PKU-04833060
算法设计与分析『实验班』

第13讲 随机算法 (上)

罗国杰

gluo@pku.edu.cn

2025年春季学期

复习：上一讲“近似算法”的内容小结

- 贪心策略
  - 多机调度的贪心、改进贪心
  - 0-1背包的简单贪心
- 构造上界
  - 三角不等式TSP的最小生成树
  - 三角不等式TSP的最小生成树+最小权匹配
- 伪多项式 + 值缩放
  - 0-1背包的FPTAS算法
- 线性规划
  - 对贪心策略的解释
  - 线性松弛 + 舍入 → 随机舍入
  - 原始-对偶模式
- 随机算法

近似算法∩随机算法

- 用随机算法计算近似解
- 随机算法的期望代价E[C]
- 如果任意给定输入大小n，都有
  - $\max(E[C]/C^*, C^*/E[C]) \le \rho(n)$
  - 那么随机算法的近似比例为 $\rho(n)$
    该算法为随机$\rho(n)$-近似算法
- 随机近似算法和确定近似算法唯一的区别：其代价是期望代价

MAX-3SAT 可满足问题

- MAX-3SAT 可满足问题
  - 给定 3SAT 公式，如果不能使所有子句（简单析取式）都满足，也要最大化可满足的子句个数
  - 假设每个子句都有三个不同的文字，而且不允许 x 和 ¬x 同时出现在同一个子句中
- 如果对每个变量随机分配0-1值（概率是 0.5 : 0.5），那么可以证明这种随机算法实际上是随机8/7-近似算法
  - “8/7”：算法期望代价的近似比

MAX-3SAT 可满足问题的随机近似算法 (1/2)

定理：假设有一个 **MAX-3SAT** 可满足性问题的实例，有 ***n*** 个变量 $x_1, x_2, \ldots, x_n$ 和 ***m*** 个子句，如果对每个随机变量随机赋值，为 **1** 的概率为 **0.5**，为 **0** 的概率也是 **0.5**，那么这个随机算法是随机 **8/7-**近似算法。

证明：定义指示器随机变量 Y_i 表示子句 $c_i = z_{i,1} \vee z_{i,2} \vee z_{i,3}$ 是否被满足。

（随机事件：$Y_i = 1$ 表示 c_i 被满足；$Y_i = 0$ 表示 c_i 不被满足）

显然，只要子句 c_i 中某一个文字被赋值为 1，则 $Y_i = 1$。

因为每个子句都有三个不同的文字（即 x 和 $\neg x$ 不同时出现在同一个子句中），则子句中三个文字的赋值是彼此独立的。只有当其中的三个文字都被赋值为 0 时子句才为 0，这个概率为

$$Pr[Y_i = 0] = (1/2)^3 = 1/8$$

MAX-3SAT 可满足问题的随机近似算法 (2/2)

因此，$Pr[Y_i = 1] \geq 1 - 1/8 = 7/8$。于是

$$E[Y_i] \geq 7/8$$

设随机变量 Y 是可以满足的子句的个数，那么

$$Y = Y_1 + Y_2 + \cdots + Y_m$$

于是

$$E[Y] = E[\sum_{i=1}^{m} Y_i] = \sum_{i=1}^{m} E[Y_i] \geq \sum_{i=1}^{m} 7/8 = 7m/8$$

由于 m 是可满足子句数的上界，所以该算法的随机近似比为 $8/7$：

$$m/E[Y] \leq m/(7m/8) = 8/7$$

算法里的概率分析和随机策略

- 被动随机：客观世界的不确定性/概率模型
  - 例如：平均情况分析
- 主动随机：算法行为或决策的随机性
  - 例如：高概率产生正确解的算法、期望运行时间是多项式的算法、等等
  - 应用场景举例：分布式系统的随机算法，减少进程的交互或同步，又能保证某些全局性质

竞争解决问题（Contention Resolution）

- 竞争解决问题
  - 给定 n 个进程 P1, P2, ..., Pn，以及一个共享的数据库 DB
  - 以离散的 “轮” (round) 为单位计算时间
  - 如果每轮尝试访问 DB 的进程数超过一个，冲突将造成此轮所有进程的访问均无效
  - 假设进程间不能通信，为使进程能够“轮流”访问 DB，该如何安排访问策略?
- 一个随机算法
  - 设置全局参数 0 < p < 1，每个进程每轮独立地以 p 概率尝试访问 DB
    - 极端情况 p=0 和 p=1 均没有进程能成功访问 DB
  - 问题：参数 p 的最优值?
  - 问题：特定进程 Pi 成功访问 DB 的时间轮数?
  - 问题：任意进程均至少成功访问 DB 一次的时间轮数（i.e., 算法步数）？

“13.1 A First Application: Contention Resolution”, in J. Kleinberg and E. Tardos, Algorithm Design. Pearson Education, 2006.

