 松弛线性规划的最优解为

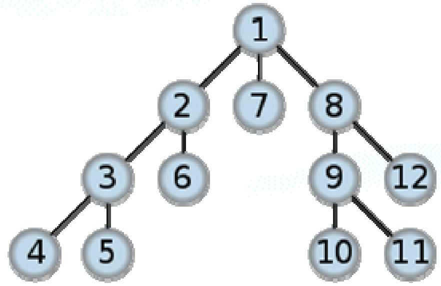
$$x_i^* = 1, i = 1, 2, \cdots, j-1, \quad x_j^* = \frac{1}{w_j}\left(C - \sum_{i=1}^{j-1} w_i \right), \quad x_i^* = 0, i = j+1, \cdots, n$$

- 松弛线性规划的最优值为 $\sum_{i=1}^{j-1} p_j + \frac{p_j}{w_j}\left(C - \sum_{i=1}^{j-1} w_i \right)$。由于物品价值均为整数，背包（整数规划）最优值的上界也可取为 $UB_1 = \sum_{i=1}^{j-1} p_i + \left\lfloor \frac{p_j}{w_j}\left(C - \sum_{i=1}^{j-1} w_i \right) \right\rfloor$

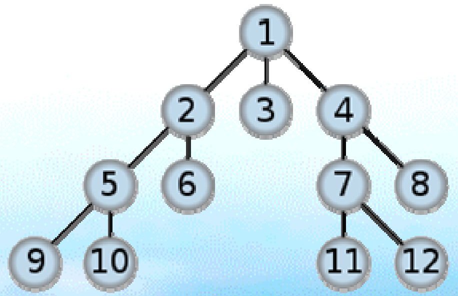
分枝定界法

数学建模

- 从根节点开始，按深探法（**Depth-first search, DFS**）或广探法（**Breadth-first search, BFS**）给定的顺序检查每个节点
  - 对每个节点，按节点含义计算上界和下界
  - 若下界大于当前下界，则修正当前下界
  - 若当前节点不满足剪枝条件，则进行分枝
- 若所有节点都检查完毕且不可再分枝，当前下界即为实例的最优值
- 分枝定界法是指数时间算法，但实际效果未必劣于伪多项式时间动态规划



深探法



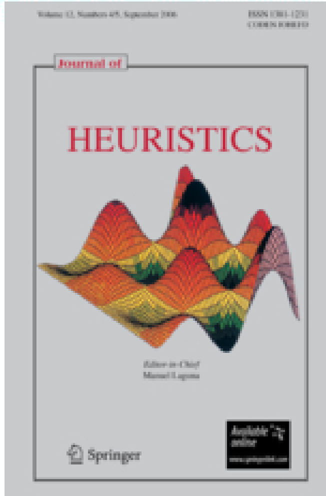
广探法

浙江大学 ZheJiang University

数学建模

近似算法

- 可给出$\mathcal{NP}$－难问题任一实例最优解的算法总需要指数时间。较为现实的思路是“用精度换时间”，在多项式时间或可接受的实际运行时间内得到一个目标值与最优值较为接近的可行解
  - 近似算法（**approximation algorithm**）：算法的时间复杂性可通过分析确定（一般要求多项式时间），算法给出的近似解与最优解目标值之间的差距可通过证明严格估计
  - 启发式算法（**heuristic**）：无法说明算法的时间复杂性，或无法估计算法给出的近似解与最优解目标值之间的差距



Journal of Heuristics

$(\epsilon\upsilon\rho\iota\sigma\kappa\epsilon\iota\nu)$

to find

最坏情况界

浙江大学 ZheJiang University

数学建模

- 设Π是一个极小化优化问题，A是它的一个算法。对Π的任意实例 I，算法 A 给出一个可行解，其目标值为 $C^A(I)$，实例 I 的最优目标值为 $C^*(I)$
- 称 $r_A = \inf\{r \geq 1 \mid C^A(I) \leq rC^*(I), \forall I\}$或 $r_A = \sup_I \left\{\frac{C^A(I)}{C^*(I)}\right\}$ 为算法的最坏情况界（**worst-case ratio**）
  - 若算法 A的最坏情况界为 r_A，则对该问题的任意实例 I，均有 $C^A(I) \leq r_A C^*(I)$
  - 最坏情况界越接近于1，说明算法给出的可行解目标值越接近于最优值，算法近似性能越好

背包问题的近似算法

数学建模

- 对极大化目标的优化问题，定义 A 的最坏情况界为 $r_A = \inf\{r \ge 1 \mid C^*(I) \le rC^A(I), \forall I\}$
- 基于贪婪（**Greedy**）思想的算法
  - 将物品按价值密度非增的顺序排列，即有 $\frac{p_1}{w_1} \ge \frac{p_2}{w_2} \ge \cdots \ge \frac{p_n}{w_n}$
  - 按上述顺序将物品依次放入背包，直至第一个不能放入的物品为止
- 记 $j = \min\left\{k \mid \sum_{i=1}^{k} w_i > C\right\}$，算法将前 $j-1$ 个物品放入背包 $C^G = \sum_{i=1}^{j-1} p_j$

