智能优化方法及其应用

第三讲 伪随机数与蒙特卡洛方法

内容

- 伪随机数生成方法
- 蒙特卡洛方法
- 蒙特卡洛树搜索
- 经验分享

伪随机数生成方法

- **随机现象**是自然过程或人工过程中由多种未知因素共同作用产生的一种只可分析其统计规律却不能预测其发生的不确定现象. 这种现象表现为一系列没有规则的数值时就成为**随机数**.

- 所谓随机数, 其实是一个特殊的数列, 该数列中的每项以同等的概率选取, 这种选取不依赖于数列中的其他项. 因此, 说一个具体的数(如50)是随机的是没有意义的, 尽管它可以是某个随机数序列中的某一项.

伪随机数生成方法

- 自然界中的很多现象,例如抛掷硬币、抛掷骰子、转轮、洗牌等都展示出随机性,我们可以利用这些现象产生一些短周期的随机数.

- 利用计算机中的某些事件,如定时中断或时钟等也可以产生随机数,但这种机械方法由于计算机硬件故障经常会使随机数偏斜,并且这些数没法重复产生.

伪随机数生成方法

- 1946年，冯.诺依曼首次给出了使用计算机程序产生随机数的方法，但事实证明这种方法产生的数也并非是随机的.一个普遍的观点是，绝对随机的随机数只是一种理想的随机数，计算机不会产生绝对随机的随机数，它只能生成相对随机的随机数，即**伪随机数**. 因此，伪随机数并不是假随机数，这里的“伪”是有规律的意思，就是产生的伪随机数既是随机的又是有规律的.

- 这样产生的数列虽然不是由真实的随机现象产生的，但具有类似于随机数的统计性质，可以作为随机数来使用.

伪随机数生成方法

- 几乎每一种智能优化算法都要用到伪随机数:
  - ✓ **遗传算法:** 随机产生初始种群, 轮转法选择个体, 随机 选择交叉点, 随机选择变异的基因
  - ✓ **禁忌搜索算法:** 随机选择初始解和多阶段禁忌搜索初始解
  - ✓ **模拟退火算法:** 随机选择邻域解, 按概率作转移决策
  - ✓ **蚁群算法:** 随机产生初始蚁群, 按概率选择路径
  - ✓ **粒子群优化算法:** 随机初始化, 移动方向的随机加权组合
  - ✓ **捕食搜索算法:** 随机初始化, 在限制区域内的随机搜索
  - ✓ **人工神经网络算法:** 随机初始化, 随机梯度下降
  - ✓ 动态进化计算、算法性能测试……

伪随机数生成方法

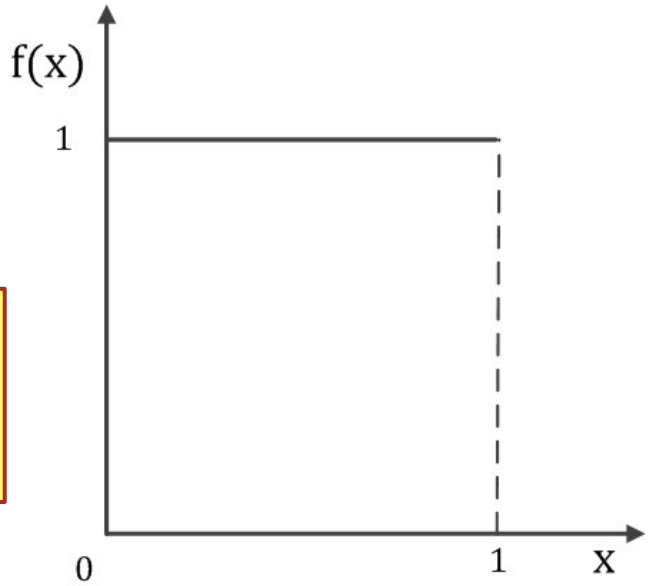
- 已有通用伪随机数生成程序不足之处:
  - ✓ 随机序列的长度和随机数的字长固定,而自编程序可以自由控制
  - ✓ 在重复计算中,不如自编的程序容易控制
  - ✓ 除了正态分布,一般没有产生其他分布的现成程序

- 掌握使用计算机产生随机数的原理和方法十分必要,在实际应用中可以得心应手.

0-1均匀分布伪随机数生成方法

- 0-1均匀分布的伪随机数是最基本也是最简单的伪随机数，它是生成一切其他分布的伪随机数的基础
- 设X是0-1均匀分布的随机变量，x是X的一个取值，记为$x \in U(0,1)$. 其中U表示均匀分布，0,1表示分布的区间. X的**概率密度函数**为

$$f(x)=\begin{cases}1, & 0 \le x < 1\\ 0, & \text{其他}\end{cases}$$

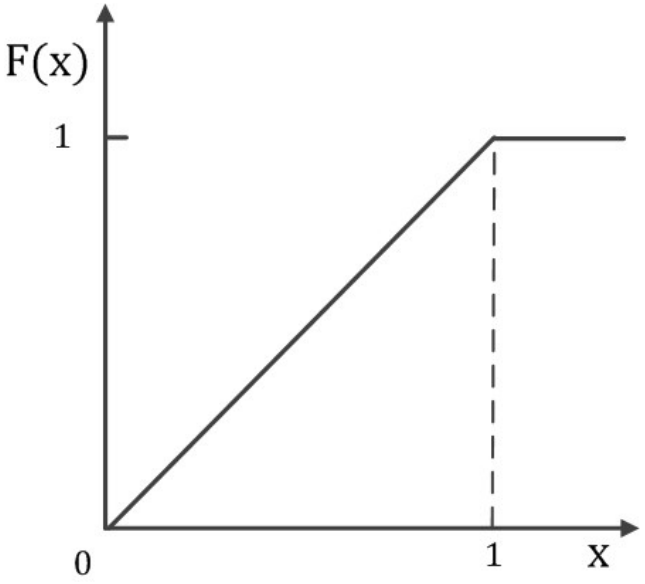


连续型随机变量的**概率密度函数**是一个描述这个随机变量的输出值，在某个确定的取值点附近的可能性的函数.

0-1均匀分布伪随机数生成方法

- 0-1均匀分布的伪随机数是最基本也是最简单的伪随机数，它是生成一切其他分布的伪随机数的基础
- 设X是0-1均匀分布的随机变量，x是X的一个取值，记为$x \in U(0,1)$. 其中U表示均匀分布，0,1表示分布的区间. X的**累积分布函数**为

$$F(x)=\begin{cases}0, & x<0\\ x, & 0\le x<1\\ 1, & x\ge 1\end{cases}$$



0-1均匀分布伪随机数生成方法

- 容易得到，$U(0,1)$的数学期望μ和标准差σ分别为

$$\mu = \frac{1}{2}, \sigma = \frac{1}{2\sqrt{3}}$$

  - ✓ **数学期望:**反映随机变量平均取值的大小,试验中每次可能结果的概率乘以其结果的总和
  - ✓ **标准差:**反映组内个体间的离散程度，所有数减去其平均值的平方和，所得结果除以该组数之个数，再把所得值开根号，所得之数就是这组数据的标准差。

- 0-1均匀分布的伪随机数通常是由均匀分布的伪随机数整数除以数列长度(周期)获得.

0-1均匀分布伪随机数生成方法

- 伪随机数生成方法好坏的判断标准:
  ① 产生的数列具有很好的随机性和均匀性
  ② 不出现重复的数列的长度(周期)要尽可能长
  ③ 产生伪随机数的计算速度要快
  ④ 计算程序占用的计算机内存要尽可能少

- **定义1** 设$\{x_i\}$是一个整数序列, 如果T是满足下列条件的最小正整数: 存在整数i_0使得所有$i \geq i_0$, 都有$x_{i+T} = x_i$, 则称T为$\{x_i\}$的**周期**.

0-1均匀分布伪随机数生成方法

- 产生均匀分布随机数的方法主要有:

① **平方取中法:** 将一个$2n$位的二进制随机数取平方后, 取中间的$2n$位作为新的随机数, 迭代公式为 $B_{k+1} = [B_k \times B_k]_{2n}$, 其中$B_k$为第$k$次迭代得到的二进制随机数, $[\cdot]_{2n}$表示取中间$2n$位数字

② **倍积取中法:** 用一个整数常数A来乘一个n位的随机数, 取中间n位作为新的随机数, 迭代公式为 $S_{k+1} = [A \times S_k]_{2n}$

③ **乘同余法:** $S_{k+1} = AS_k \ mod(M)$

④ **线性同余法:** $S_{k+1} = (AS_k + C) \ mod(M)$

0-1均匀分布伪随机数生成方法

- **乘同余法**基本思想: 用一个整数常数A来乘一个随机数, 将得到的积除以模M的余数作为新的随机数. 迭代公式如下:

$$S_{k+1} = AS_k \ mod(M)$$

其中S_k表示第k次迭代得到的随机数, $mod(\cdot)$表示取模运算.