竞争解决问题：参数 p 的最优值

- 随机算法：每个进程每轮独立地以 p 概率尝试访问 DB
- 随机事件 $\mathcal{A}(i,t)$：进程 Pi 在第 t 轮尝试访问 DB
  - 显然，$Pr[\mathcal{A}(i,t)] = p$ 且 $Pr[\overline{\mathcal{A}(i,t)}] = 1-p$
- 随机事件 $\mathcal{S}(i,t)$：进程 Pi 在第 t 轮成功访问 DB
  - $\mathcal{S}(i,t) = \mathcal{A}(i,t) \cap \left(\bigcap_{j\neq i}\overline{\mathcal{A}(j,t)}\right)$
  - $Pr[\mathcal{S}(i,t)] = p \cdot (1-p)^{n-1}$
  - 上述概率在 $p = 1/n$ 时取得最大值 $Pr^*[\mathcal{S}(i,t)] = 1/n \cdot (1-1/n)^{n-1}$
    - 当 n 从 2 增长到 $+\infty$，$(1-1/n)^n$ 从 1/4 增长至 1/e
    - 当 n 从 2 增长到 $+\infty$，$(1-1/n)^{n-1} = (1-1/n)^n \cdot n/(n-1)$ 从 1/2 降低至 1/e
    - 得到 $1/(en) \leq Pr^*[\mathcal{S}(i,t)] \leq 1/(2n)$
  - 因此，$Pr^*[\mathcal{S}(i,t)] = \Theta(1/n)$

竞争解决问题：随机算法的性质分析

(以下假设 p=1/n)

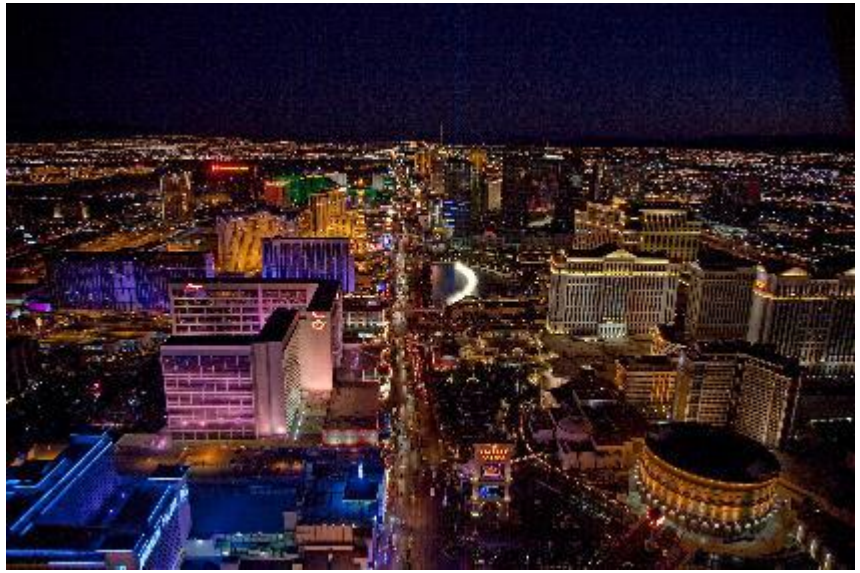
- 随机事件 $\mathcal{F}(i,t)$：进程 Pi 从第 1 轮至第 t 轮均未能成功访问 DB
  - $Pr[\mathcal{F}(i,t)] = Pr[\cap_{r=1}^{t}\overline{\mathcal{S}(i,r)}] = \prod_{r=1}^{t} Pr[\overline{\mathcal{S}(i,r)}] = (1-1/n\cdot(1-1/n)^{n-1})^{t} \le (1-1/(en))^{t}$
- 经过 $\Theta(n)$ 轮，Pi 未成功访问 DB 的概率低于 $1/e$
  - 由于 $Pr[\mathcal{F}(i,\lceil en\rceil)] \le 1/e$
- 经过 $\Theta(n\ln n)$ 轮，Pi 未成功访问 DB 的概率低于 $(1/n)^{c}$
  - 由于 $Pr[\mathcal{F}(i,\lceil en\rceil\cdot(c\ln n))] \le (1/e)^{c\ln n} = (1/n)^{c}$

竞争解决问题：随机算法的性质分析

（以下假设 p=1/n）

- 随机事件 $\mathcal{F}_t$：存在进程在第 1 轮至第 t 轮内均未成功访问 DB
  - $\mathcal{F}_t = \bigcup_{i=1}^{n} \mathcal{F}(i,t)$
- 基础知识：并的界（Union Bound）
  - $Pr[\bigcup_{i=1}^{n} \mathcal{E}_i] \leq \sum_{i=1}^{n} Pr[\mathcal{E}_i]$
- 算法经过 $t = 2\lceil en \rceil \ln n$ 轮，满足 $Pr[\mathcal{F}_t] \leq \sum_{i=1}^{n} Pr[\mathcal{F}(i,t)] \leq n \cdot (1/n)^2 = 1/n$