物品	1	2
价值	2	$2M$
大小	1	$2M$
背包容量 $2M$		

$C^G(I_1) = 2$，$C^*(I_1) = 2M$

$$\frac{C^*(I_1)}{C^G(I_1)} = M \to \infty (M \to \infty)$$

证明最坏情况界之前可通过构造实例对最坏情况界作出估计

背包问题的近似算法

浙江大学 ZheJiang University

数学建模

- 算法改进
  - 将物品 j 之后可以放入背包的物品放入背包

改进无效

- 基于复合思想的改进算法
  - 运行基于贪婪思想的算法 G
  - 将 G 所得目标值与最大物品价值 $p_{\max}$ 进行比较，取优者作为输出

$$C^C(I) = \max\left\{\sum_{i=1}^{j-1} p_j, p_{\max}\right\}$$

物品	1	2
价值	2	$2M$
大小	1	$2M$
背包容量 $2M$		

$C^G(I_1) = 2$，$C^*(I_1) = 2M$

$$\frac{C^*(I_1)}{C^G(I_1)} = M \to \infty (M \to \infty)$$

复合算法

浙江大学 ZheJiang University

数学建模

- 复合算法的最坏情况界为 **2**
  - 用松弛线性规划的最优值作为最优值的上界

$$C^*(I) \le \sum_{i=1}^{j-1} p_i + \frac{p_j}{w_j}\left(C - \sum_{i=1}^{j-1} w_i\right)$$

$$\le \sum_{i=1}^{j-1} p_i + \frac{p_j}{w_j} w_j \le \sum_{i=1}^{j-1} p_i + p_{\max}$$

$$\frac{C^*(I)}{C^C(I)} \le \frac{\sum_{i=1}^{j-1} p_i + p_{\max}}{\max\left\{\sum_{i=1}^{j-1} p_i, p_{\max}\right\}} \le 2$$

物品	1	2	3
价值	1	M	M
大小	1	M	M
背包容量 $2M$			

$$C^C(I) = M + 1, C^*(I) = 2M$$

$$\frac{C^*(I)}{C^C(I)} = \frac{2M}{M+1} \to 2 (M \to \infty)$$

$$C^C(I) = \max\left\{\sum_{i=1}^{j-1} p_j, p_{\max}\right\} \quad j = \min\left\{k \mid \sum_{i=1}^{k} w_i > C\right\}$$

平均情况界

- 假设实例中的数据服从一定概率分布，$\frac{C^A(I)}{C^*(I)}$为一随机变量，其期望$\mathbb{E}\left(\frac{C^A(I)}{C^*(I)}\right)$即为算法 A 的平均情况界（**average-case ratio**）
  - 平均情况界依赖于所假设的实例中数据所服从的概率分布
  - 在多数情况下，缺乏足够的依据来判断一个实例中的数据来自何种分布
  - 即便已得到某一算法在某个分布下的平均情况界，对来自该分布的一个实例 I，也无法对$\frac{C^A(I)}{C^*(I)}$的值作出估计
  - 平均情况界的证明比最坏情况界更为复杂

近似方案

浙江大学 ZheJiang University

数学建模

- 算法族$\{A_\varepsilon\}$称为多项式时间近似方案（**Polynomial Time Approximation Scheme，PTAS**），若对任意给定的ε，算法A_ε的最坏情况界为$1+\varepsilon$，且A_ε的时间复杂性为$f(n)=O(p(n))$，这里$p(x)$为一多项式
- 算法族$\{A_\varepsilon\}$称为完全多项式时间近似方案（**Fully Polynomial Time Approximation Scheme，FPTAS**），若对任意给定的ε，算法A_ε的最坏情况界为$1+\varepsilon$，且A_ε的时间复杂性为$f(n,\varepsilon)=O\left(p\left(n,\frac{1}{\varepsilon}\right)\right)$，这里$p(x,y)$为一二元多项式
  - 时间复杂性为$O\left(\frac{n}{\varepsilon^2}\right)$的算法族可以成为一**FPTAS**，时间复杂性为$O\left(n^{\frac{1}{\varepsilon}}\right)$的算法族可以成为一**PTAS**，但不能成为一**FPTAS**
- 近似方案具有几乎最好的近似性能，但算法实现通常较为复杂，实际运算时间可能很长

