Review

课程主要内容

问题处理策略

计算复杂性理论

算法分析方法

算法设计技术

基础知识

函数的阶

- **阶的符号**：$O, \Omega, \Theta, o, \omega$
- **阶的高低**

  至少指数级：$2^n, 3^n, n!, \ldots$

  多项式级：$n, n^2, n\log n, n^{1/2}, \ldots$

  $\log n$的多项式级：$\log n, \log^2 n, \ldots$

- **注意**

  阶反映的是大的n（$n>n_0$）的情况 可以忽略前面的有限项

递推方程求解

- 主要的求解方法

  迭代+ 进行序列求和

  递归树+ 求和

  主定理：注意条件验证

- 一些常见的递推方程的解

$$f(n) = af(\frac{n}{b}) + d(n)$$

分治策略

- 适用条件：归约为独立求解子问题
- 设计步骤：归约方法，初始子问题的计算，子问题解的综合方法.　注意子问题划分均衡，类型相同
- 递归算法分析：求解递推方程
- 改进途径：减少子问题数，预处理

改进分治算法的途径2:增加预处理

例子：平面点对问题

输入: 平面点集 P 中有n 个点, $n>1$
输出: P中的两个点，其距离最小

蛮力算法:
$C(n,2)$个点对,计算最小距离，$O(n^2)$

分治策略: P 划为大小相等的P_L和P_R

1. 分别计算 P_L、P_R中最近点对
2. 计算 P_L与P_R中各一个点的最近点对
3. 上述情况下的最近点对是解

动态规划

- 适用条件：优化问题, 多步判断求 解, 满足优化原则,子问题重叠
- 设计步骤：确定子问题边界, 列关于 目标函数的递推方程及初值；自底 向上,备忘录存储；标记函数及解的 追踪方法
- 复杂度分析：备忘录, 递推方程

背包问题 (Knapsack Problem)

一个旅行者随身携带一个背包. 可以放入背包的物品有n 种, 每种物品的重量和价值分别为 w_i, v_i. 如果背包的最大重量限制是 b, 每种物品可以放多个. 怎样选择放入背包的物品以使得背包的价值最大？ 不妨设上述 w_i, v_i, b 都是正整数.

实例： $n = 4$， $b = 10$

$v_1 = 1$， $v_2 = 3$， $v_3 = 5$， $v_4 = 9$，

$w_1 = 2$， $w_2 = 3$， $w_3 = 4$， $w_4 = 7$，

贪心法

- 适用条件：组合优化问题, 多步判 断求解, 有贪心选择性质
- 设计步骤：局部优化策略的确定及 算法正确性证明(直接证明， 数学归 纳法, 交换论证)
- 复杂度分析

回溯和分支限界

- 适用条件：搜索或优化问题, 多步 判断求解, 满足多米诺性质
- 设计步骤：确定解向量, 搜索树结构, 搜索顺序,结点分支搜索的约束条件 与代价函数, 路径存储
- 搜索树结点数估计
- 复杂度分析

着色问题

输入：

无向连通图 G和 m 种颜色的集合

用这些颜色给图的顶点着色，每个顶点一种颜色. 要求是：G 的每条边的两个顶点着不同颜色.

输出：所有可能的着色方案.
如果不存在着色方案，回答 “**No**”.

分支限界

停止分支回溯父结点的依据：

1. 不满足约束条件
2. 对于极大化问题，代价函数值小于当前界（对于极小化问题是大于界）

界的更新

对极大化问题，如果一个新的可行解的优化函数值大于 (极小化问题为小于) 当前的界，则把界更新为该可行解的值

实例

背包问题：

4 种物品，重量 w_i 与价值 v_i 分别为

$v_1=1,\ v_2=3,\ v_3=5,\ v_4=9$

$w_1=2,\ w_2=3,\ w_3=4,\ w_4=7$

背包重量限制为10

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$

$$2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10$$

$$x_i \in \mathrm{N},\ i = 1,2,3,4$$

线性规划

- 几个线性规划的例子
- 二维线性规划的图解法
- 线性规划的标准形
- 标准形的可行解的性质
- 单纯形法
- 单纯形表的例子

- **对偶规划的例子**
- **原始规划和对偶规划**
- **对偶单纯形法**
- **整数线性规划的分支限界方法**
- **应用实例**

标准形

$$\min z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \geq 0, \quad i = 1,2,\ldots,m$$

$$x_j \geq 0, \quad j = 1,2,\ldots,n$$

化成标准形

(1) 把 $\max z$ 替换成 $\min z' = -z$, 即取 $c_j' = -c_j$.