拉斯维加斯（Las Vegas）



蒙特卡洛（Monte Carlo）

随机算法的分类与局限性

- 有效的拉斯维加斯型随机算法
  - (sometimes fast but always correct)
  - 零错误概率，ZPP (zero-error probabilistic polynomial time)
- 蒙特卡洛型随机算法
  - (always fast and probably correct)
  - 单侧错误概率，RP (randomized polynomial time)，coRP
  - 双侧错误概率，BPP (bounded-error probabilistic polynomial)
- 随机算法的局限性
  - 错误概率有界的多项式时间随机算法不太可能解决NP完全问题

拉斯维加斯型随机算法

- 拉斯维加斯型算法的特征
  - 这种算法的结果总是正确的，区别只在于运行时间的长短
  - 拉斯维加斯型随机算法的运行时间是一个随机变量
    - 例子：快速排序算法总是给出已经排序的数组，期望的时间则是 2n ln n.
- 有效的拉斯维加斯型算法
  - 算法期望运行时间是输入规模的多项式，总是给出正确答案
  - 快速排序算法是有效的拉斯维加斯型算法

随机快速排序算法

算法　随机快速排序算法

输入：包含 n 个元素的数组

输出：经过排序的 n 个元素的数组

1. 若数组包含 0 或 1 个元素则返回
2. 从数组中随机选择一个元素作为枢轴元素
3. 把数组元素分为三个子数组，并且按照 A, B, C 顺序排列
   - A：包含比枢轴元素小的元素；
   - B：包含与枢轴元素相等的元素；
   - C：包含比枢轴元素大的元素.
4. 对 A 和 C 递归地执行上述步骤.

算法分析

定理 设数组含n个不同元素，对任意常数$\varepsilon>0$，随机快速排序算法的期望比较次数

$$T(n) \le 2\, n \ln n.$$

证明：求解递推式

随机选取枢轴元素，其位于排序后第 i 位置 (i=1,2,…,n)的概率是1/n，A和C的元素数分别是 i 个和 n–i–1个, 得

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}[T(i) + T(n-i-1)] = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} T(i)$$

解为$\Theta(n\log n)$. 可归纳证明精确上界是 $T(n) \le 2n \ln n$.

$$T(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} 2i\ln i \le (n-1) + \frac{2}{n}\int_1^n 2x\ln x dx$$

$$\le (n-1) + \frac{2}{n}\left(n^2\ln n - \frac{n^2}{2} + \frac{1}{2}\right) \le 2n\ln n$$

n后放置的随机选择

算法BoolQueen(*n*)

```
1. k←1                                  // k 放皇后的行号
2. count←0                              // count 放好的皇后数
3. while k≤n do
4.     for i←1 to n do                  // i 为待选列号
5.         检查 i 与前面 k-1个皇后的相容性
6.         如果相容则将 i 加入S
7.     if S≠∅ then
8.         j←Random(1,|S|)
9.         x_k←S[j]
10.        count←count+1
11.        k←k+1
12.    else k←n+1
13. return count
```

n后问题的随机算法

算法QueenLV(*n*)　　//重复调用随机算法BoolQueen

1. *p*←BoolQueen(*n*)
2. while *p*<*n* do
3. 　*p*←BoolQueen(*n*)