- 显然, 我们总是希望随机数序列的周期越长越好
  - ✓ 周期如何计算?
  - ✓ 如何让周期尽量长?

0-1均匀分布伪随机数生成方法

- **乘同余法:** $S_{k+1} = AS_k \, mod(M)$
- 根据数论的理论可以证明: 位数为L的计算机, 如果取模数$M = 2^L$, 当下面两个条件满足: 1) $A = 8k \pm 3$且$A = 4k + 1$, k为正整数; 2) S_0为奇数.乘同余法获得的随机数序列周期最长, 为2^{L-2}

- **例1** 若L为6, 则$M = 64$, 取$A = 13$, $S_0 = 1$, 可获得一个周期为$2^4 = 16$的随机数序列:
  $\{1, 13, 41, 21, 17, 29, 57, 37, 33, 45, 9, 53, 49, 61, 25, 5, 1, \cdots\}$

0-1均匀分布伪随机数生成方法

- **线性同余法**基本思想: 用一个整数常数A来乘一个随机数再加上一个整数常数C, 将得到的结果除以模M的余数作为新的随机数. 迭代公式如下:

$$S_{k+1} = (AS_k + C)\ mod(M)$$

其中S_k表示第k次迭代得到的随机数, $mod(\cdot)$表示取模运算.

- 显然, 当$C = 0$时, 线性同余法退化为乘同余法.
- 在同等位数的计算机上, 线性同余法能够产生比乘同余法生成的周期更长的随机 数序列

0-1均匀分布伪随机数生成方法

- **线性同余法:** $S_{k+1} = (AS_k + C)\ mod(M)$
- 根据数论的理论可以证明: 位数为L的计算机, 如果取模数$M = 2^L$, 当下面两个条件满足: 1) $A = 4k + 1, k$为正整数; 2) C与M互素. 线性同余法获得的随机数序列周期最长, 为2^L

- **例2** 若L为5, 则$M = 32$, 取$A = 13, C = 5, S_0 = 1$, 可获得一个周期为$2^5 = 32$的随机数序列:

$\{\mathbf{1}, 18, 15, 8, 13, 14, 27, 4, 25, 10, 7, 0, 5, 6, 19, 28, 17, 2,$
$31, 24, 29, 30, 11, 20, 9, 26, 23, 16, 21, 22, 3, 12, \mathbf{1}, \cdots\}$

0-1均匀分布伪随机数生成方法

- **线性同余法:** $S_{k+1} = (AS_k + C)\ mod(M)$
- **乘同余法:** $S_{k+1} = AS_k\ mod(M)$
- 位数为L的计算机
  - ✓ **线性同余法最大周期:** 2^L
  - ✓ **乘同余法最大周期:** 2^{L-2}

- 虽然线性同余法比乘同余法要多做一次加法,但是在同一计算机上生成的随机数序列的周期是乘同余法的4倍!

伪随机数生成方法

- 从**初等数论**角度介绍两种伪随机数生成方法.

- 第一种方法是莱梅于1949年发明的**线性同余方法**.

  设整数n, a, c, x_0满足下列条件:

  $2 \le a < n,\ 0 \le c < n,\ 0 \le x_0 \le n,$

  那么伪随机数序列由下面的公式给出:

  $x_{i+1} = (ax_i + c)\ mod\ n.$

  这里n, a, c, x_0分别被称为**模数**, **倍数**, **增量**和**种子**.

  上面的公式也称作**线性同余生成器**,

  它给出0到$n-1$的整数序列$x_0, x_1, \cdots, x_i, \cdots$.

  如果我们想生成$[0,1]$内的伪随机数, 那么只需用n去除序列$x_0, x_1, \cdots, x_i, \cdots$中的每项即可.

  如果把这个序列中的每项都模2, 则得到一个0, 1伪随机数序列.

伪随机数生成方法

- 容易从$x_{i+1} = (ax_i + c)\ mod\ n$中得到$\{x_i\}$的通项:

$$x_i = \left(a^i x_0 + \frac{a^i - 1}{a - 1} \cdot c\right)\ mod\ n$$

作为一个练习, 同学们可使用数学归纳法证明上式的确是$\{x_i\}$的通项.

- **例3** 设$n = 9,\ a = 7, c = 4, x_0 = 3,$
- 则由线性同余生成器可产生序列$3, 7, 8, 6, 1, 2, 0, 4, 5, 3, \cdots$.

显然, 这个序列在重复出现$3, 7, 8, 6, 1, 2, 0, 4, 5$之前有$9(n = 9)$个不同的数.

伪随机数生成方法

- **例4** 设$n=12, a=3, c=4, x_0=5$, 则由线性同余生成器计算有

$$x_1 = (3\times 5+4)\ mod\ 12 = 7,$$
$$x_2 = (3\times 7+4)\ mod\ 12 = 1,$$
$$x_3 = (3\times 1+4)\ mod\ 12 = 7,$$
$$\vdots$$

因此该线性同余生成器产生序列$5, 7, 1, 7, 1, 7, 1, \cdots$.

值得注意的是, 这个序列在重复出现$7, 1$之前仅有$3(<n)$个不同的数.

伪随机数生成方法

- 在上面两个例子中可以发现，线性同余生成器产生的伪随机数具有周期性.

- 这是因为$x_i \in \mathbb{Z}_n$仅有n个可能的取值，于是必然存在整数$j > i$，使得$x_j = x_i$，从而由线性同余生成器$[\boldsymbol{x_{i+1} = (ax_i + c) \bmod n}]$计算知，对任意正整数$k$都有$x_{j+k} = x_{i+k}$，所以序列在至多$n$个项后必定重复.

伪随机数生成方法

- **定理1** 设线性同余生成器$x_{i+1} = (ax_i + c)\ mod\ n$产生的伪随机数序列的周期为$T$, 则$T = n$当且仅当下述条件成立:

  (1)$(c, n) = 1$.

  (2) $a \equiv 1(mod\ p)$, **这里p跑遍n的所有素因数.**

  (3) 当$4|n$时, $a \equiv 1\ (mod\ 4)$.

- 上述定理的证明相当复杂, 有兴趣的同学可参见下面的参考文献.

  [D. Knuth(高德纳). The Art of Computer Programming, vol. 2: Seminumerical Algorithms (3rd Edition). Reading, MA: Addison-Wesley. 1997]

- **线性同余法:** 位数为L的计算机, 如果取模数$M = 2^L$, 当下面两个条件满足: 1) $A = 4k + 1$, k为正整数; 2) C与M互素, 线性同余法获得的随机数序列周期最长, 为2^L

伪随机数生成方法

- 尽管线性同余生成器被广泛用于产生伪随机数，但它也存在一些不足. 例如，当已知该伪随机数序列的一些项后，我们有可能计算出$x_{i+1} = (ax_i + c) \bmod n$中的一些参数$a, c$和$n$，于是可以得到该序列的所有项. 如果这样的伪随机数应用于密码协议中，必然影响密码体制的安全.

- 为了增强伪随机数的安全性，1986年L. Blum等人发明了**平方伪随机数生成器**，该生成器产生的序列$x_0, x_1 \ldots, x_i, \ldots$具有这样的性质：从任意项$x_i$在合理的时间内无法计算出$x_{i-1}$. 这里，从$x_i$计算$x_{i-1}$的困难性是基于大整数分解的困难性假设的.

伪随机数生成方法

- **平方伪随机数生成器**[了解]

  设n是正整数, $x_0 \in \mathbb{Z}$, 那么**平方伪随机数生成器**由下面的公式给出: $x_{i+1} = {x_i}^2 \ mod \ n$,

  也即: $x_i = {x_0}^{2^i} \ mod \ n$,

  这里n和x_0分别是**模数**和**种子**.

- **例5** 设$n = 209, x_0 = 6$, 那么平方伪随机数生成器产生下面的序列:

  $6, 36, 42, 92, 104, 157, 196, 169, 137, 168, 9, 81, 82, 36, 42, \ldots,$

  它的周期为$T = 12$.

伪随机数生成方法

- **定理2** 设$ord_n x_0 = 2^t s$, 其中s是奇数, t是非负整数, 则平方伪随机数生成器$x_{i+1} = {x_i}^2 \ mod \ n$生成的伪随机数序列的周期是$ord_s 2$. [了解]

- 例如, 在例5中$n = 209, x_0 = 6$, 计算知$ord_n x_0 = ord_{209} 6 = 90 = 2 \times 45$, 所以平方伪随机数生成器用模数209和种子6生成的伪随机数序列的周期为$T = ord_s 2 = ord_{45} 2 = 12$, 这与我们在例5中观察到的一致.