(2) $b_i < 0$. 两边同时变号, $\leq$ 改变成 $\geq$, $\geq$ 改变成 $\leq$.

(3) $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$. 引入松弛变量 $y_i \geq 0$ ，替换成

$$\sum_{j=1}^{n} a_{ij} x_j + y_i = b_i$$

(4) $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$. 引入剩余变量 $y_i \geq 0$ ，替换成

$$\sum_{j=1}^{n} a_{ij} x_j - y_i = b_i$$

(5) 自由变量 x_j 替换成 $x_j' - x_j''$ ，$x_j' \geq 0$, $x_j'' \geq 0$.

标准形的可行解的性质

定义 设A的秩为m, A的m个线性无关的列向量称作标准形的基. 给定基$B=(P_{i_1},P_{i_2},\cdots,P_{i_m})$, 对应基中列向量的变量 $x_{i_1},x_{i_2},\cdots,x_{i_m}$ 称作基变量, 其余的变量称作非基变量.

基变量构成的向量记作x_B, 非基变量构成的向量记作x_N. 令$x_N=0$, 等式约束变成

$$Bx_B=b$$

解得$x_B=B^{-1}b$. 这个向量x满足约束$Ax=b$且非基变量全为0, 称作关于基B的基本解. 如果x是一个基本解且$x\geq 0$, 则称x是一个基本可行解, 对应的基B为可行基.

定理1 如果标准形有可行解, 则必有基本可行解.

定理2 如果标准形有最优解, 则必存在一个基本可行解是最优解.

单纯形法

基本步骤

(1) 确定初始基本可行解.

(2) 检查当前的基本可行解.

若是最优解或无最优解, 计算结束;

否则作基变换, 用一个非基变量替换一个基变量, 得到一个新的可行基和对应的基本可行解, 且使目标函数值下降(至少不升).

(3) 重复(2).

单纯形法

算法 单纯形法 (针对最小化)

1. 设初始可行基 $B = (P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(m)})$, $\alpha = B^{-1}A$,
   $\beta = B^{-1}b$, $\lambda^T = c^T - c_B^T B^{-1}A$, $z_0 = c_B^T B^{-1} b$.
2. 若所有 $\lambda_j \geq 0$ $(1 \leq j \leq n)$, 则 $x_B = \beta$, $x_N = 0$ 是最优解, 计算结束
3. 取 $\lambda_k < 0$. 若所有 $\alpha_{ik} \leq 0 (1 \leq i \leq m)$, 则无最优解, 计算结束.
4. 取 l 使得

$$\beta_l / \alpha_{lk} = \min\{ \beta_i / \alpha_{ik} \mid \alpha_{ik} > 0, \ 1 \leq i \leq m \}$$

5. 以 x_k 为换入变量、$x_{\pi(l)}$ 为换出变量做基变换.
6. 转2.

对最大化, 2中 $\lambda_j \geq 0$ 改为 $\lambda_j \leq 0$, 3中 $\lambda_k < 0$ 改为 $\lambda_k > 0$.