改进算法---与回溯相结合

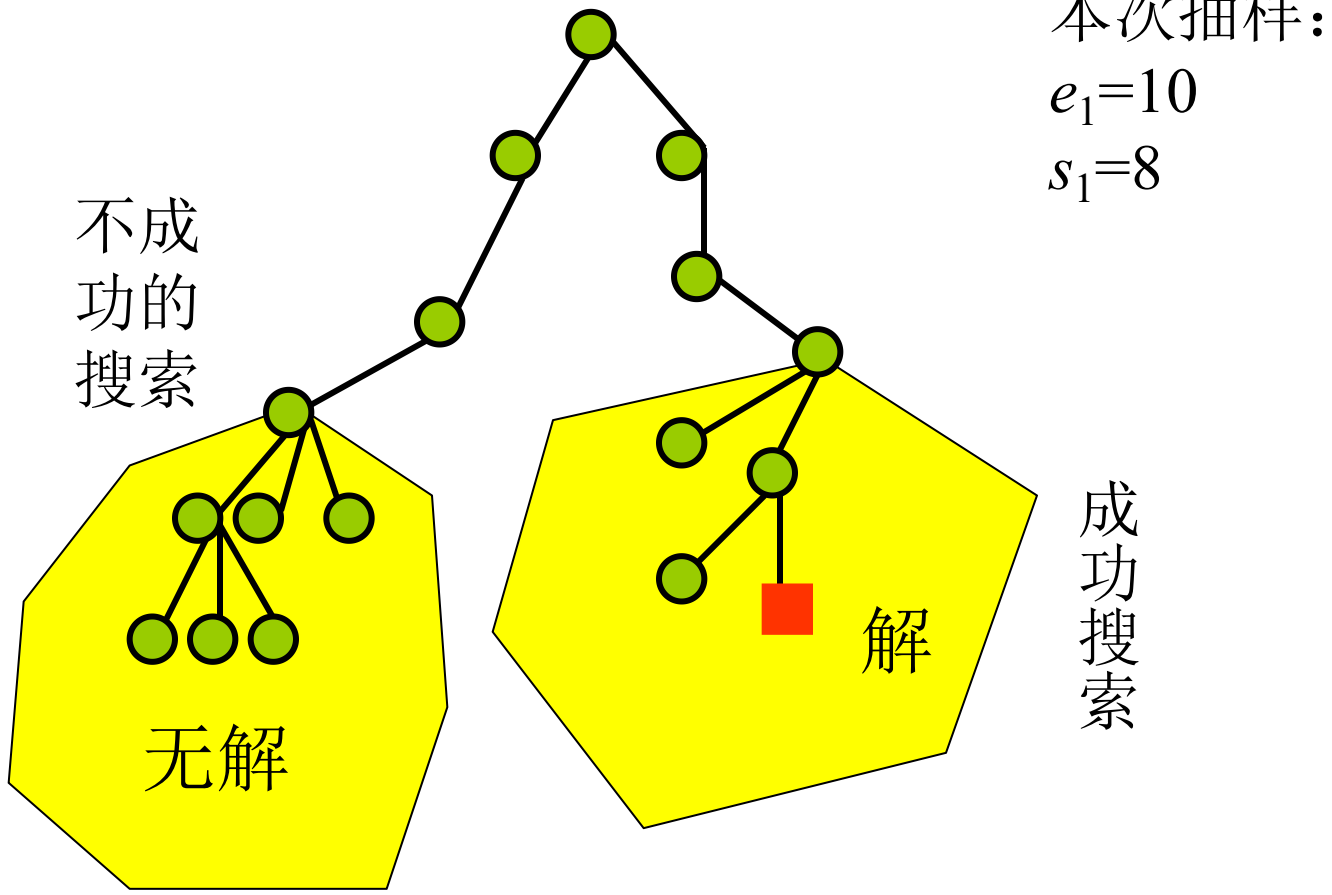
设 *stopVegas* ≤ *n*, 表示用QueenLV算法放置的皇后数

剩下 *n* - *stopVegas* 个皇后用回溯方法放置

stopVegas = 0 时是完全的回溯算法

stopVegas = *n* 时是完全的Las Vegas算法

成功搜索与不成功搜索



改进算法的分析

对于不同的 *stopVegas* 值，设

p 为算法成功概率

s 为一次成功搜索访问的结点数的平均值

e 为一次不成功搜索访问的结点数的平均值

t 为算法找到一个解的平均时间

$$t = ps + (1-p)(e+t) \Rightarrow t = s + e\frac{1-p}{p}$$

n=12时的统计数据：*stopVegas* = 5时算法效率高

stopVegas	*p*	*s*	*e*	*t*
0	1.0000	262.00	-	262.00
5	0.5039	33.88	47.23	80.39
12	0.0465	13.00	10.20	222.11

T. J. Rolfe, "Las Vegas does n-queens," ACM SIGCSE Bull., vol. 38, no. 2, pp. 37–38, Jun. 2006, doi: 10.1145/1138403.1138429.

Min-Conflict Algorithms for N-Queen (and CSP)

nearly linear run time

```
algorithm MIN-CONFLICTS is
    input: console.csp, A constraint satisfaction problem.
           max_steps, The number of steps allowed before giving up.
           current_state, An initial assignment of values for the variables in the
csp.
    output: A solution set of values for the variable or failure.

    for i ← 1 to max_steps do
        if current_state is a solution of csp then
            return current_state
        set var ← a randomly chosen variable from the set of conflicted variables
CONFLICTED[csp]
        set value ← the value v for var that minimizes
CONFLICTS(var, v, current_state, csp)
        set var ← value in current_state

    return failure
```