0-1均匀分布伪随机数生成方法

- **线性同余法:** $S_{k+1} = (AS_k + C)\ mod(M)$
- **乘同余法:** $S_{k+1} = AS_k\ mod(M)$

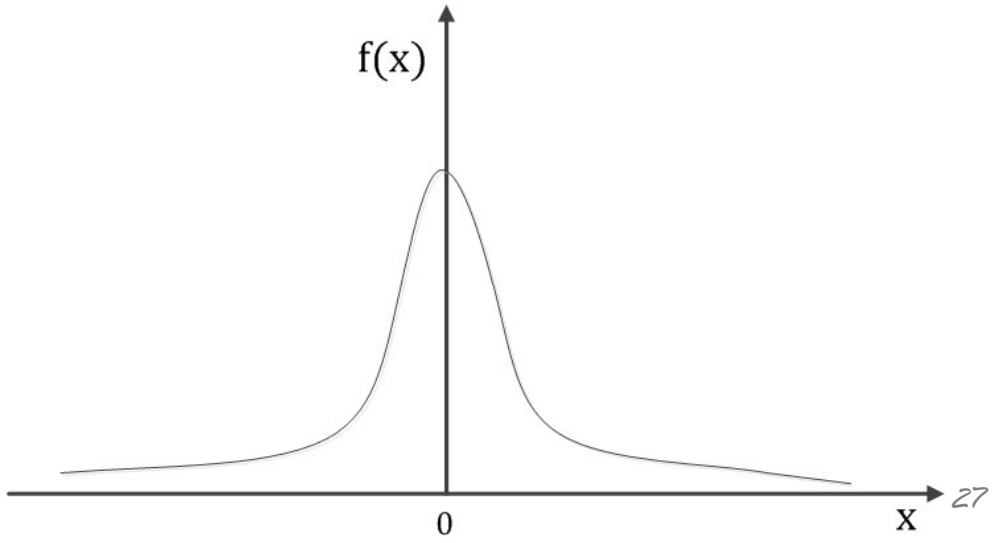
- 获得随机数S_i后，$0-1$均匀分布的随机数$x_i \in U(0,1)$就可以用除以模的方式获得: $x_i = S_i/M$
- $[a,b]$之间均匀分布的随机数$y_i \in U(a,b)$则可以由下式获得: $y_i = a + (b-a)x_i$
- 智能优化算法中经常要产生1和n之间的整数K，也可由$0-1$均匀分布的随机数生成:

$$K = 1 + floor[xn], \qquad x \in U(0,1)$$

正态分布伪随机数生成方法

- 正态分布也称高斯分布，是最常见的随机分布，在智能优化方法中也经常用到
- 若X服从参数为μ和σ的正态分布，即$X \sim N(\mu, \sigma^2)$或$x \in N(\mu, \sigma^2)$. 其中，μ和σ分别为X的期望值和标准差. 当$\mu = 0$和$\sigma = 1$时，称为$0-1$正态分布，即$N(0,1)$. 其概率密度函数为

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$



正态分布伪随机数生成方法

- 其他正态分布都可以用0-1正态分布$N(0,1)$生成

- 由概率论知，若$Y_1, Y_2, \cdots, Y_N$是n个独立同分布的随机变量，当n足够大时

$$X = Y_1 + Y_2 + \cdots + Y_n$$

就近似于一个正态分布. 一般来说，当$n > 6$时，X就是一个正态分布的很好的近似. X的均值和方差分别为

$$\mu_x = \mu_1 + \mu_2 + \cdots + \mu_n = n\mu$$
$$\sigma_x^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2 = n\sigma^2$$

正态分布伪随机数生成方法

- $Y_1, Y_2, \cdots, Y_N$独立同分布

$$X = Y_1 + Y_2 + \cdots + Y_n$$

$$\mu_x = n\mu, \qquad \sigma_x^2 = n\sigma^2$$

- 于是有$X \sim N(n\mu, n\sigma^2)$或$x \in N(n\mu, n\sigma^2)$, 令

$$z = \frac{x - \mu_x}{\sigma_x}$$

可将其转变为$N(0,1)$, 也即$z \in N(0,1)$. 显然有

$$z = \frac{\sum_{i=1}^{n} y_i - n\mu}{\sqrt{n\sigma^2}}$$

正态分布伪随机数生成方法

- $Y_1, Y_2, \cdots, Y_N$独立同分布

$$X = Y_1 + Y_2 + \cdots + Y_n$$

$$z = \frac{\sum_{i=1}^{n} y_i - n\mu}{\sqrt{n\sigma^2}}, \qquad z \in N(0,1)$$

- 令Y_i为$0-1$均匀分布$U(0,1)$, 已知$\mu = \frac{1}{2}, \sigma = \frac{1}{2\sqrt{3}}$, 代入

$$z = \frac{\sum_{i=1}^{n} y_i - n/2}{\sqrt{n/12}}$$

取$n = 12$, 得

$$z = \sum_{i=1}^{12} y_i - 6$$

正态分布伪随机数生成方法

- **正态分布伪随机数生成方法总结**

产生12个$0-1$均匀分布的伪随机数$y_i \in U(0,1)$

计算

$$z = \sum_{i=1}^{12} y_i - 6$$

得到z是服从$0-1$正态分布的随机数, $z \in N(0,1)$

给定数学期望μ和标准差σ, 计算

$$x = \mu + \sigma z$$

便可得到服从$N(\mu, \sigma^2)$正态分布的随机数x

其他分布伪随机数生成方法

- 除了均匀分布和正态分布之外，智能优化算法中还可能用到其他分布函数的伪随机数. 这里介绍产生其他**已知分布函数**的随机数的**逆变法**

① 由给定密度函数$f(x)$计算分布函数$F(x)$，或$F(x)$已知

② 推导$F(x)$的逆函数$F^{-1}(y)$ $(y = F(x), x = F^{-1}(y))$

③ 产生$0-1$均匀分布的随机数序列$\{y_i\}, y \in U(0,1)$

④ 由公式$x = F^{-1}(y)$计算得到一个分布为$F(x)$的随机数序列$\{x_i\}$

其他分布伪随机数生成方法

- **逆变法**的推导
- X是一个已知分布函数为$F(x)$的随机变量

$Y = F(X)$是一个关于X的, 以X的分布函数$F(\cdot)$为函数的随机变量, 对于X的一个样本值x, 有$y = F(x)$是Y的一个样本值. 记$F^{-1}(\cdot)$是$F(\cdot)$的逆函数, 有$x = F^{-1}(y)$

按照分布函数的定义, Y的分布函数为

$$P(Y < y) = P(F(X) < y) = P\left(X < F^{-1}(y)\right)$$
$$= P(X < x) = F(x) = y$$

若Y的定义域为$[0,1]$, 则Y服从$0-1$均匀分布

因此只要产生一个$y \in U(0,1)$, 就可以使用分布函数的逆函数$F^{-1}(\cdot)$生成分布函数为$F(\cdot)$的随机数.

其他分布伪随机数生成方法

- **例6** 产生负指数分布的随机数x

负指数分布的密度函数为$f(x)=\lambda e^{-\lambda x},(x\geq 0)$

可得分布函数

$$F(x)=\int_0^x \lambda e^{-\lambda t}dt=1-e^{-\lambda x},\qquad x\geq 0$$

令$y=1-e^{-\lambda x}$, 求逆函数得$x=-\frac{1}{\lambda}\ln(1-y)$

y是$0-1$均匀分布时, $u=1-y$也是$0-1$均匀分布.

因此, 生成$u\in U(0,1)$, 计算$x=-\frac{1}{\lambda}\ln u$

可得负指数分布的随机数x

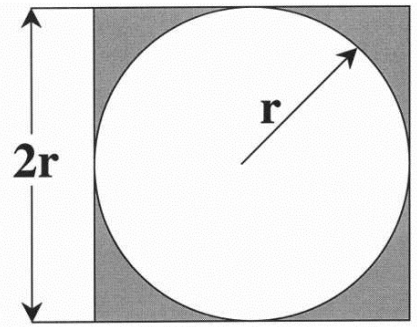
蒙特卡洛方法

- 蒙特卡洛方法是一类依靠重复随机采样来获得数值结果的计算方法的统称. 它们的核心思想是利用随机性来求解原则上可能是确定性的问题.

- 理论上, 蒙特卡洛方法可以用于求解任何在统计意义下可解释的问题. 蒙特卡洛方法的理论基础是大数定律. 大数定律(law of large numbers)是一种描述当试验次数很大时所呈现的概率性质的定律. 大数定律通俗一点来讲, 就是样本数量很大的时候, 样本均值和真实均值充分接近.