单纯形表

			c_1	c_2	$\dots$	c_n	
c_B	x_B	b	x_1	x_2	$\dots$	x_n	θ
$c_{\pi(1)}$	$x_{\pi(1)}$	β_1	α_{11}	α_{12}	$\dots$	α_{1n}	
$c_{\pi(2)}$	$x_{\pi(2)}$	β_2	α_{21}	α_{22}	$\dots$	α_{2n}	$\theta_i = \beta_i / \alpha_{ik}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	
$c_{\pi(m)}$	$x_{\pi(m)}$	β_m	α_{m1}	α_{m2}	$\dots$	α_{mn}	
	$-z$	$-z_0$	λ_1	λ_2	$\dots$	λ_n	

$$-z+\lambda_1 x_1+\lambda_2 x_2+\dots+\lambda_n x_n = -z_0$$

对偶单纯形法

定义 设 B是一个基, 如果 $\lambda \geq 0$, 则称 B 是**正则的**.

如果 B 是正则的, 那么 y 是 (D)的可行解, 从而只要 x 是(P)的可行解, 亦即 $x_B = B^{-1}b \geq 0$, 则 x 和 y 分别是 (P) 和 (D) 的最优解.

单纯形法 保持 x 是 (P) 的可行解 (保持B是可行基), 即保持 $B^{-1}b \geq 0$, 通过基变换使 y 逐步成为 (D) 的可行解 (B变成正则基), 即逐步使 $\lambda \geq 0$.

对偶单纯形法 保持 y 是 (D) 的可行解 (保持B是正则基), 即保持 $\lambda \geq 0$, 通过基变换使 x 逐步成为 (P) 的可行解 (B变成可行基), 即逐步使 $B^{-1}b \geq 0$.

对偶单纯形法

算法2 对偶单纯形法

1. 取正则基 B.

2. 如果 $\beta \geq 0$, 则 x 是最优解, 计算结束.

3. 取 $\beta_l < 0$. 若所有$\alpha_{lj} \geq 0$ $(1 \leq j \leq n)$, 则无可行解, 计算结束.

4. 取 k 使得

$$|\lambda_k / \alpha_{lk}| = \min\{ |\lambda_j / \alpha_{lj}| : \alpha_{lj} < 0\}$$

5. 以 $x_{\pi(l)}$ 为换出变量、 x_k 为换入变量做基变换.

6. 转2.

网络流算法

- 网络流及其性质
- 最大流最小割定理
- Fold-Fulkerson最大流算法（增广链）
- 辅助网络（最短增广链，一次多条增广链）
- Dinic最大流算法
- Dinic算法分析
- 最小费用流及Floyd算法
- 最小费用流的负回路算法
- 最小费用流的最短路径算法
- 运输问题（最小费用流思路）
- 二部图的最大匹配（增广交错路径）
- 赋权二部图的匹配（匈牙利算法）

最小费用流的负回路算法

算法 最小费用流的负回路算法

1. 调用最大流算法, 若求得流量 v_0 的可行流 f, 则转2; 若最大流量小于v_0, 则输出"无流量v_0的可行流", 计算结束.
2. 构造 $N(f)$.
3. 用 Floyd算法检测 $N(f)$ 中是否存在权aw的负回路. 若存在负回路C, 则转4; 若不存在负回路, 则输出 f, 计算结束.
4. 令$\delta \leftarrow \min\{ ac(i, j) \mid \langle i, j\rangle \in E(C)\}$ // h^C的环流量为δ
5. $\forall \langle i, j\rangle \in E(C)$, 若$\langle i, j\rangle \in E$, 令 $f(i,j) \leftarrow f(i,j)+\delta$; 若$\langle j, i\rangle \in E$, 令 $f(j, i) \leftarrow f(j, i)-\delta$. // $f \leftarrow f + h^C$
6. 转2

最小费用流的最短路径算法

算法基本想法: 从一个初始最小费用流 f (如零流)开始, 如果$v(f)<v_0$, 找一条费用最少的 s-t 增广链 P, 需要用Floyd算法计算P . 修改 P上的流量, 得到新的可行流 f'. 重复进行, 直至流量等于 v_0 为止. 关键是要证明这样得到的 f'也是最小费用流.