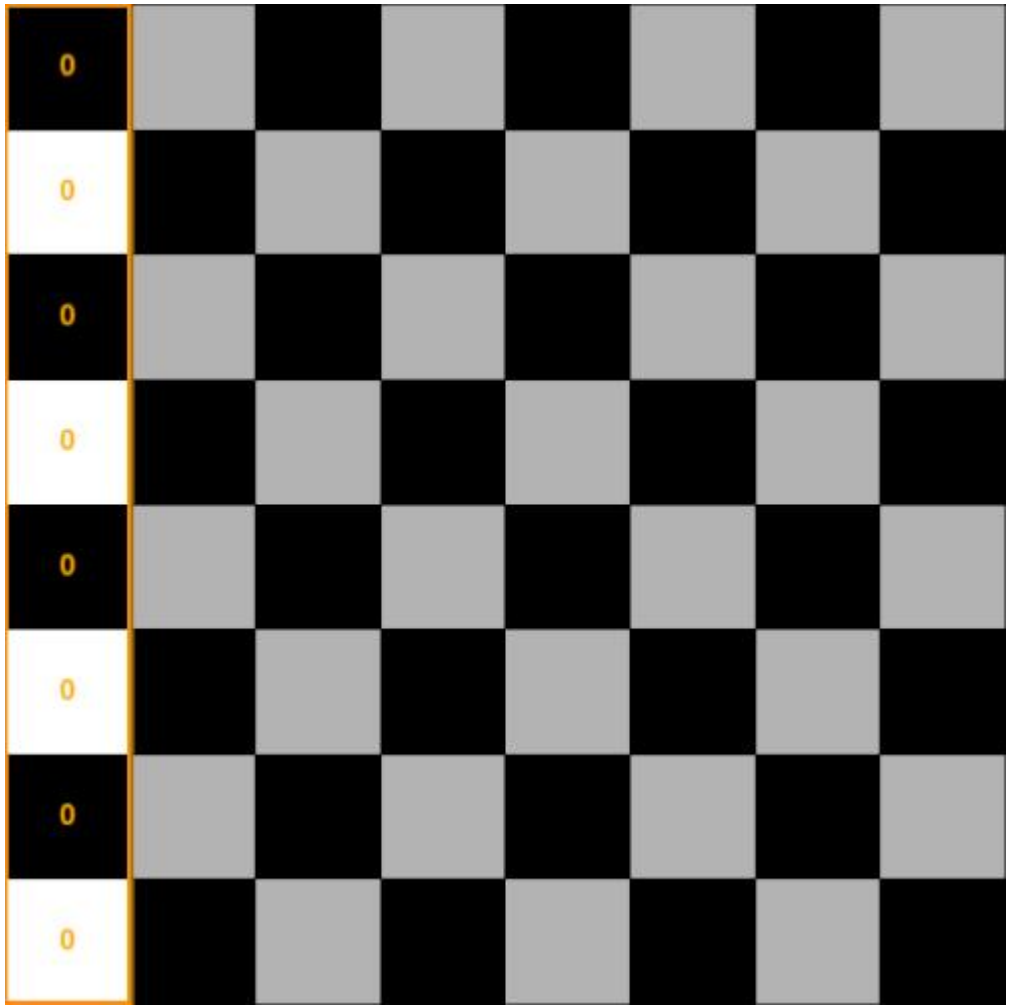


image source: wikipedia.org

S. Minton, et al.. "Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method," AAAI 1990.

拉斯维加斯型随机算法小结

- 拉斯维加斯型算法的特点
  - 通过修改确定性算法得到，一般将算法的某步的确定型选择变成随机选择
  - 一次运行可能得不到解；若得到解，则解一定是正确的
  - 改进途径：与确定型算法相结合
  - 有可能改进确定型算法平均情况下的时间复杂度
- 有效的拉斯维加斯型算法
  - 运行时间是随机变量，期望运行时间是输入的多项式且总能给出正确答案的随机算法

蒙特卡洛型随机算法

- 蒙特卡洛型算法的特征
  - 这种算法有时会给出错误的答案.
  - 其运行时间和出错概率都是随机变量，通常需要分析算法的出错概率
- 总是在多项式时间内运行且出错概率有界的随机算法称为有效的蒙特卡洛型算法
  - 只要让算法独立重复执行足够多次，出错概率可以降到任意小
- 例子：素数检验、多项式恒等检验的随机算法和布尔可满足性问题的随机游动算法都是这类算法

单侧错误和双侧错误

- 弃真型单侧错误（RP）
  - 如果 $I \in Y$，则 $Pr[A(I) = YES] \geq 1/2$
  - 如果 $I \notin Y$，则 $Pr[A(I) = NO] = 1$

  （**当算法宣布接受时，结果一定是对的**；当算法宣布拒绝时，结果有可能是错的）
  - 例如，后面将要介绍的 2SAT 随机游动 算法
- 取伪型单侧错误（co-RP）
  - 如果 $I \in Y$，则 $Pr[A(I) = YES] = 1$
  - 如果 $I \notin Y$，则 $Pr[A(I) = NO] \geq 1/2$

  （当算法宣布接受时，结果有可能是错的；**当算法宣布拒绝时，结果一定是对的**）
  - 例如，后面将要介绍的 矩阵乘法检验 算法
- 双侧错误
  - 所有输入上同时出现上述两种不同的错误

矩阵乘法检验的随机算法（co-RP 算法例子）

- 矩阵乘法检验问题
  - 输入：n×n 的矩阵 A, B, C；输出：是否满足 AB = C
  - *假设矩阵元素只有{0,1}，加法结果模2*；i.e., 元素运算在 GF(2) 中
- 朴素的检验方法
  - 直接检查矩阵乘法结果，需要 $O(n^3)$（直接相乘）或 $O(n^{2.371552})$（Williams-Xu-Xu-Zhou）
- 随机的检验方法
  - 构造 n 维向量 r，其中 $Pr[r_i=1]=1/2$
  - 检查 A Br =? Cr：如果相等，输出 YES；否则输出 NO
  - 时间 $O(n^2)$，三次矩阵-向量乘法：Br、A Br 和 Cr