蒙特卡洛方法

- 蒙特卡洛方法通常包括以下几个步骤:
  ① 定义输入数据的范围
  ② 在指定范围内生成满足一定概率分布的随机输入数据
  ③ 对输入数据进行确定性计算
  ④ 合成计算得出结果



- **实例1** 用蒙特卡洛方法计算π:
  ① 给定一个2×2正方形, 画出其内切圆
  ② 在正方形内随机均匀绘制散点
  ③ 统计正方形内的散点总数N_S和内切圆内的散点数量N_C
  ④ 计算$\frac{N_C}{N_S}$, 由$\frac{N_C}{N_S} = \frac{\pi}{4}$, 得$\pi = 4\frac{N_C}{N_S}$

蒙特卡洛方法

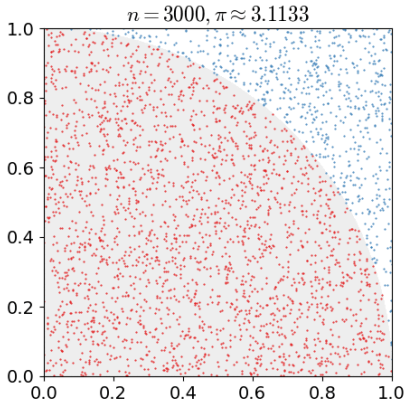
- **实例1** 用蒙特卡洛方法计算π:

① 给定一个2×2正方形, 画出其内切圆

② 在正方形内随机均匀绘制散点

③ 统计正方形内的散点总数N_S和内切圆内的散点数量N_C

④ 计算$\frac{N_C}{N_S}$, 由$\frac{N_C}{N_S} = \frac{\pi}{4}$, 得$\pi = 4\frac{N_C}{N_S}$



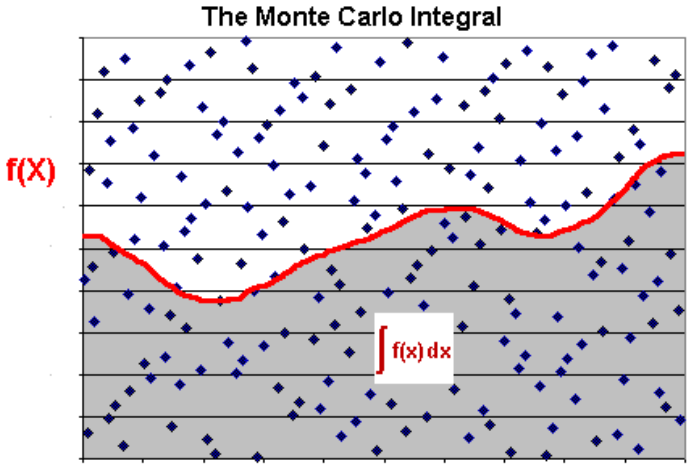
- **要点:**

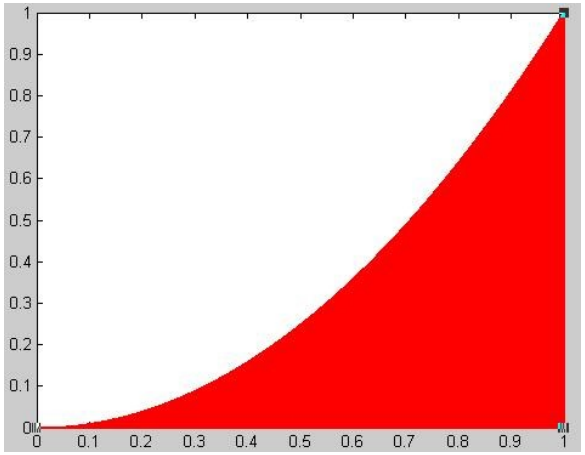
① 散点必须是在正方形内随机均匀分布

② 散点数量必须足够多, 若只有少量散点, 近似效果差; 散点数量越多, 近似效果越好

蒙特卡洛方法

- 用蒙特卡洛方法计算积分:





- **实例2** 用蒙特卡洛方法计算$\int_0^1 x^2 dx$

① 确定函数$y = x^2$的取值范围: $x \in [0,1], y \in [0,1]$

② 在上述正方形区域内随机均匀绘制散点

③ 统计正方形内的散点总数N_S和红色区域内的散点数量N_R

④ 计算$\frac{N_R}{N_S}$, 得$\int_0^1 x^2 dx = \frac{N_R}{N_S}$

蒙特卡洛方法

- 用蒙特卡洛方法模拟系统内部的随机运动:

- **实例3** 用蒙特卡洛方法模拟单车道的交通情况

  根据 Nagel-Schreckenberg 模型, 车辆的运动满足以下规则:

  - ✓ 当前速度是v
  - ✓ 如果前面没车, 它在下一秒的速度会提高到$v + 1$, 直到达到规定的最高限速
  - ✓ 如果前面有车, 距离为d, 且$d < v$, 那么它在下一秒的速度会降低到$d - 1$
  - ✓ 此外, 司机还会以概率p随机减速, 将下一秒的速度降低到$v - 1$

蒙特卡洛方法

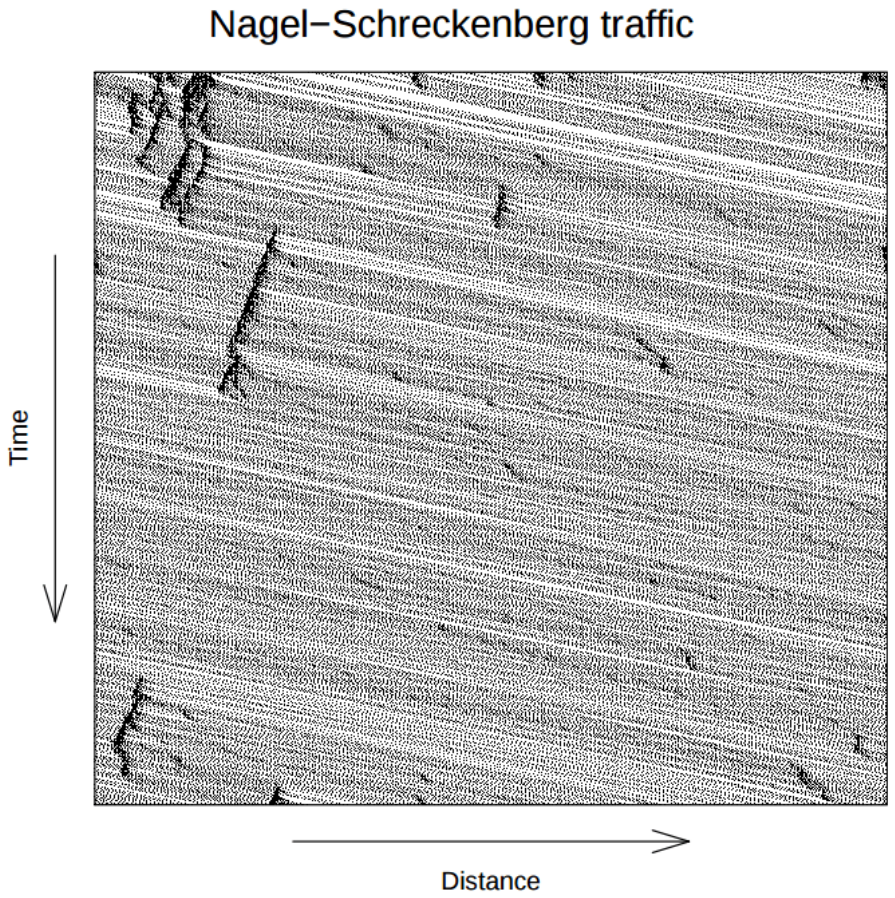
- 用蒙特卡洛方法模拟系统内部的随机运动:

- **实例3** 用蒙特卡洛方法模拟单车道的交通情况

  根据 Nagel-Schreckenberg 模型, 在一条直线上, 随机产生100个点, 代表道路上的100辆车, 另取概率p为 0.3

蒙特卡洛方法

- **实例3** 用蒙特卡洛方法模拟单车道的交通情况



右图中，横轴代表距离(从左到右)，纵轴代表时间(从上到下)，因此每一行就表示下一秒的道路情况.