最小费用流思路求解运输问题

1. 首先类似求最大流，
   确定一个对应的初始运输方案
2. 其次以初始方案为基础，
   类似消减负回路的方式，
   不断优化运输方案，
3. 直到不能优化为止，获得最优运输方案

算法分析与问题的计算复杂度

- 问题的计算复杂度分析
- 决策树模型
- 检索问题的时间复杂度

  决策树
- 冒泡排序算法
- 堆排序算法

- 堆排序算法的时间复杂度分析
- 排序算法的决策树
- 选最大及最小问题的时间复杂度分析

  构造最坏输入
- 一般选择问题的时间复杂度分析

  构造最坏输入
- 通过归约估计问题难度的下界

决策树与问题复杂度

决策树的特点：

- 以比较作基本运算的算法模型
- 一个问题确定了一类决策树，具有相同的构造规则，该决策树类决定了求解该问题的一个算法类
- 结点数(或树叶数)等于输入规模
- 最坏情况下的时间复杂度对应于决策树的深度
- 平均情况下的时间复杂度对应于决策树的平均路径长度

用决策树模型界定确定问题难度

- 给定结点数(或树叶数)的决策树的深度至少是多少？
- 给定结点数(或树叶数)的决策树的平均路径长度至少是多少？

NP完全性

NP完全性理论

- 易解的问题与难解的问题
- 判定问题
- **NP**类
- 多项式时间变换
- **NP**完全性及其性质
- **Cook-Levin**定理

多项式时间变换

如何比较两个问题的难度?

定义9.4 设判定问题 $\Pi_1 = \langle D_1, Y_1 \rangle$, $\Pi_2 = \langle D_2, Y_2 \rangle$. 如果函数 $f : D_1 \rightarrow D_2$ 满足条件:

(1) f 是多项式时间可计算的,

(2) 对所有的 $I \in D_1$, $I \in Y_1 \Leftrightarrow f(I) \in Y_2$,

则称 f 是Π_1 到 Π_2 的多项式时间变换.

如果存在 Π_1 到 Π_2 的多项式时间变换, 则称 Π_1 可多项式时间变换到 Π_2, 记作 $\Pi_1 \leq_p \Pi_2$.

NP完全性

定义 9.5 如果对所有的 $\Pi'\in$NP, $\Pi'\le_p \Pi$, 则称Π是 **NP难的**. 如果 Π 是 NP 难的且 $\Pi\in$NP, 则称 Π 是 **NP完全的**.

NP 完全问题是 NP 中最难的问题.

定理 9.4 如果存在 NP 难的问题 $\Pi\in$P, 则 P = NP.

推论 9.2 假设P ≠ NP, 那么, 如果 Π是 NP难的, 则 $\Pi\notin$P.

定理 9.5 如果存在 NP 难的问题 Π' 使得 $\Pi'\le_p \Pi$, 则 Π 是 NP 难的.

推论 9.3 如果 $\Pi\in$NP 并且存在 NP 完全问题 Π' 使得 $\Pi'\le_p \Pi$, 则Π 是NP完全的.

证明 NP完全性的“捷径”