A_ε

启发式算法

数学建模

- 启发式算法通过数值模拟的方法来衡量算法的性能，从而免去了理论证明的困难和局限，可以充分地利用经验和技巧，自由地调整算法的步骤和参数，设计出性能尽可能好的算法
  - 快速
  - 准确
  - 稳健
  - 简单
  - 新问题或重要问题
  - 设计思想有创新
  - 可移植性强
  - 有益于理论研究
- 算法性能测试
  - 解的质量
  - 计算资源
  - 稳健性分析
  - 真实性和可重复性
  - 结果、图、表可读性
- 试验实例来源
  - 现实数据
  - 随机生成实例
  - 实例库

启发式算法

- **Zanakis, S. H., Evans, J. R., Heuristic "Optimization": Why, When, and How to Use It, *Interfaces*, 11, 84-91, 1981.**
- **Barr, R. S.. Golden, B. L., Kelly, J. P., Resende, M. G. C., Stewart W. R., Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics*, 1, 9-32, 1995**
- **Rardin, R. L., Uzsoy, R., Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial, *Journal of Heuristics*, 7, 261-304, 2001.**

TSPLIB

TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types.

TSPLIB : http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95

MIPLIB - Mixed Integer Problem Library

ZIB

In response to the needs of researchers for access to real-world mixed integer programs a group of researchers Robert E. Bixby, E.A. Boyd, and R.R. Indovina created in 1992 the MIPLIB, an electronically available library of both pure and mixed integer programs. This was updated in 1996 by Robert E. Bixby, Sebastian Ceria, Cassandra M. McZeal, and Martin W.P. Savelsbergh. The library was updated again in 2003 by Alexander Martin, Tobias Achterberg, and Thorsten Koch.

Since its introduction, MIPLIB has become a standard test set used to compare the performance of mixed integer optimizers. Its availability has provided an important stimulus for researchers in this very active area.

MIPLIB 2010

MIPLIB: http://miplib.zib.de

Meta-heuristic

浙江大学
ZheJiang University
数学建模

- **Meta-heuristic算法的特征**
  - **Metaheuristics are strategies that "guide" the search process. The goal is to efficiently explore the search space in order to find (near-) optimal solutions**

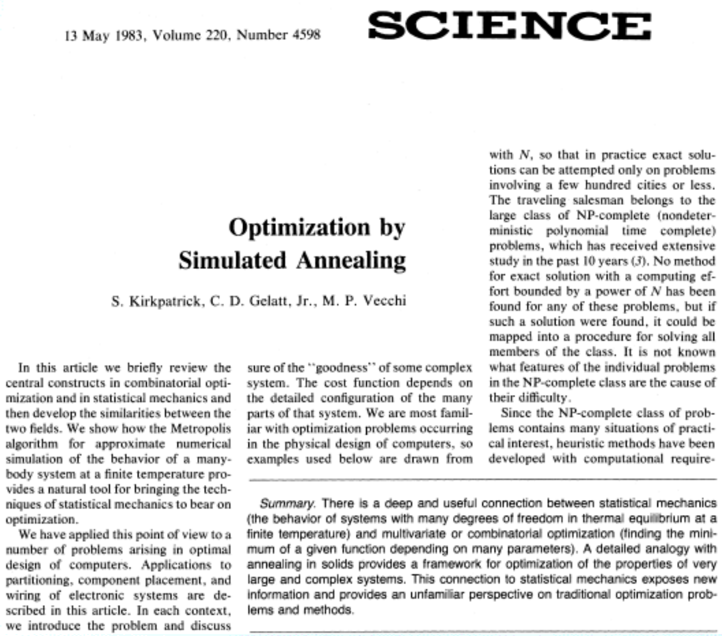
Meta	**Heuristic**
beyond, in an upper level	(ευρισκειν) to find

  - **Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes**
  - **Metaheuristic algorithms are approximate and usually non-deterministic**
  - **They may incorporate mechanisms to avoid getting trapped in confined areas of the search space**
  - **The basic concepts of metaheuristics permit an abstract level description. Metaheuristics are not problem-specific**

浙江大学
ZheJiang University
数学建模

Meta-heuristic

- 常用**Meta-heuristic算法**
  - 遗传算法（**genetic algorithm**）
  - 模拟退火算法（**simulated annealing**）
  - 禁忌搜索（**tabu search**）
  - 变邻域搜索（**variable neighborhood search**）
  - 蚁群算法（**ant colony optimization**）
  - 粒子群优化算法（**particle swarm optimization**）



13 May 1983, Volume 220, Number 4598 **SCIENCE**

Optimization by Simulated Annealing

S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi

In this article we briefly review the central constructs in combinatorial optimization and in statistical mechanics and then develop the similarities between the two fields. We show how the Metropolis algorithm for approximate numerical simulation of the behavior of a many-body system at a finite temperature provides a natural tool for bringing the techniques of statistical mechanics to bear on optimization.

