近似、随机与局部搜索

近似算法

- 许多NP难的问题不能用回溯法有效地解决
- 求一个与精确解相差不算太多的近似解
- 效率比较高

近似度

- 绝对近似度：近似解与最优解不超过某个常数。
- 相对近似度：近似解与最优解不超过某个比例。
- 极少NP难的问题具有绝对近似度

装箱问题

- 最先适配法（FF）：把物品放入第一个能装下的箱子里
- 最优适配法（BF）：把物品放入最空的一个箱子里
- 递减最先适配（FFD）：先将物品从大到小排序，然后用FF
- 递减最佳适配（BFD）：先将物品从大到小排序，然后用BF

FF的近似度

- 可以把问题转化为每个物品体积在[0, 1]区间内，箱子容积为1。
- 半空的箱子（装的物品体积和<=0.5）不会超过两个
- $FF(I) <= 2\sum V_i$
- $OPT(I) >= \sum V_i$
- $FF(I)/OPT(I) <= 2$

四种算法的近似度

- FF：17/10
- BF：17/10
- FFD：11/9
- BFD：11/9

欧几里得哈密尔顿回路

- 对于平面上的点集求最短哈密尔顿回路。
- 贪心法：每次寻找一个最近的点。
- 近似度无界

近似度为2的算法

- 构造一个最小生成树
- 复制每条边
- 构造欧拉回路
- 用跳过已经访问过的节点的办法转换成哈密尔顿回路

近似度证明

- T=最小生成树边长和
- T<OPT(I)
- 2T<2OPT(I)
- 最后得到的回路长<2T<2OPT(I)

最小顶点覆盖

- 贪心法：每次取一个顶点，然后删除与它相邻的边
- 近似度无界

近似度为2的算法

- 每次取一条边，把它的两个端点都同时加入结果中，然后删除所有相连的边。
- 证明：这样将得到一个极大匹配M，所取顶点数=2|M|。同时OPT(I)>=|M|。

多机任务调度

- 给定一个任务集合以及它们的执行所需时间。要在m台机器上执行这些任务，使得完成时间最短。
- 贪心法：每次选择第一台可用的机器
- 近似度为2-1/m
- 将任务按照执行时间从大到小排序
- 近似度为4/3-1/3m

平面图的k中心

- 给定一个平面点集，选择其中k个点使得其他点到这k个点的最小距离尽可能地小。
- 贪心法，每次选择一个离已经选中的点最远的一个点
- 近似度为2

背包问题

- 思路：把所有物品的体积都增加为某个整数k的倍数，从而可以把体积统一转换为较小的整数。用动态规划求解。

随机算法

- Las Vegas：总是给出正确的解，或者找不到解。
- Monte Carlo：总是有解，但不一定正确。

随机快速排序

- 在每次递归时随机选择一个元素和第一个元素进行交换，然后执行通用的快速排序算法。

素数测试

- 费马小定理：如果n是素数，那么对于所有不是n的倍数的整数a，有$a^{n-1} \equiv 1 (\mathrm{mod}\ n)$
- 思路：随机选择多个整数进行检验。

求 $a^m \pmod n$

- result := 0;
- b := 1;
- while m > 0 do
- begin
- b := (b * a) mod n;
- if Odd(m) then result := (result + b) mod n;
- m := m div 2;
- end;
- return result;

素数测试算法

- 随机选择多个整数进行测试，对于通过测试的数再用完整的素数验证算法进行验算。
- 事实上存在出错概率更小的素数测试算法。

字符串匹配

- 为简化问题，只考虑01串。
- 基本思想：可以把字符串看成一个整数。一个01串其实就是一个整数的2进制 表示。随机选择一个质数p。那么两个01串A、B相等的必要条件是$A \equiv B \pmod{p}$。

算法

- 读入01串A、B
- 假设A长度>B
- 随机选择素数p
- 计算X=B mod p、Y=substr(A, 0, len(B)) mod p以及 $Z=2^{len(B)} \bmod p$
- for 每个起始位置i do
- 　if X=Y then return 当前位置
- 　$Y = Y * 2 - A_i * Z + A_{i+len(B)}$
- return 找不到任何匹配

改进

- 通过选择多个素数同时求模可以减少出错的概率
- 每次出现模相等时都进行一次完整的比较来验算。
- 通过选择适当的素数可以把出错概率控制在1/n之内
- 算法复杂度=O(n + m)

表达式

- 给定两个只包含+-*运算符的表达式，问它们是否等价。
- 例如(X+Y)*(X-Y)和X*X-Y*Y是完全等价的。
- 一种思路：可以给变量随机赋值，检查两个表达式得到的结果是否相等。

随机化贪心

- 在使用贪心算法时，往往可以加入随机化因子，从而在一定程度上避免过分贪心的缺陷。
- for A←局部最优解to 局部最差解
- 　以概率p返回A
- return 局部最差解