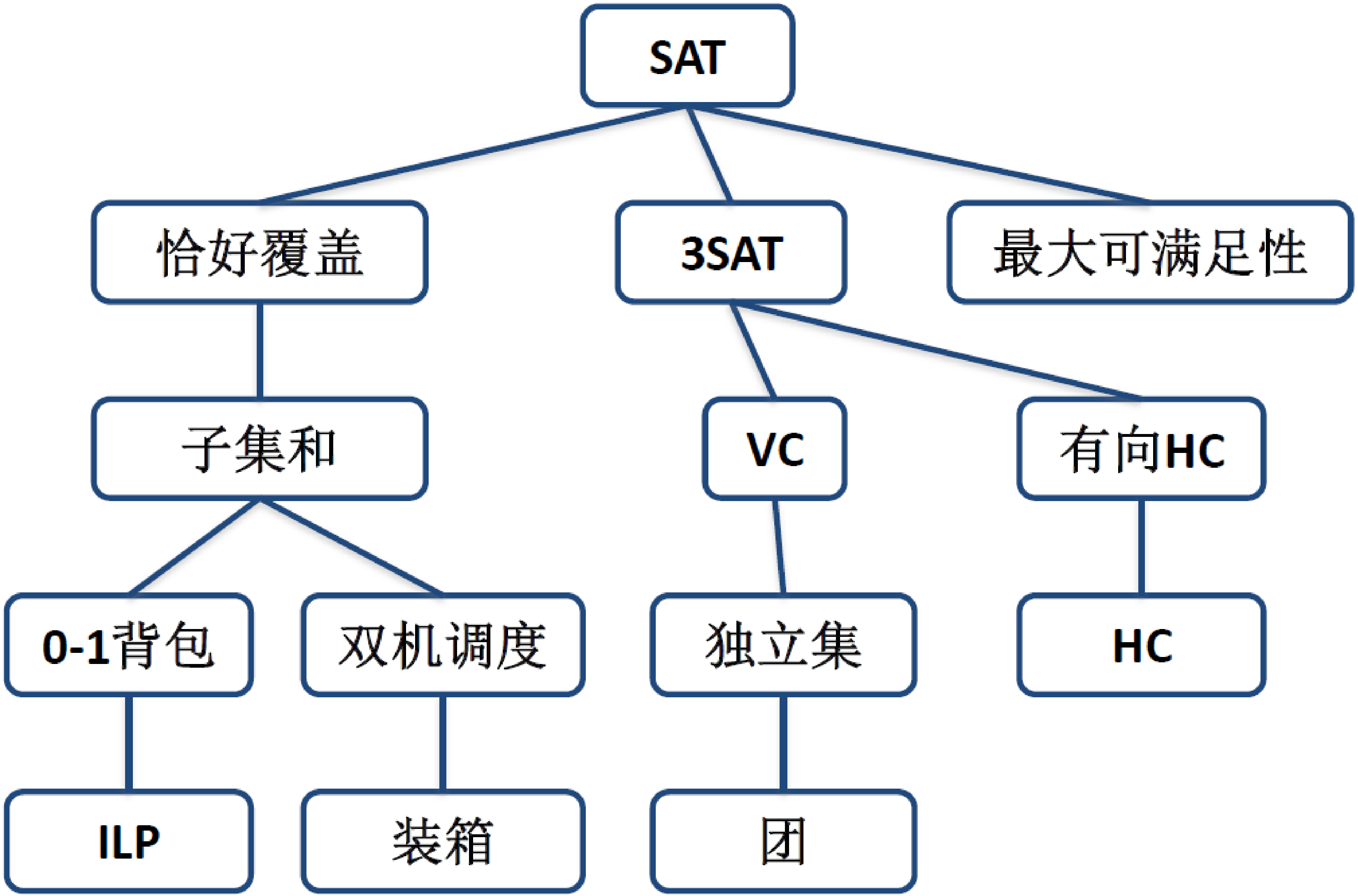
为了证明 Π 是 NP完全的, 只需要做两件事:

(1) 证明 $\Pi \in$NP;

(2) 找到一个已知的 NP 完全问题 Π', 并证明 $\Pi' \leq_p \Pi$.

问题：是否存在NP完全问题？如果存在，如何找到第一个NP完全问题?

几个NP完全性问题



NP难的证明小结

- 证明一个问题是NP难的，
  首先要选择好一个已知的NP完全问题。
- 理论上，任何一个NP完全问题都能够多项式时间变换到一个NP难的问题。
- 但找到一个方便做多项式时间变换的NP完全问题往往并不容易。
- 课程中的证明主要使用了下述方法：
  - 限制法、局部替换法和构件设计法

限制法

- 如果限制问题$\boldsymbol{\Pi}$的实例满足某些条件时得到的子问题$\boldsymbol{\pi}'$已知是NP完全的。则很容易把$\boldsymbol{\pi}'$多项式时间变换到问题$\boldsymbol{\Pi}$
- 例子
  - SAT$\leq_p$最大可满足性
  - 子集和$\leq_p$0-1背包