矩阵乘法检验的随机算法：证明

- 构造 n 维向量 r，其中 $Pr[r_i=1]=1/2$
- 若 AB≠C，对每个满足 A Br = Cr 的向量 r，都存在向量 r'=r+v 满足 A Br' ≠ Cr'
  - 设 D = AB - C，则 AB≠C ⇔ ∃ dij≠0
  - 其中 v 是只有元素 $v_j=1$ 的单位向量
  - 由于 $d_{ij}≠0$，有 $(Dv)_i = 1$ 和 Dv ≠ 0
  - 可得 Dr' = Dr + Dv = 0 + Dv ≠ 0
- 因此，若 AB≠C，有 Pr[A Br = Cr] ≤ Pr[A Br ≠ Cr]，即 Pr[A Br ≠ Cr] ≥ 1/2
- 该随机算法是 co-RP 的
  - 若 AB=C，有 Pr[Dr=0] = 1（输出 YES）
  - 若 AB≠C，有 Pr[Dr=1] ≥ 1/2 （输出 NO）

2SAT 的随机游动算法（RP 算法例子）

输入：一个有n个变元和m个子句的合取范式，每个子句刚好有两个文字

输出：一个满足赋值，或宣布没有满足赋值

1. 任意给所有变量赋值；
2. 若当前赋值是满足赋值，则输出这个赋值，算法结束；
3. 均匀随机选一个不满足子句，从中均匀随机选一个文字，改变该文字的赋值；
4. 重复第2-3步$2mn^2$次，若一直没有找到满足赋值，则宣布没有满足赋值.

2SAT 的随机游动算法：定理

- 定理 若输入公式是不可满足的，则上述随机游动算法正确宣布其不可满足. 若输入公式是可满足的，则上述算法以$1-1/2^m$概率找到满足赋值.
- 证明　显然，当输入公式是不可满足的，算法无法找到满足赋值，因此将宣布不可满足. 当输入公式可满足时，算法有可能找不到满足赋值而出错. 我们证明算法出错的概率为$1/2^m$.

2SAT 的随机游动算法：定理证明（续）

- 假设输入公式可满足，
  - 设a^*是某个可满足赋值，
  - 设a_t是算法在执行 t 遍第3步以后的赋值，
  - 设 X_t 等于a^*和 a_t 赋值相同的变量个数.
- 令变量数为n，
  - 则当 $X_t=n$ 时$a_t=a^*$，算法将找到满足赋值a^*而停止.

（注意：在 $X_t=n$之前，算法也可能找到别的满足赋值而停止.）

2SAT 的随机游动算法：定理证明（续一）

考虑 X_t 和 X_{t+1} 的关系

当 $X_t=0$ 时，显然有 $X_{t+1}=1$ ，所以

$$\Pr[X_{t+1}=1|X_t=0]=1.$$

当 $X_t=j$ 且 $0<j\le n-1$ 时，$X_{t+1}=j+1$ 或 $X_{t+1}=j-1$。假设算法选择修改子句 $C=a\vee b$ 的一个变量的赋值，则 C 是不满足子句，所以在赋值 a_t 之下 $a=b=$ 假。在 a^* 下，$a=$ 真且 $b=$ 假、或 $a=$ 假且 $b=$ 真、或 $a=b=$ 真。因此随机选择改变 a 或 b 的赋值时，至少有一半机会让 a^* 和 a_t 赋值相同的变量个数增加1个。所以

$$\Pr[\,X_{t+1}=j+1|X_t=j\,]\ge 1/2,\quad \Pr[\,X_{t+1}=j-1|X_t=j\,]\le 1/2.$$

（注意：$\{X_0, X_1, X_2,\ldots\}$ 可能不是马氏链.）

2SAT 的随机游动算法：离散随机过程

- 一个离散随机过程就是一组随机变量

$$X = \{X_t \mid t \in T,\ T\text{是可数集}\}$$

通常 t 代表时间，X_t 就是 X 在时刻 t 的状态。

- 例如，赌徒每次抛一枚均匀硬币来赌博
  - 若正面向上，赢1元钱；反面向上，就输1元钱
  - 假设初始赌本为X_0，在时刻 t 的赌本为X_t，
    则 $\{X_t \mid t=0,1,2,\ldots\}$就是一个离散随机过程。

2SAT 的随机游动算法：有限马氏链

- 一个有限马氏链是满足下列条件的离散随机过程 $\{X_0, X_1, X_2, \ldots\}$，其中每个$X_t$都从一个有限集中取值

$$\Pr[X_t=a|X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$

$$= \Pr[X_t=a|X_{t-1}=b] = p_{b,a}$$

即X_t的值依赖于X_{t-1}的值，而不依赖之前的历史.