可以看到，该模型会随机产生交通拥堵(图形上黑色聚集的部分). 这就证明了，**单车道即使没有任何原因，也会产生交通堵塞.**

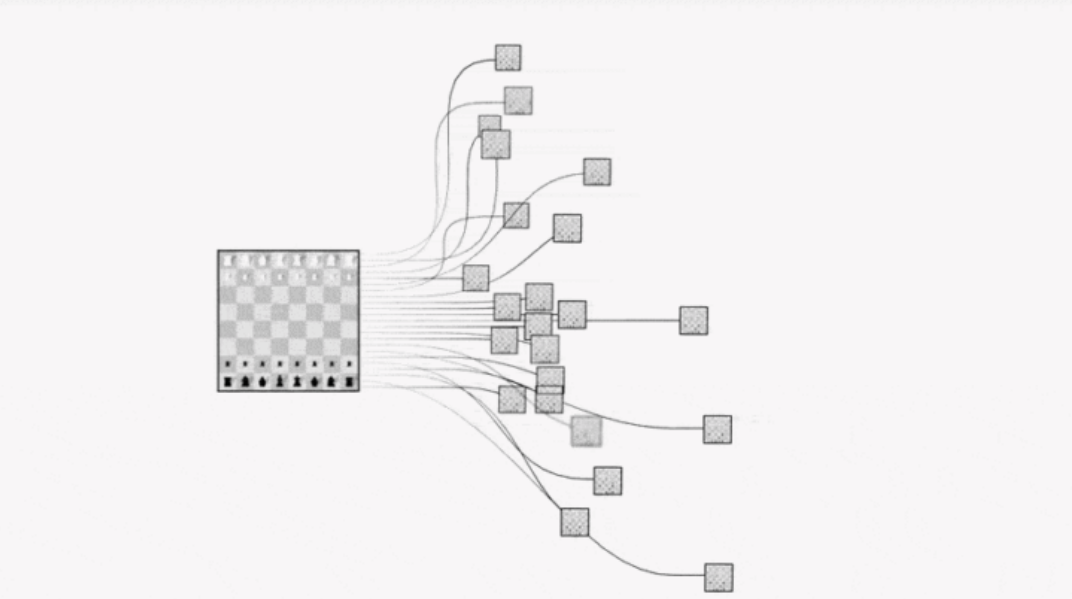
蒙特卡洛树搜索

- 蒙特卡洛树搜索全称 Monte Carlo Tree Search(MCTS), 是一种人工智能问题中做出最优决策的方法, 一般是在组合博弈中的行动规划形式. 它结合了随机模拟的一般性和树搜索的准确性.

- MCTS受到快速关注主要是由计算机围棋程序的成功以及其潜在的在众多难题上的应用所致. 超越博弈游戏本身, MCTS 理论上可以被用在以 {状态 state, 行动 action} 对定义和用模拟进行预测输出结果的任何领域.

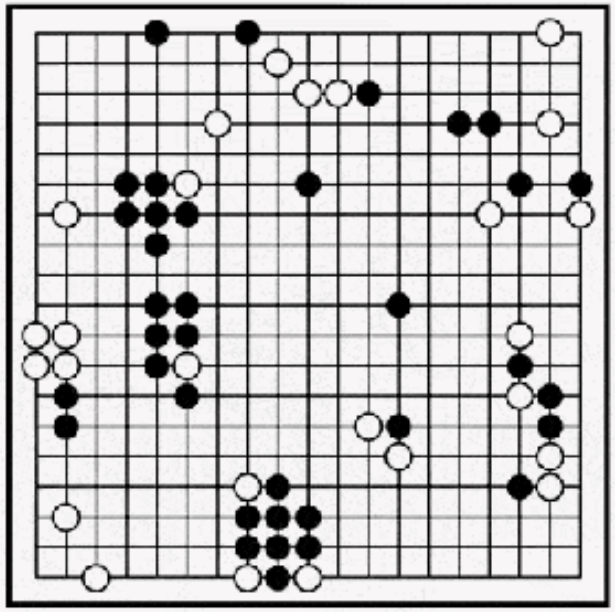
蒙特卡洛树搜索

- **人/机器下棋的基本决策过程**: 根据当前棋面状态, 确定下一步动作
- 那么, 该下哪步才能保证后续赢球的概率比较大?
- 最容易想到的就是枚举之后的每一种下法, 然后计算每步赢棋的概率, 选择概率最高的就好.

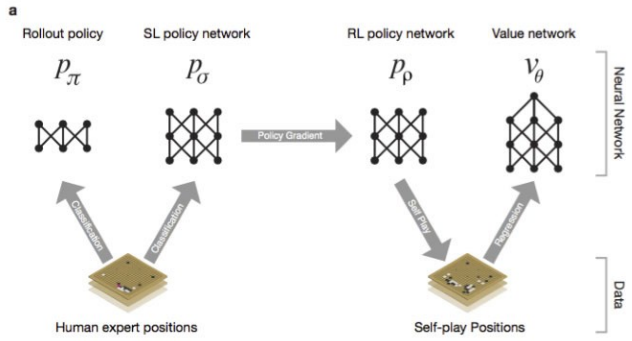


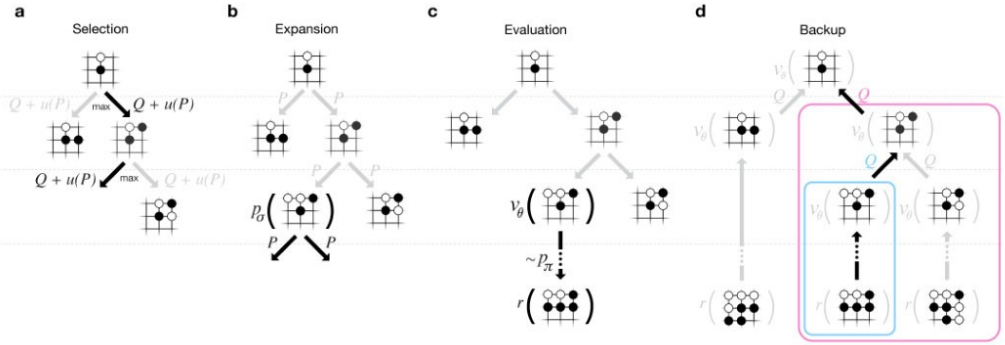
蒙特卡洛树搜索

- 中国象棋、国际象棋等都能暴力求解.
- 但是, 对于围棋而言, 状态空间实在太大, 无法枚举.
- AlphaGo整合深度学习和蒙特卡洛树搜索解决该问题



蒙特卡洛树搜索


a
Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy Gradient
Classification
Classification
Self Play
Regression
Neural Network
Data
Human expert positions
Self-play Positions
b
Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s'


a
Selection
b
Expansion
c
Evaluation
d
Backup
Q + u(P)
max
p_σ
v_θ
$\sim p_\pi$
r
P
Q


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 410
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card forty years on
PAGE 482
NATURE.COM/NATURE