We have applied this point of view to a number of problems arising in optimal design of computers. Applications to partitioning, component placement, and wiring of electronic systems are described in this article. In each context, we introduce the problem and discuss sure of the "goodness" of some complex system. The cost function depends on the detailed configuration of the many parts of that system. We are most familiar with optimization problems occurring in the physical design of computers, so examples used below are drawn from

with *N*, so that in practice exact solutions can be attempted only on problems involving a few hundred cities or less. The traveling salesman belongs to the large class of NP-complete (nondeterministic polynomial time complete) problems, which has received extensive study in the past 10 years (*3*). No method for exact solution with a computing effort bounded by a power of *N* has been found for any of these problems, but if such a solution were found, it could be mapped into a procedure for solving all members of the class. It is not known what features of the individual problems in the NP-complete class are the cause of their difficulty.

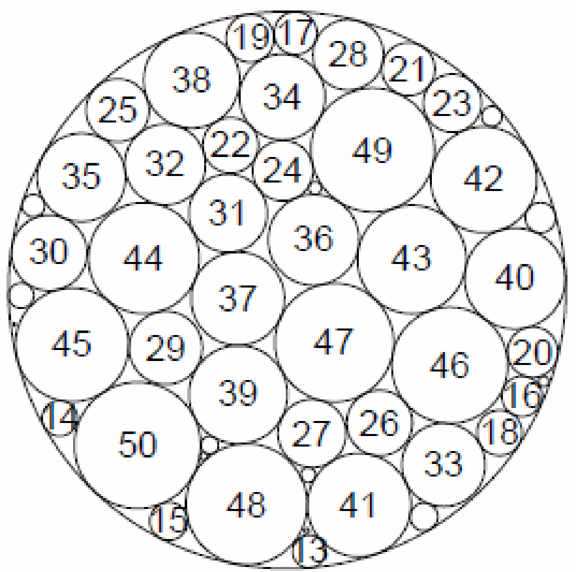
Since the NP-complete class of problems contains many situations of practical interest, heuristic methods have been developed with computational require-

Summary. There is a deep and useful connection between statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters). A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. This connection to statistical mechanics exposes new information and provides an unfamiliar perspective on traditional optimization problems and methods.

Blum, C., Roli, A., Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys*, 35, 268-308, 2003.

Circle Packing

浙江大学
ZheJiang University
数学建模



N=50, R=220.6004187

Packing, Improved
（2009, Rank 37）
TIME

PHYSICAL REVIEW E **79**, 021102 (2009)

Packing a multidisperse system of hard disks in a circular environment

André Müller, Johannes J. Schneider, and Elmar Schömer*
Department of Physics, Mathematics, and Computer Science, Johannes Gutenberg University of Mainz, Staudinger Weg 7, 55099 Mainz, Germany
(Received 11 July 2008; revised manuscript received 27 December 2008; published 2 February 2009)

We consider the problem of finding the densest closed packing of hard disks with proposed different radii in a circular environment, such that the radius of the circumcircle is minimal. With our approach, we are able to find denser packings for various problem instances than known from the literature. Both for the dynamics of the simulation and for the optimum values of the radii of the circumcircles, we find various scaling laws.

DOI: 10.1103/PhysRevE.79.021102 PACS number(s): 64.60.De, 64.75.Gh, 65.40.Ba, 89.75.Da

Circle Packing

浙江大学
ZheJiang University
数学建模

N	Contest record	Our value	Record
5	9.0013977	9.0013977	matched
6	11.0570404	11.0570404	matched
7	13.4621107	13.4621107	matched
8	16.2217467	16.2217467	matched
9	19.2331939	19.2331939	matched
10	22.0001930	22.0001930	matched
23	71.1994616	71.1994616	matched
24	75.7527041	75.7491426	beaten
25	80.2858644	80.2858644	matched
26	85.0764012	84.9899391	beaten
27	89.7921816	89.7509627	beaten
28	94.5499865	94.5265365	beaten
47	202.1856117	201.7279256	beaten
48	208.6359467	208.0901593	beaten
49	214.6619520	214.2954475	beaten
50	221.0897526	220.6004187	beaten

Al Zimmermann's Programming Contests

http://www.azspcs.net/

Circle Packing Total prizes: $500
Contest start: Sunday 10/30/2005
Contest end: Saturday 01/14/2006

Pack N non-overlapping discs with radii from 1 to N into as small a circle as possible

The best known packings of unequal circles with integer radii in a square（www.packomania.com）


浙江大学
ZheJiang University
运筹与统计
排序与装箱

排序问题

数学建模

- 排序（**scheduling**）主要研究如何利用有限资源，在给定的限制条件下，将一批任务安排在某些时间段内完成，并使效益最大
  - 早期研究的排序问题背景源自工业生产，习惯上把可用的资源称为机器（**machine**），需要完成的任务称为工件（**job**）
  - 对部分排序问题，可行解由工件加工的顺序决定，这类问题最早也被称作**sequencing**