- 在上述赌徒的例子中，当 $b \neq 0$ 或 n 时，

$$\Pr[X_t=b+1 | X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$

$$= \Pr[X_t= b+1|X_{t-1}=b] = 1/2$$

$$\Pr[X_t=b-1 | X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$

$$= \Pr[X_t= b-1|X_{t-1}=b] = 1/2$$

即

$$p_{b,b+1} = p_{b,b-1} = 1/2$$

2SAT 的随机游动算法：一步转移矩阵

- 对于有限马氏链，不妨设 X_t 取值的状态空间为 $\{0,1,2,\ldots,n\}$. 于是 $p_{i,j}$ 可组成一个$n+1$阶方阵

$$P = [p_{i,j}]_{(n+1)\times(n+1)},$$

叫做一步转移矩阵. 其每一行元素之和等于1，即对任意 i，有$\Sigma_j\, p_{i,j} = 1$.

2SAT 的随机游动算法：m步转移矩阵

- 设 $p_i(t)$ 表示在时刻 t 处在状态 i 的概率，则

$$p_i(t) = p_0(t-1)p_{0,i} + p_1(t-1)p_{1,i} + \ldots + p_{n-1}(t-1)p_{n-1,i} + p_n(t-1)p_{n,i}$$

- 设$p(t)$表示向量$(p_0(t), p_1(t), \ldots, p_{n-1}(t), p_n(t))$，则

$$p(t) = p(t-1)\ P.$$

- 对任意m，定义**m步转移矩阵**，

$$P^{(m)} = [p_{i,j}^{(m)}]_{(n+1)\times(n+1)}$$

其中

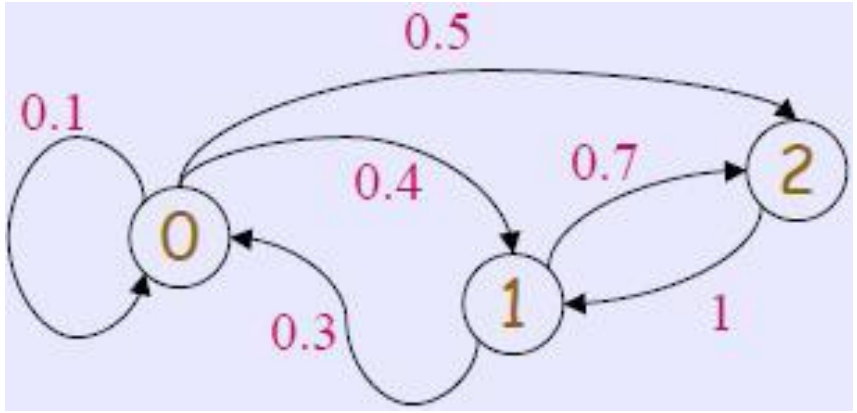
$$p_{i,j}^{(m)} = \Pr[X_{t+m} = j \mid X_t = i]$$

- 于是 $P^{(m)}=P^m$

$$p(t+m)=p(t)P^m$$

2SAT 的随机游动算法：有限马氏链的图示

- 有限马氏链还可以用有向带权图来表示
  - 每个状态作为一个顶点
  - 两个状态 i, j之间的有向边所带的权就是这两个状态之间的转移概率 $p_{i,j}$
- 如图所示就是一个三状态马氏链.



2SAT 的随机游动算法：定理证明（续二）

定义真正的马氏链$\{Y_0, Y_1, Y_2, \ldots\}$如下：

$Y_0 = X_0$

$\Pr[Y_{t+1}=1 | Y_t=0]=1$

$\Pr[Y_{t+1}=j+1 | Y_t=j]=1/2$

$\Pr[Y_{t+1}=j-1 | Y_t=j]=1/2$

注意到

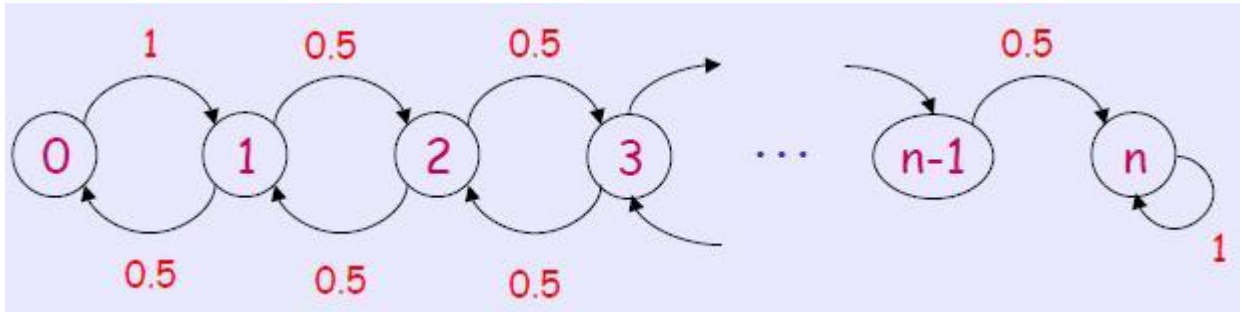
$\Pr[\ Y_{t+1}=j+1 | Y_t=j\ ] = 1/2 \le \Pr[X_{t+1}=j+1 | X_t=j]$

$\Pr[\ Y_{t+1}=j-1 | Y_t=j\ ] = 1/2 \ge \Pr[\ X_{t+1}=j-1 | X_t=j\ ]$

从任意状态出发，Y_t将比 X_t 花费更长的期望时间才能进入状态n.