ALPHAGO
0:00:00
LEE SEDOL
01:59:31
AlphaGo
Lee Sedol


Google
@Google黑板报

蒙特卡洛树搜索

- 基本的 MCTS 算法非常简单——根据模拟的输出结果，按照节点构造搜索树

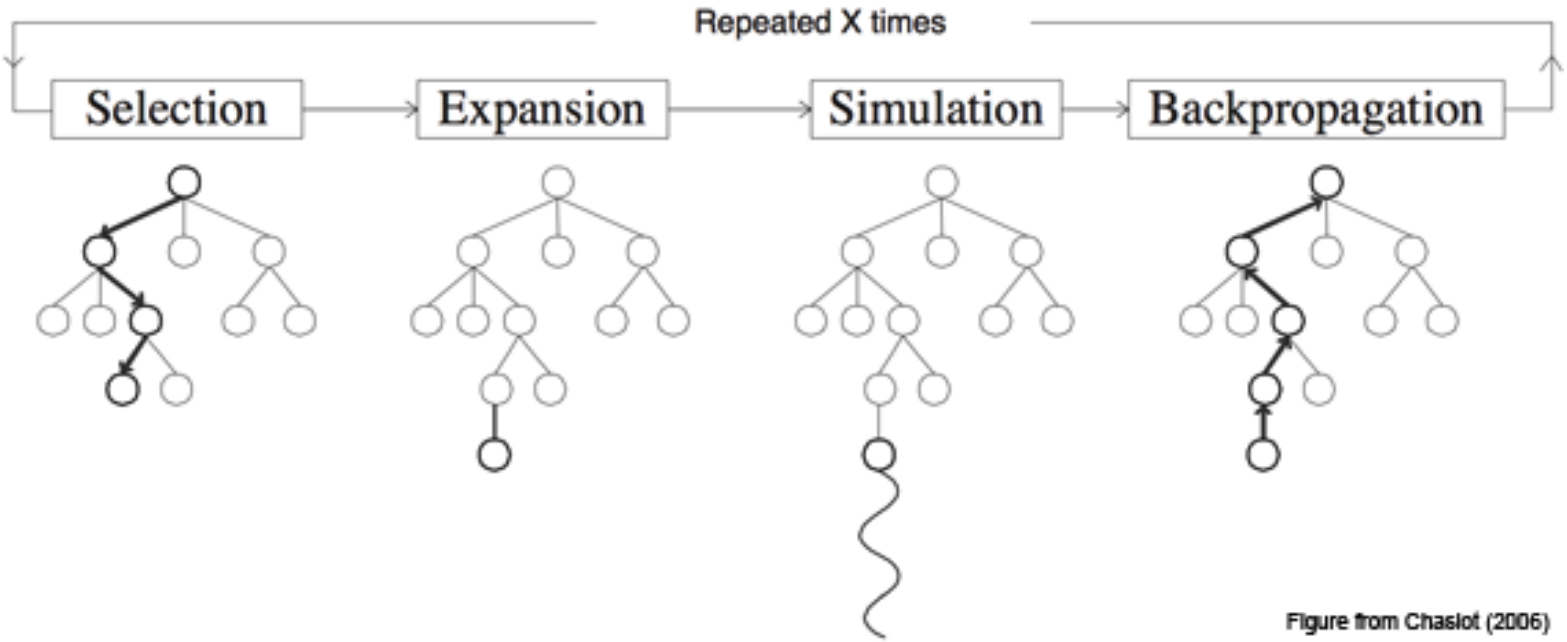
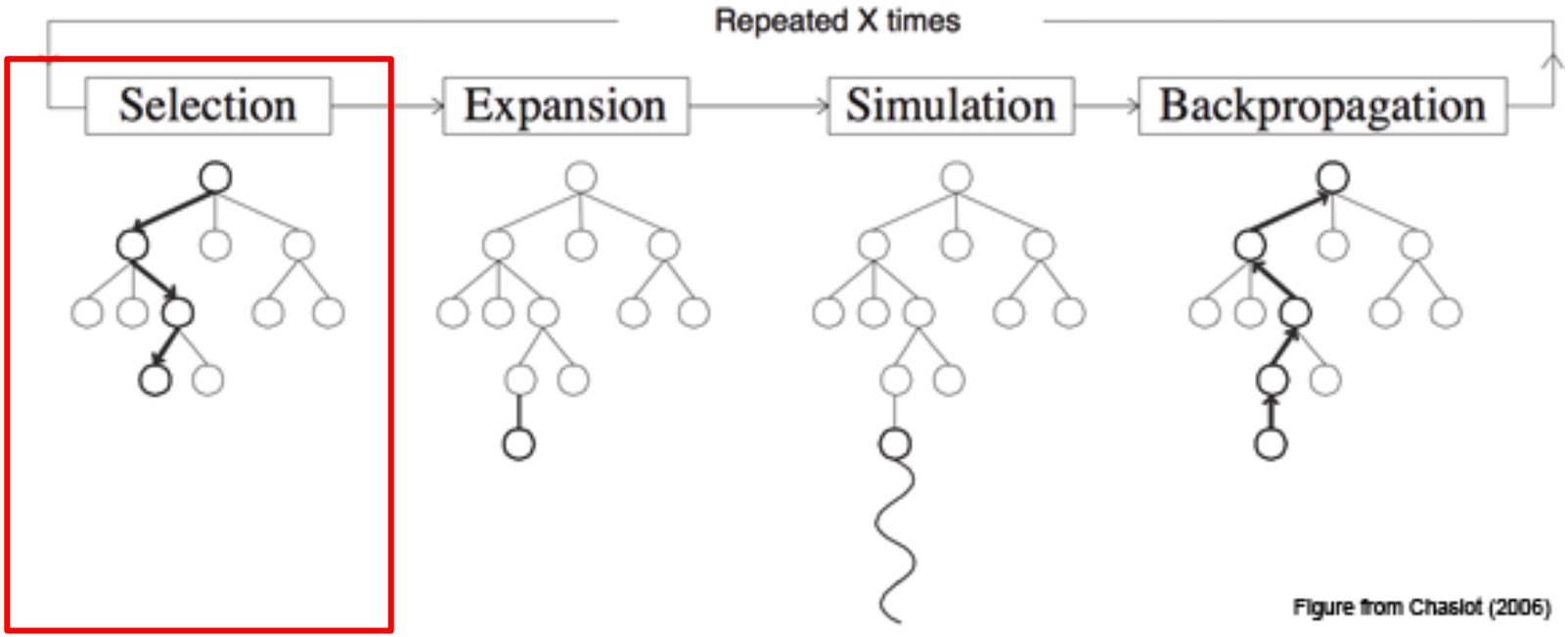


Figure from Chaslot (2006)

每个节点必须包含两个重要的信息：一个是根据模拟结果估计的值和该节点已经被访问的次数

蒙特卡洛树搜索

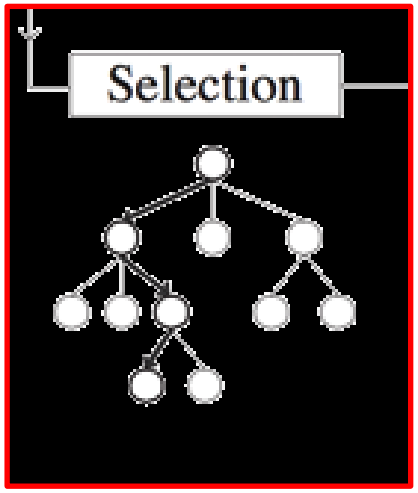
- 基本的蒙特卡洛树搜索(MCTS)算法



① **选择 Selection**: 从根节点 R 开始, 递归选择最优的子节点直到达到叶子节点 L

蒙特卡洛树搜索

- 基本的蒙特卡洛树搜索(MCTS)算法



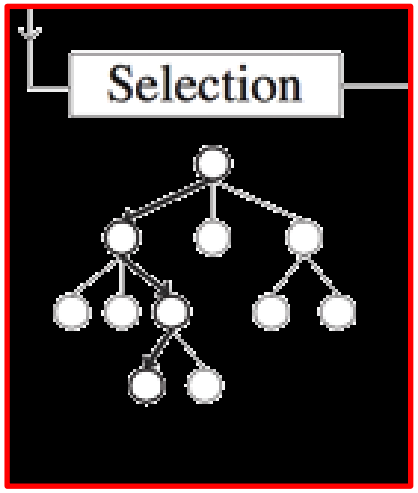
在树向下遍历时的节点选择操作是通过选择最大化某个量来实现, 这类似于求解多臂赌博机问题(Multiarmed bandit problem), 其中的参与者必须选择一个赌博机(bandit)来最大化每一轮的估计的收益. 我们可以使用 Upper Confidence Bounds(UCB)公式来计算这个量:

$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

其中, v_i是节点估计的值, n_i是节点被访问的次数, N是其父节点已经被访问的总次数, C是可调参数

蒙特卡洛树搜索

- 基本的蒙特卡洛树搜索(MCTS)算法



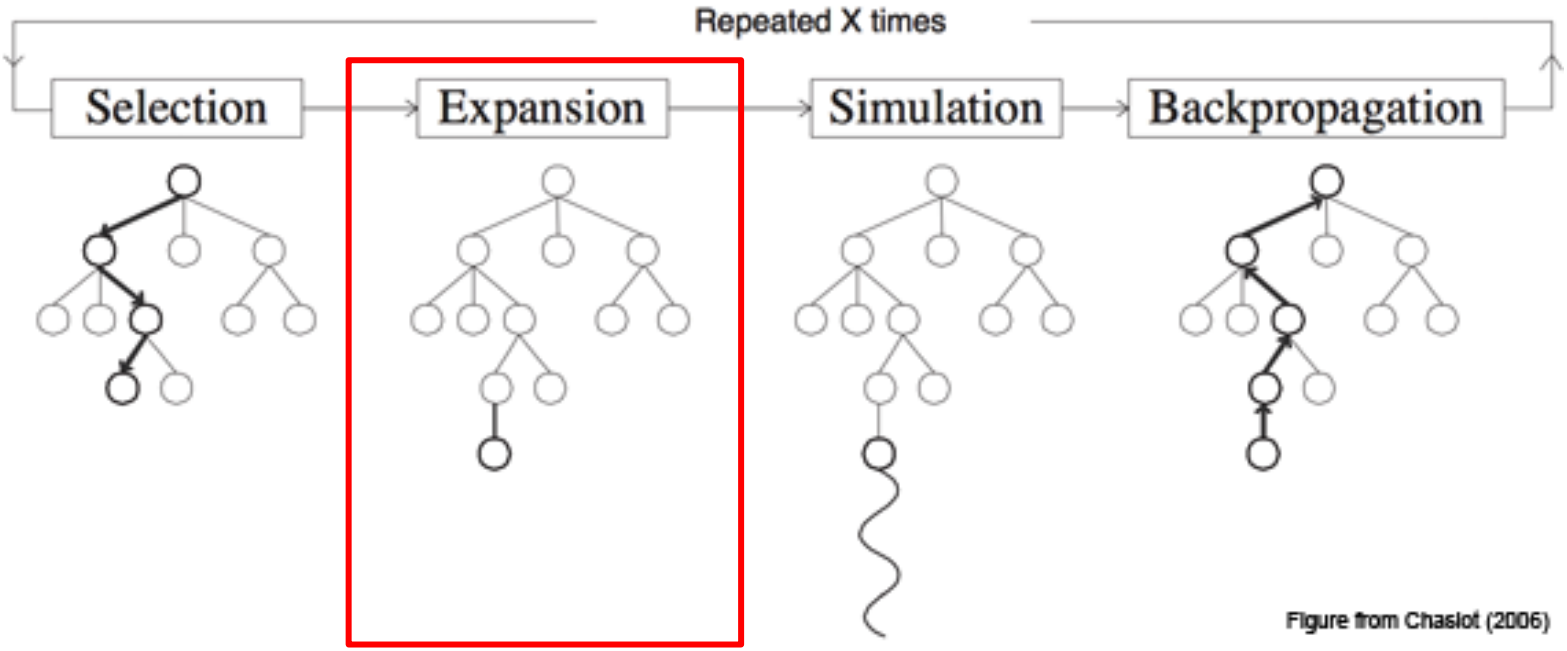
- **利用**和**探索**的平衡:

  UCB 公式对已知收益的**利用**和鼓励接触那些相对未曾访问的节点的**探索**之间进行平衡. 收益估计基于随机模拟, 所以节点必须被访问若干次来确保估计变得更加可信; MCTS 估计会在搜索的开始不大可靠, 而最终会在给定充分的时间后收敛到更加可靠的估计上, 在无限时间下能够达到最优估计.