并行计算

- 输入一个由+，-，，，(，)，变量(大写字母)组成的四则运算表达式，输出一段指令，控制有两个运算器的并行计算机在尽量短的时间内正确计算表达式的值。

贪心方案

- 每次选取耗时长的运算符
- 每次选取最早空闲的运算器
- 加入随机因子，并非完全贪心

局部搜索

- 基本思想：通过局部调整来达到某个极大值

SAT

- 目标：尽可能减少一个CNF公式中值为假的子句数目
- 随机产生一组变量赋值
- 每次改变一个变量的赋值，如果值为假的子句数目减少，就接受，否则重新选取变量

n皇后问题

- 目标：使能够相互攻击的皇后对数目尽可能地小
- 随机生成一个布局
- 每次改变一个皇后的位置。如果能够相互攻击的皇后对数目减小，则接受，否则重新选取皇后和位置。

贪婪局部搜索

- 局部搜索往往配合贪心法，即选择当前最好的调整方法。
- SAT：选择使为假子句减少最快的方案
- n皇后：选择与其他皇后冲突最小的方案。

局部搜索的缺点

- 容易遇到局部最大值
- 侧向移动：在无法找到一个更好的解的时候，选择一个相等的解。需要限制侧向移动的步数，因为很容易陷入死循环。
- 随机行走：在无法找到一个更好的解的时候，对局部做随机调整。

禁忌搜索

- 禁忌搜索是对局部搜索的一种改进，它的主要思想是禁止连续进行某些重复的局部调整操作。

禁忌搜索算法描述

- 选定一个初始解。禁忌表为空。
- 若满足停止规则，停止计算；否则在满足禁忌要求的候选集中选出一个评价值最佳的解。
- 更新禁忌表，并进行局部调整。

01串

- 给定7个整数N,A0,B0,L0,A1,B1,L1，要求设计一个01串S=s1s2…si…sN，满足：
- si=0或si=1，1<=i<=N；
- 对于S的任何连续的长度为L0的子串sjsj+1…sj+L0-1(1<=j<=N-L0+1)，0的个数大于等于A0且小于等于B0;
- 对于S的任何连续的长度为L1的子串sjsj+1…sj+L1-1(1<=j<=N-L1+1)，1的个数大于等于A1且小于等于B1;
- 例如，N=6,A0=1,B0=2,L0=3,A1=1,B1=1,L1=2，则存在一个满足上述所有条件的01串S=010101。

定义

- $g_0(x,i)$为解x中从第i个字符开始连续L0个字符中“0”的个数。
- $g_1(x,i)$为解x中从第i个字符开始连续L1个字符中“1”的个数。

$$f(x)=\sum_{i=1}^{N-L_0+1}\begin{cases} g_0(x,i)-B_0 \mid g_0(x,i)>B_0 \\ 0 \mid A_0 \le g_0(x,i) \le B_0 \\ A_0-g_0(x,i) \mid g_0(x,i)<A_0 \end{cases} + \sum_{i=1}^{N-L_1+1}\begin{cases} g_1(x,i)-B_1 \mid g_1(x,i)>B_1 \\ 0 \mid A_1 \le g_1(x,i) \le B_1 \\ A_1-g_1(x,i) \mid g_1(x,i)<A_1 \end{cases}$$

- 求一个解x’，使得f(x')=0。在求解的过程中，目标是寻找f(x)尽可能小的解“x”。
- N=10,A0=1,B0=2,L0=3,A1=1,B1=1,L1=3。
- 假设：初始解X_0=1111111111。

第一步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	22	20	18★	18	18	18	18	18	20	22	
禁忌长度	0	0	0	0	0	0	0	0	0	0	

第二步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	17	16	24T	14	13	12★	12	12	14	16	
禁忌长度	0	0	3	0	0	0	0	0	0	0	

第三步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	11	10	18T	9	9	18T	8	7★	8	10	
禁忌长度	0	0	2	0	0	3	0	0	0	0	

第四步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	6	5	13T	4★	4	12T	7	12T	5	6	
禁忌长度	0	0	1	0	0	2	0	3	0	0	

第五步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	3	5	8	7T	7	8T	4	9T	2★	3	
禁忌长度	0	0	0	3	0	1	0	2	0	0	

第六步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	1★	3	6	5T	5	6	5	5T	4T	4	
禁忌长度	0	0	0	2	0	0	0	1	3	0	

第七步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	2T	5	4	4T	4	5	4	4	3T	3★	
禁忌长度	3	0	0	1	0	0	0	0	2	0	

第八步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	4T	7	6	6	6	7	6	3★	2T	1T	
禁忌长度	2	0	0	0	0	0	0	0	1	3	

第九步

操作	1	2	3	4	5	6	7	8	9	10	
评价值	4T	7	6	6	6	8	0★	3T	6	4T	
禁忌长度	1	0	0	0	0	0	0	3	0	2	

特赦规则

- 某些时候，被禁止的操作能够让目标函数值大幅变优，此时可以取消该禁忌。

模拟退火算法

- 模拟退火算法是把局部搜索和随机化思想结合起来应用。
- 在每次选择局部调整的时候，随机选择一个调整方案。如果调整方案得到的解优于当前解，则接受。否则以某个小于1的概率接受。这个概率和解的“恶劣程度”成指数关系，越“恶劣”则越小。并且随着迭代的深入，这个概率也会越来越小。

模拟退火算法描述

- for t := 1 to MAX do
- begin
- T := f(t);
- if T = 0 then 返回当前解;
- 随机选择一种局部调整方案
- ΔE=调整方案值-当前解值
- if ΔE > 0 then 进行局部调整
- else 以$e^{\Delta E/T}$的概率进行局部调整
- end;