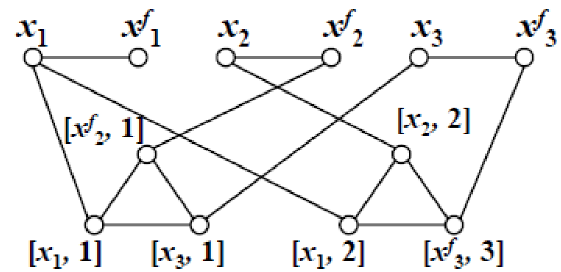
局部替换法

- 如果问题$\boldsymbol{\Pi}$已知是NP完全的，要证明其子问题$\boldsymbol{\pi}'$也是NP完全的则要困难得多
- 但通常问题与子问题存在相似的结构
- 局部替换法可以把问题$\boldsymbol{\Pi}$的实例中某种“基本单位”替换成满足给定限制的子问题的结构
- 例子
  - SAT $\leq_p$3SAT
  - 有向HC $\leq_p$ HC

构件设计法

- 这种方法的技巧性最强
- 通常在上述两种方法（限制法和局部替换法）都失效时所采用的方法。
- 3SAT由于其简单整齐的结构，常常被用作已知NP完全问题，通过构件设计法构造多项式时间变换
- 例子
  - 3SAT $\leq_p$VC
  - 3SAT $\leq_p$有向HC

构件设计法



上述证明中设计了2种“构件”

—— 变元构件 和 简单析取式构件。

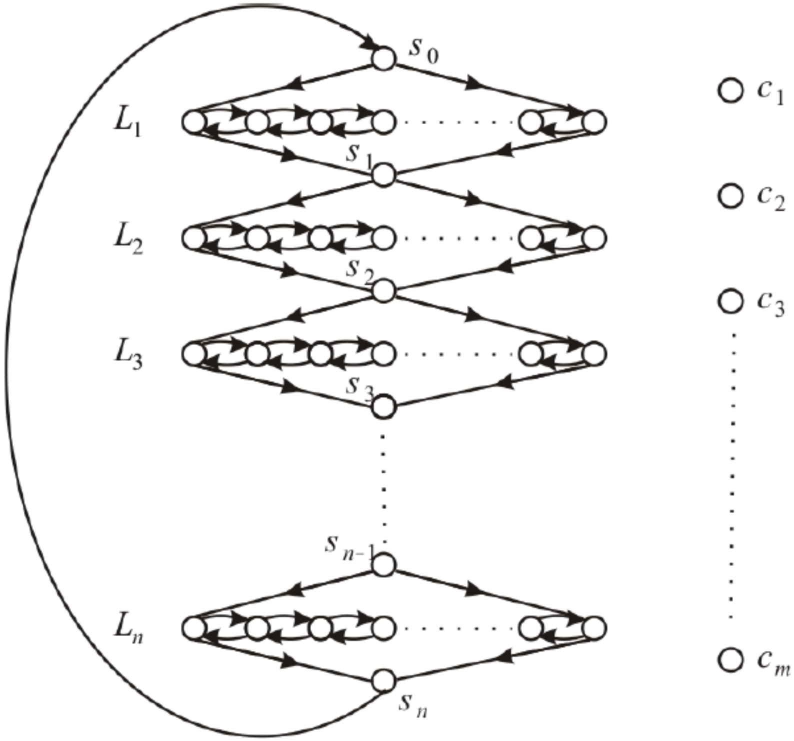
- 变元构件是一对顶点x_i, x_i^f及连接它们的边；
- 简单析取式构件是三角形。