$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$其中, v_i是节点估计的值, n_i是节点被访问的次数, N是其父节点已经被访问的总次数, C是可调参数

蒙特卡洛树搜索

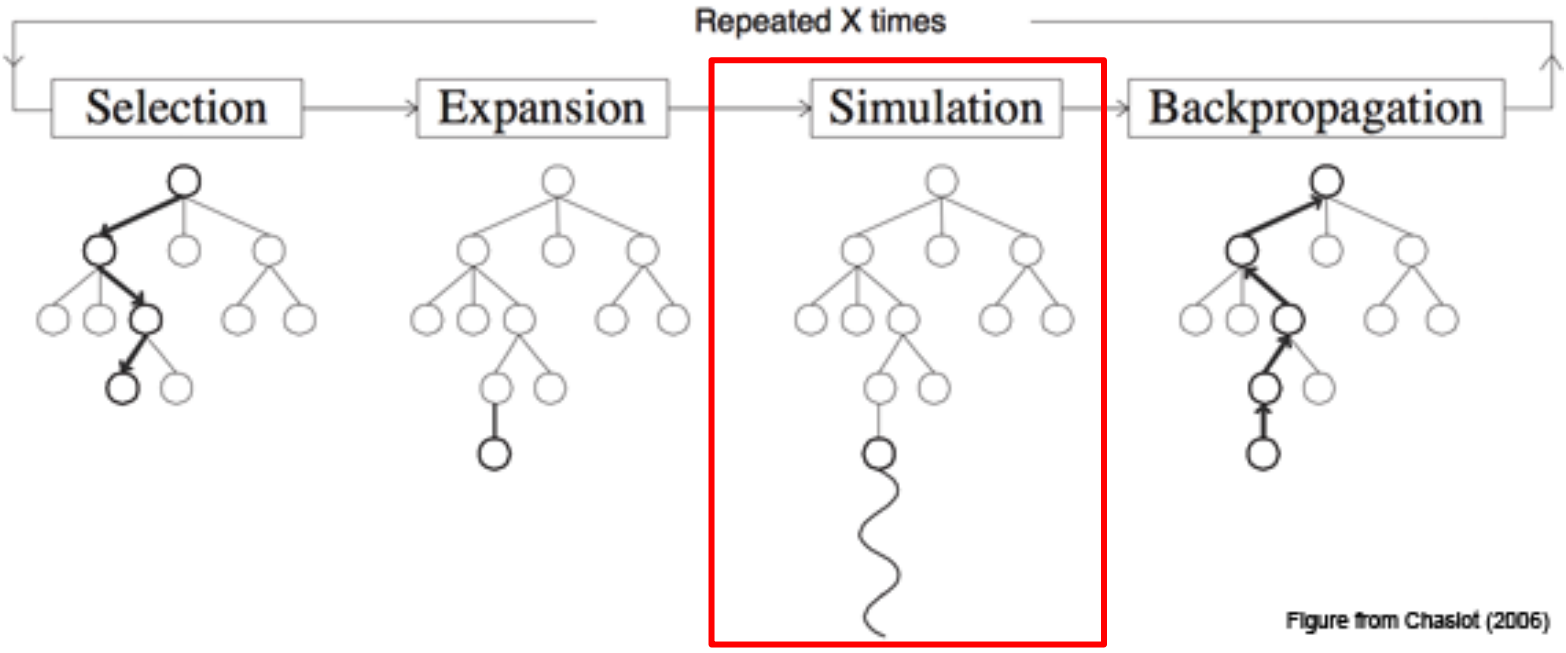
- 基本的蒙特卡洛树搜索(MCTS)算法



② **扩展 Expansion:** 如果 L 不是一个终止节点(也即, 不会导致博弈游戏终止) 那么就创建一个或者更多的子节点, 选择其中一个 C

蒙特卡洛树搜索

- 基本的蒙特卡洛树搜索(MCTS)算法



③ **模拟 Simulation**: 从 C 开始运行一个模拟的输出, 直到博弈游戏结束

蒙特卡洛树搜索

- 基本的蒙特卡洛树搜索(MCTS)算法

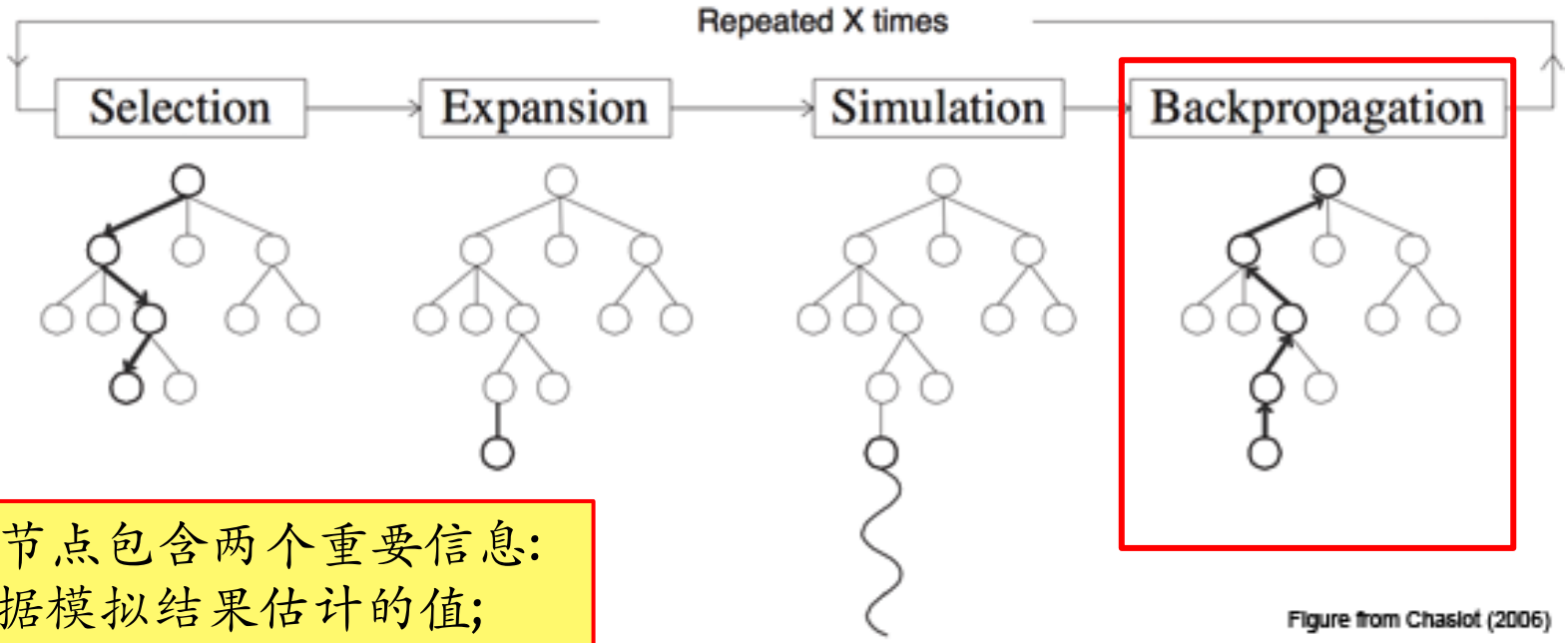


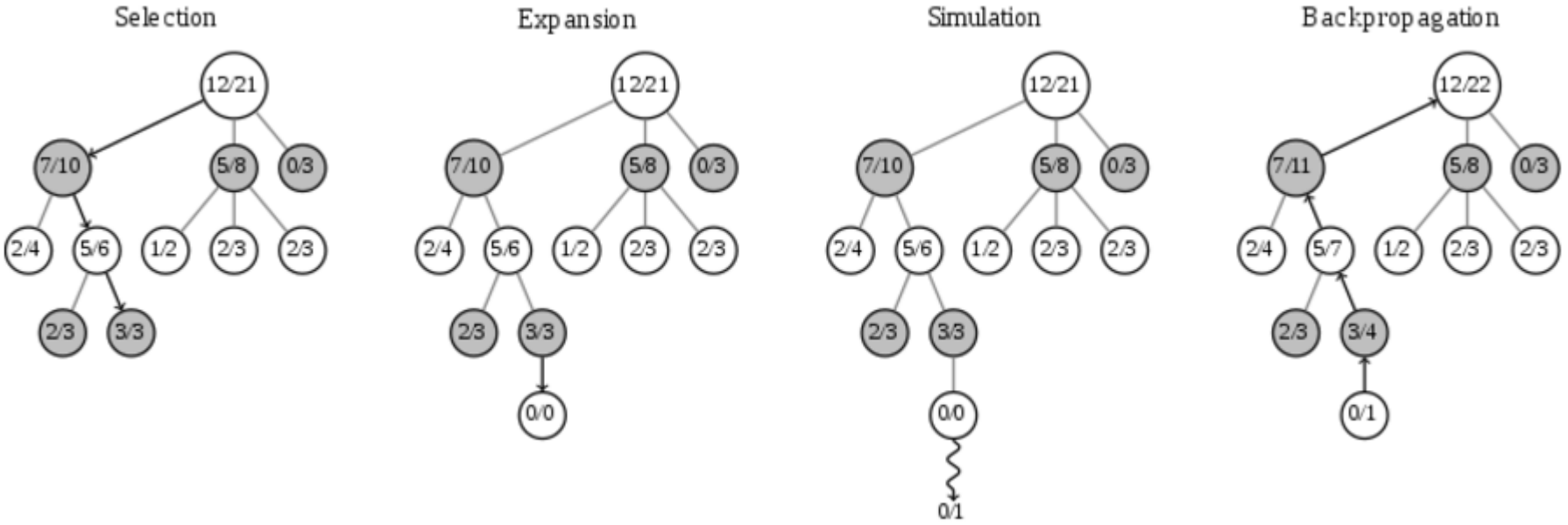
Figure from Chaslot (2006)

每个节点包含两个重要信息:
1)根据模拟结果估计的值;
2)该节点已经被访问的次数

④ **反向传播** Backpropagation: 用模拟的结果输出更新当前行动序列, 也即更新访问过的节点上的信息.

蒙特卡洛树搜索

- 基本的蒙特卡洛树搜索(MCTS)算法



蒙特卡洛树搜索

- 蒙特卡洛树搜索算法优点

① MCTS 不要求任何关于给定的领域策略或者具体实践知识来做出合理的决策. 这个算法可以在没有任何关于博弈游戏除基本规则外的知识的情况下进行有效工作; 这意味着一个简单的 MCTS 实现可以重用在很多的博弈游戏中, 只需要进行微小的调整.

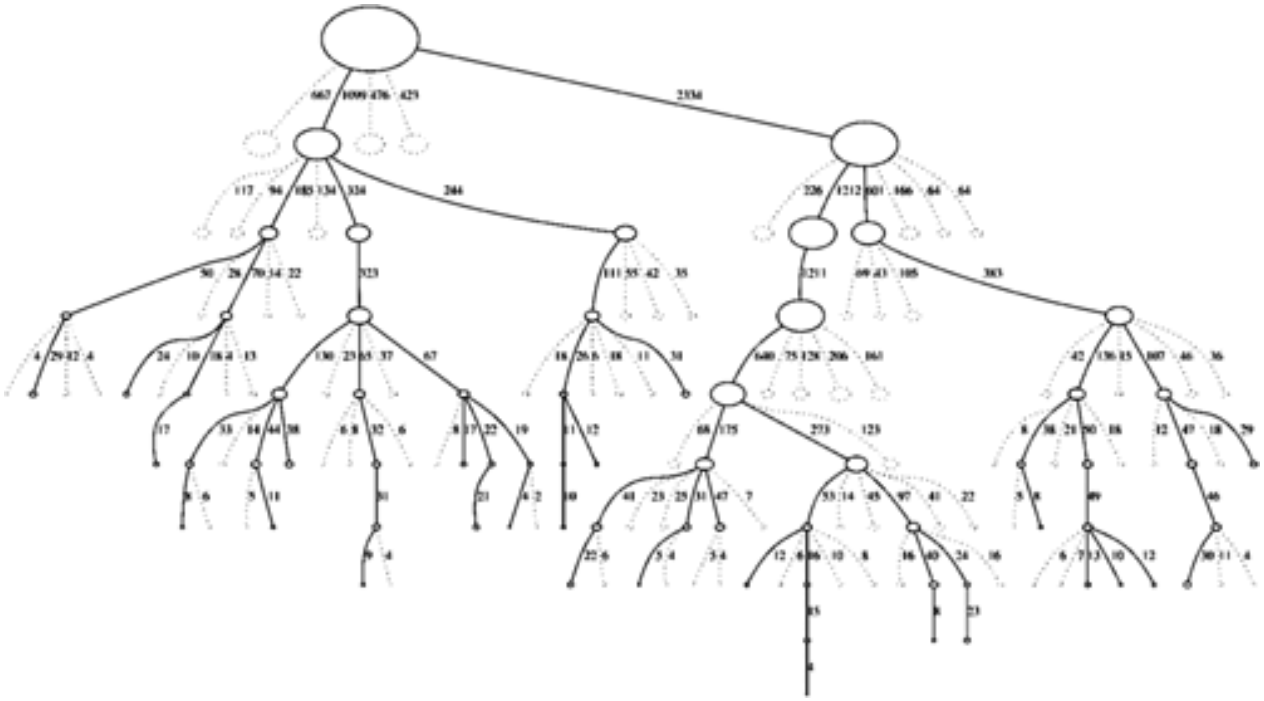
② MCTS 执行一种非对称的树的适应搜索空间拓扑结构的增长. 这个算法会更频繁地访问更加有趣的节点, 并聚焦其搜索时间在更加相关的树的部分. 这使得 MCTS 更加适合那些有着更大的分支因子的博弈游戏, 比如说 19X19 的围棋, 这么大的组合空间会给标准的基于深度或者宽度的搜索方法带来问题, 所以 MCTS 的适应性说明它(最终)可以找到那些更加优化的行动, 并将搜索的工作聚焦在这些部分.

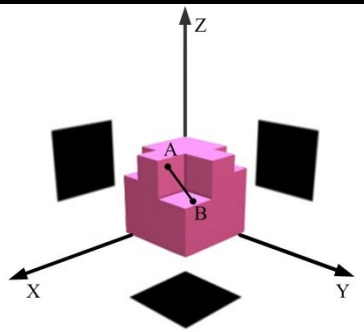