Journal of Scheduling

排序问题

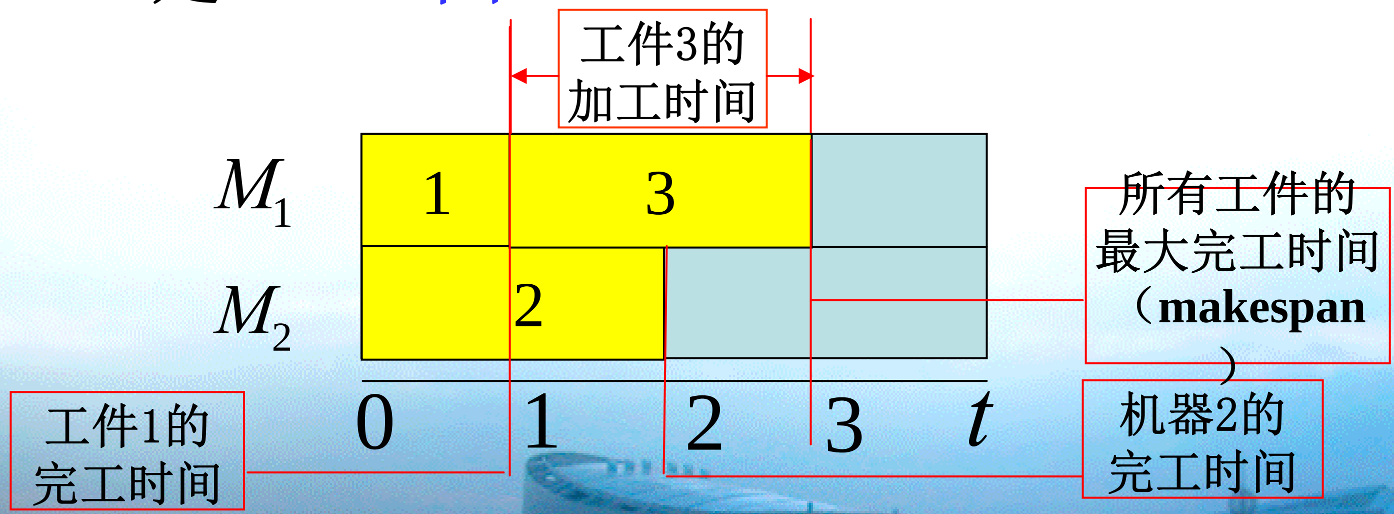
浙江大学
ZheJiang University
数学建模

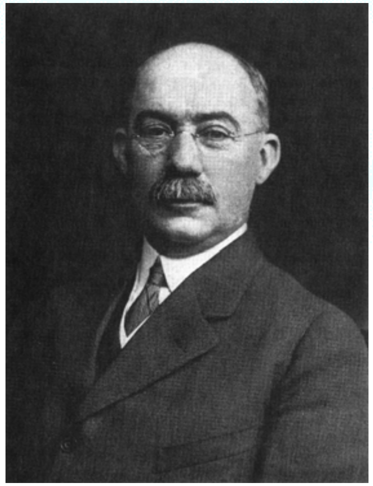
- **Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics*, 5, 287–326, 1979**
    - 三参数表示法（**three-field notation)** $\alpha\,|\,\beta\,|\,\gamma$
- **Lawler, E.L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D.B., Sequencing and Scheduling: Algorithmus and Complexity, Volume 4 of *Handbook in Operations Research and Managment Science*, North-Holland, 1993**

Sequencing and scheduling is concerned with the *optimal allocation of scarce resources to activities over time*. Of obvious practical importance, it has been the subject of extensive research since the early 1950's, and an impressive amount of literature has been created. Any discussion of the available material has to be selective. We will concentrate on the area of deterministic machine

排序问题

- 排序是组合优化中模型最丰富、应用最广泛的问题之一
- 描述排序问题及其可行解的常用工具是**Gantt 图**





Henry Gantt
(1861-1919)
美国管理学家

单台机问题

浙江大学 ZheJiang University

数学建模

- 现有 n 个工件，工件 j 的加工时间为 p_j，预定交工期（**duedate**）为 $d_j\,, j=1,\cdots,n$。机器在同一时刻只能加工一个工件，工件加工不可中断。如何确定工件的加工顺序，可使得
  - 误工（完工时间大于预定交工期）的工件数最少 $\mathcal{P}$ 问题
  - 各工件延误时间（完工时间与预定交工期之差）的最大值最小 **EDD**规则：将工件按预定交工期的非减序排列
  - 所有误工工件的总延误时间最小 $\mathcal{NP}$ 一完全问题

Santa Claus问题

数学建模

- 圣诞老人欲将 n 件礼物分给 m 位小朋友。一件礼物只能分给一位小朋友，但一位小朋友可以拿到多件礼物。第 i 位小朋友拿到礼物 j 时他的满意度为 p_{ij}，$i=1,\cdots,m$，$j=1,\cdots,n$，一位小朋友拿到多件礼物时的满意度为各礼物相应满意度之和。希望给出一礼物分配方案，使满意度最小的小朋友的满意度尽可能大