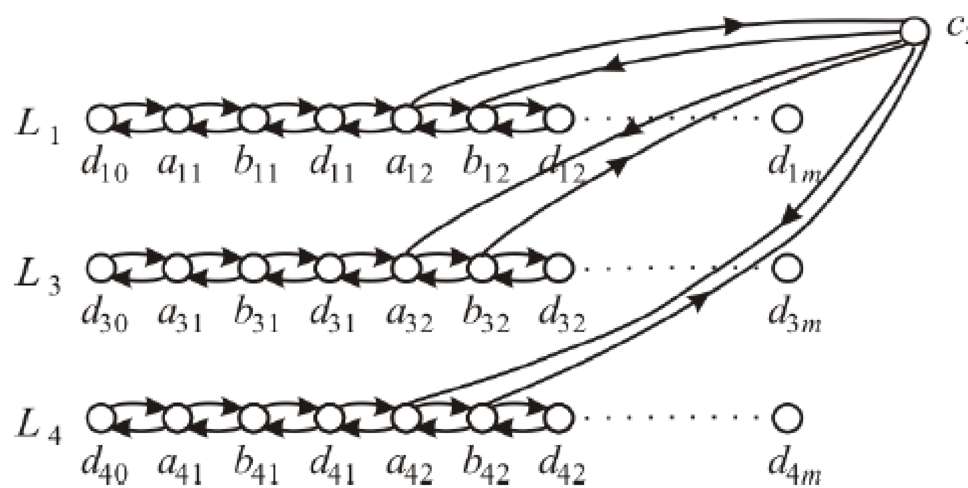
用这些构件及构件之间的连接构成G，

每个构件各有其功能，通过这种方式到达

用VC的实例表达3SAT的实例的目的。

有向哈密顿回路（证明续）


s_0
L_1
s_1
L_2
s_2
L_3
s_3
s_{n-1}
L_n
s_n
c_1
c_2
c_3
c_m


c_2
L_1
d_{10} a_{11} b_{11} d_{11} a_{12} b_{12} d_{12} d_{1m}
L_3
d_{30} a_{31} b_{31} d_{31} a_{32} b_{32} d_{32} d_{3m}
L_4
d_{40} a_{41} b_{41} d_{41} a_{42} b_{42} d_{42} d_{4m}

近似算法

关于问题是否可近似计算的分析

多机调度
贪心法G-MPS
算法DG-MPS

货郎问题
最邻近法
最小生成树法
最小匹配法
可近似性估计

0-1背包
贪心法G-KK
算法PTAS
算法FPTAS

近似算法的设计思想与分析方法

指标要求

时间——多项式时间

性能——近似比为常数

一个例子：最小顶点覆盖

算法A的近似比r

设OPT(I)表示实例 I 的最优解的值，

(1)Π是最小化问题，$r_A(I)$=A(I)/OPT(I)

(2)Π是最大化问题，$r_A(I)$=OPT(I)/A(I)

最优化算法A: 对所有的实例 I 恒有

A(I)=OPT(I)，即 $r_A(I) = 1$.

A的近似比为 r (A是r–近似算法):

对每一个实例 I，$r_A(I) \le r$.

A具有常数近似比： r是一个常数.

随机算法及应用

处理NP难问题的策略

随机算法的分类及局限性

主元素测试
串相等测试

素数测试

蒙特卡洛型随机算法

随机快速排序与随机选择算法

*n*后问题随机算法

拉斯维加斯型随机算法

概率论基本知识

随机算法基本概念

随机算法的分类

常见的两类随机算法：

- Las Vegas算法运行结束时总能给出正确的解，但其运行时间每次有所不同。
- Monte Carlo算法可能得到不正确的结果，但这种概率是小的且有界的。

重复性定律

- 设 ε 是一次随机试验的成功概率，则N次独立随机试验的成功概率为

$$P=1-(1-\varepsilon)^N \approx N\varepsilon$$

- 例如： $\varepsilon=0.05$，$N=20$，$P\approx 1$

- 最小割问题
- 亚线性时算法（全0数组判定、数组有序性判定、最小生成树）
- 有限马氏链（2SAT的随机游动算法）

处理难解问题的策略

1　对问题施加限制（ 2SAT、霍恩 SAT 问题）

2　固定参数算法（顶点覆盖问题）

3　改进指数时间算法（3SAT问题）

4　启发式方法（局部搜索法：模拟退火法、Hopfield神经网络、最大割、纳什均衡）

5　平均情形的复杂性（哈密顿回路的概率图模型）

6　难解算例生成（可满足性相变、RB模型产生随机实例）

7　基于统计物理的消息传递算法（随机 3SAT 的 SP 算法）

8　量子算法（道奇算法）