2SAT 的随机游动算法：定理证明（续三）

- 下面我们分析Y_t进入状态n所需要的期望时间，以此作为算法期望运行时间的上界.
- 注意马氏链$Y=\{Y_0, Y_1, Y_2,\ldots\}$可以用下图所示的带权有向图表示.



- 设h_j表示从状态j到达状态n的期望运行时间
  - $h_n=0$和$h_0=h_1+1$.
  - 对于其他 j 值 $$h_j = \frac{1}{2}(h_{j+1}+1)+\frac{1}{2}(h_{j-1}+1)$$

2SAT 的随机游动算法：定理证明（续四）

用归纳法易证：对于所有j，

$$h_j \le n^2 - j^2 \le n^2$$

$$\mathrm{E}[X\text{从}X_0\text{到达}X_t=\mathrm{n}\text{的时间}]$$

$$\le \mathrm{E}[Y\text{从}Y_0\text{到达}Y_t=\mathrm{n}\text{的时间}] \le n^2$$

由马尔科夫不等式，就有

$$\Pr[X\text{从}X_0\text{到达}X_t=n\text{的时间}>2n^2] \le n^2/(2n^2) = 1/2$$

马尔科夫（Markov）不等式

- 令 X 为非负随机变量
- 且假设 $E[X]$ 存在
- 则对任意 $t > 0$，有 $Pr[X \geq t] \leq \frac{E[X]}{t}$

$$
\begin{aligned}
E(X) = \int_0^{\infty} x f(x) dx &= \int_0^{t} x f(x) dx + \int_t^{\infty} x f(x) dx \\
&\geq \int_t^{\infty} x f(x) dx \\
&\geq t \int_t^{\infty} f(x) dx \\
&= tP(X > t)
\end{aligned}
$$

2SAT 的随机游动算法：定理证明（续完）

用归纳法易证：对于所有j，

$$h_j \le n^2 - j^2 \le n^2$$

$$\mathrm{E}[X\text{从}X_0\text{到达}n\text{的时间}]$$

$$\le \mathrm{E}[Y\text{从}Y_0\text{到达}n\text{的时间}] \le n^2$$

由马尔科夫不等式，就有

$$\Pr[X\text{从}X_0\text{到达}n\text{的时间}>2n^2] \le 1/2$$

即若以$2n^2$步为一个阶段，则在一个阶段中

算法找不到满足赋值的出错概率不超过1/2.

当重复执行每个阶段m次，即总共执行$2mn^2$步时，

算法仍然找不到满足赋值的出错概率小于$1/2^m$.

有效随机算法的复杂性类

- 有效的拉斯维加斯型算法被称为ZPP类
  - 即零错误概率多项式时间随机算法
- 有效的蒙特卡洛型算法被称为BPP类
  - 即错误概率有界的多项式时间随机算法
  - 有效的弃真型单侧错误随机算法称为RP类
  - 有效的取伪型单侧错误随机算法称为coRP类
- 如果把ZPP类算法可以求解的判定问题类记作ZPP类，类似可以定义BPP、RP、co-RP等判定问题类，则

$$P \subseteq ZPP = RP \cap coRP \subseteq BPP$$

有效随机算法的局限性

- **Karp-Lipton定理：**

  若$\mathbf{NP \subseteq P/poly}$，则 $\mathbf{PH = \sum_2^P}$

  - **P/poly表示多项式规模电路能够求解的问题类**
  - **PH表示复杂性类从低到高的层次结构，也叫做多项式谱系，$\mathbf{\Sigma_2^P}$是其中的第2层**

- **不太可能为NP完全问题设计出多项式时间的随机算法**
  - **已知 $\mathbf{BPP \subseteq P/poly}$**
  - **如果 $\mathbf{NP \subseteq BPP}$，则 PH“塌方”到第2层**
  - **研究计算复杂性的人一般认为PH是不塌方的**

本讲小结

- 随机近似算法
  - MAX-3SAT 可满足
- 竞争解决随机算法的分析
- 拉斯维加斯型随机算法
  - 随机快速排序
- 蒙特卡洛型随机算法
  - RP 算法例子：2SAT随机游动
  - co-RP 算法例子：矩阵乘法检验
- 基础知识
  - Union Bound
  - Markov 不等式