礼　物——工　件　　满意度 ——加工时间

小朋友——机　器　　小朋友的满意度—机器完工时间

目标函数——极大化最小机器完工时间　$Rm\,||\,C_{\min}$

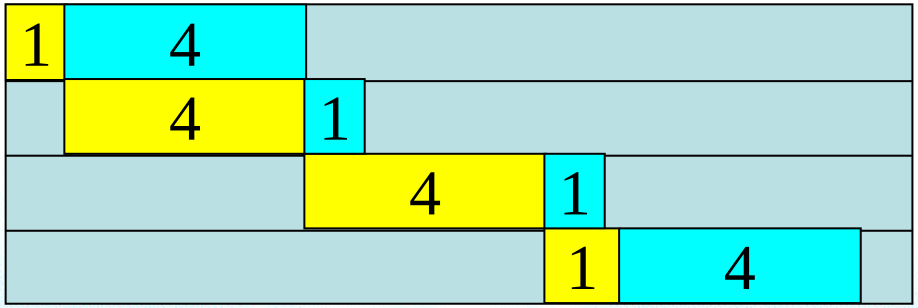
STOC

Bansal, N., Sviridenko, M., The Santa Claus problem，*Proceedings of the 38th Annual ACM Symposium on Theory of computing*, 31-40, 2006

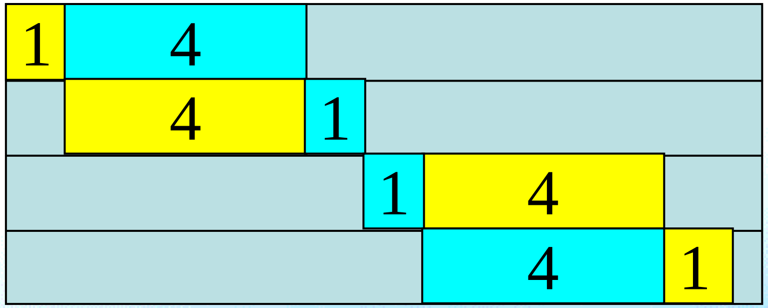
车间作业

数学建模

- 每个工件加工需经过若干道工序，每道工序需在给定机器上经过一定的加工时间才能完成
  - 流水作业（**Flowshop**）：所有工件有相同的工序，前一道工序完工后才能开始下一道工序



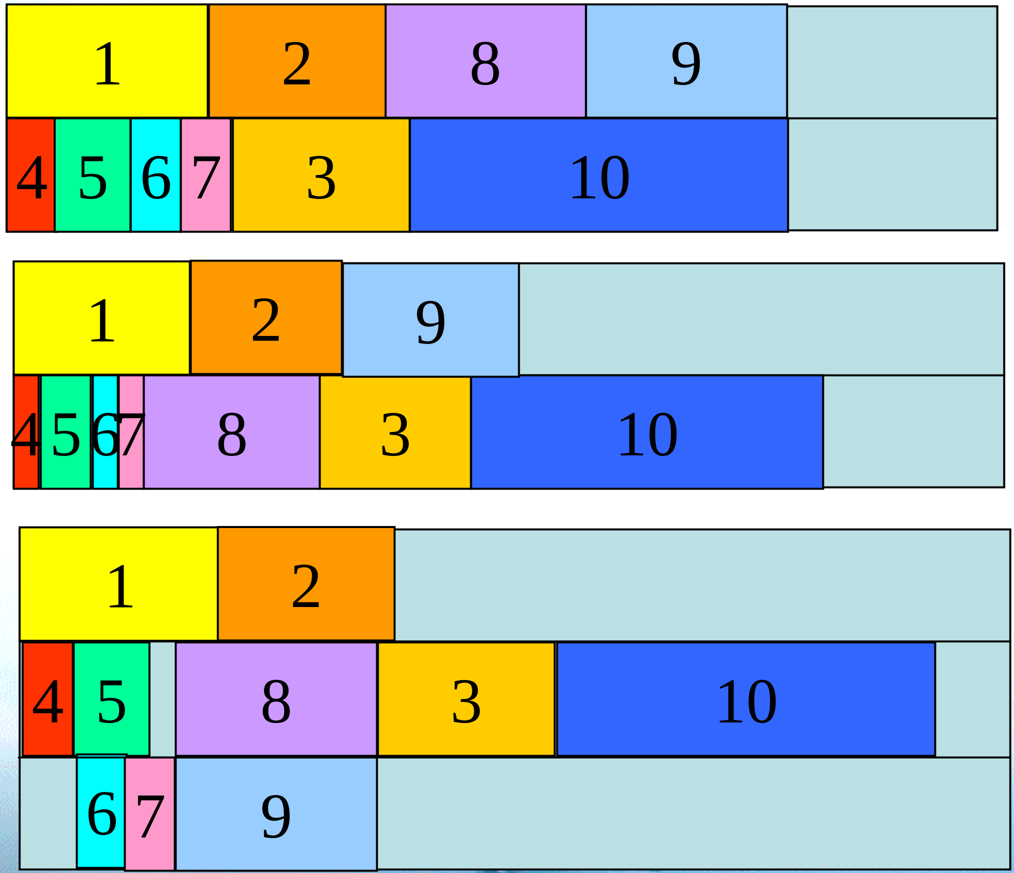
makespan为14



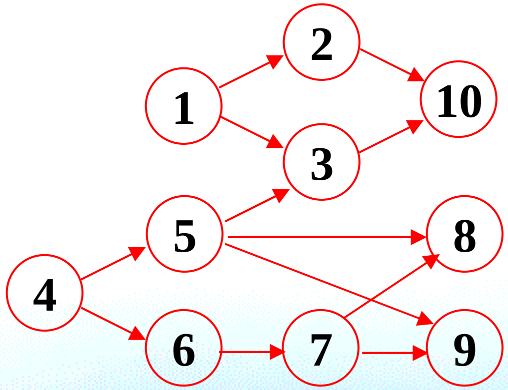
最优makespan为12

各台机器上工件加工的顺序未必相同

排序悖论



工件	1	2	3	4	5	6	7	8	9	10
加工时间	8	7	7	2	3	2	2	8	8	15
	7	6	6	1	2	1	1	7	7	14

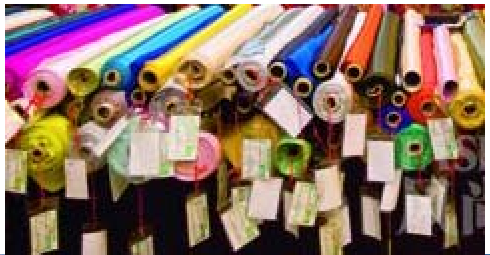


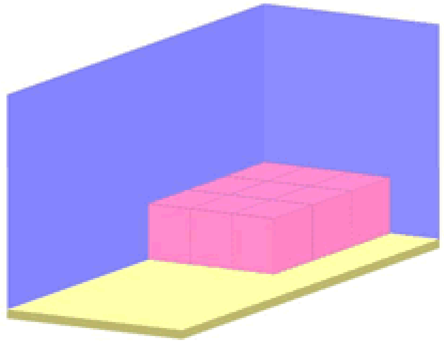
目标：尽早完工

贪婪思想：可以加工的工件尽早开工

装箱问题

- 装箱问题（**bin-packing**）研究如何将一系列物品放入容量一定的若干箱子中，使得在放入每个箱子中的物品大小之和不超过箱子容量的前提下，所用箱子数尽可能少
  - 一维装箱：钢管下料、文件存储
  - 二维装箱：板材切割、布匹裁剪
  - 三维装箱：集装箱拼箱、货柜装车





一维装箱

浙江大学 ZheJiang University

数学建模

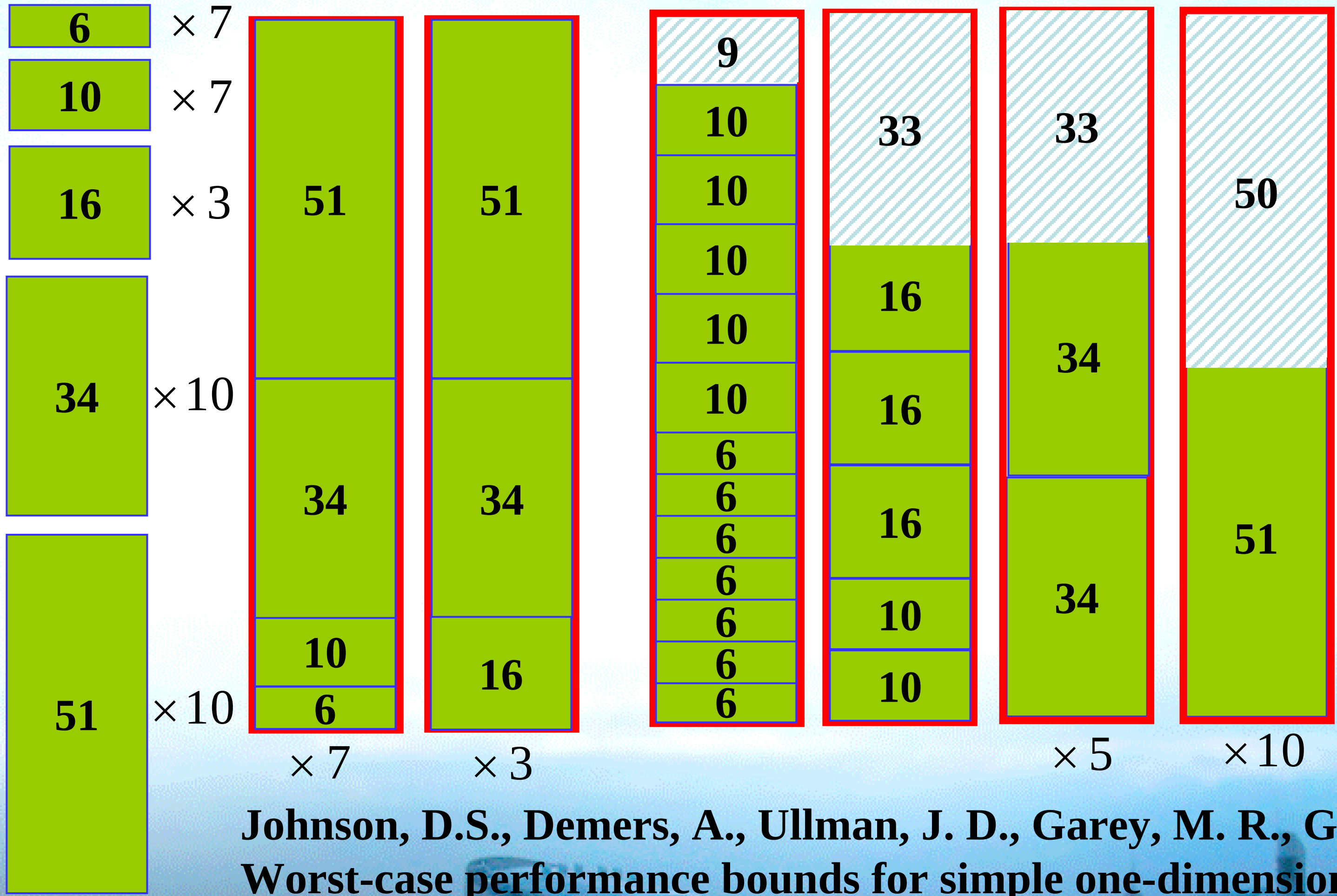
- 箱子容量为 C，物品 j 的大小为 $w_j, 0 \le w_j \le C$,
  - **First Fit**（**FF**）算法：将物品放在按箱子启用顺序第一个能放下的箱子中
    - **FF**算法的最坏情况界 $\underset{(1994)}{\dfrac{7}{4}} \rightarrow \underset{(2010)}{\dfrac{12}{7}} \rightarrow \underset{(2013)}{\dfrac{17}{10}}$

Simchi-Levi, D., New worst case results for the bin-packing problem. *Naval Research Logistics*, 41:579–585, 1994

Xia, B. Z., Tan, Z. Y., Tighter bounds of the First Fit algorithm for the bin-packing problem. *Discrete Applied Mathematics*, 158:1668–1675, 2010

Dósa, G., Sgall, J., First Fit bin packing: A tight analysis. *Proceeding of 30th Symposium on Theoretical Aspects of Computer Science*, 538-549, 2013

First Fit算法



$$C = 101$$

$$C^* = 10$$

$$C^{FF} = 17$$

$$\frac{C^{FF}}{C^*} = \frac{17}{10}$$

Johnson, D.S., Demers, A., Ullman, J. D., Garey, M. R., Graham, R. L., Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal on Computing*, 3, 299-325, 1974

一维装箱

浙江大学 ZheJiang University

数学建模

- **First Fit Decreasing**（**FFD**）算法：将物品按大小从大到小的顺序重新排列，再用**First Fit**算法装箱
  - **FFD**算法的最坏情况界为 $\frac{3}{2}$，即 $C^{FFD}(I) \le \frac{3}{2} C^{*}(I)$
- 任何多项式时间近似算法的最坏情况界至少为$\frac{3}{2}$，除非$\mathcal{P}=\mathcal{NP}$

$$C^{FFD}(I) \le \frac{11}{9} C^{*}(I) + \frac{6}{9} \qquad C^{A}(I) \le (1+\varepsilon) C^{*}(I) + 1$$

常数项

- 在 $\mathcal{P} \ne \mathcal{NP}$假设下，是否存在多项式时间算法 A，使得 $C^{A}(I) - C^{*}(I) \le \text{const}$ 仍是未知的

太阳能屋顶

浙江大学 ZheJiang University

数学建模

- 用不同类型的光伏电池铺设给定长宽的矩形屋顶
- 设类型为 i 的光伏电池是长为 a_i，宽为 b_i 的矩形，每块该类型电池预期效益为 p_i
- 铺设时光伏电池不能互相覆盖，也不能超出屋顶之外，但不必覆盖屋顶所有区域
- 采用怎样的铺设方案可使效益最大

二维装箱？ 背包？

多背包？ 多维背包？

面积和？ 长宽和？


浙江大学
ZheJiang University
谢 谢