蒙特卡洛树搜索

- 蒙特卡洛树搜索算法优点

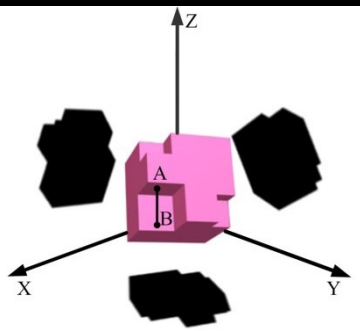
③ 算法可以在任何时间终止, 并返回当前最有的估计. 当前构造出来的搜索树可以被丢弃或者供后续重用.

④ 算法实现非常方便.



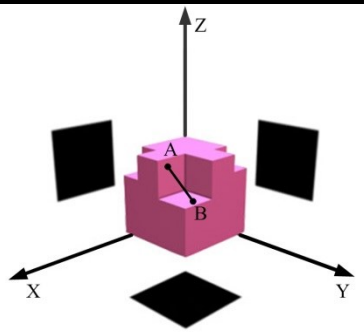


经验分享

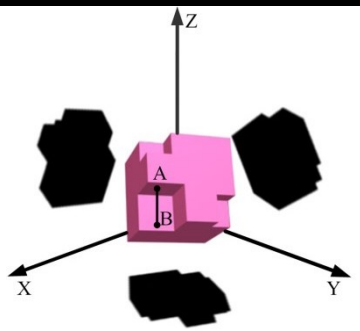


- The ratio $R(M,\alpha,\beta,\gamma)=\frac{Pview(M,\alpha,\beta,\gamma)}{Pface(M,\alpha,\beta,\gamma)}$ relates closely to the convexity of a 3D mesh M.
- $R(M,\alpha,\beta,\gamma)$ changes when the rotation angles α,β,γ vary, then a set of such ratios can be obtained by randomly (or uniformly) rotating the mesh around its center.
- A new shape descriptor CD (i.e., Convexity Distribution) can be constructed based on the distribution of the above-mentioned ratios.

[1] ***Zhouhui Lian***, Afzal Godil, Paul L. Rosin, Xianfang Sun. A New Convexity Measurement for 3D Meshes, CVPR 2012



经验分享



Advantage: The Convexity Distribution employs a histogram instead of a single value to better describe the convexity-related properties of 3D shapes compared to other existing convexity measurements.

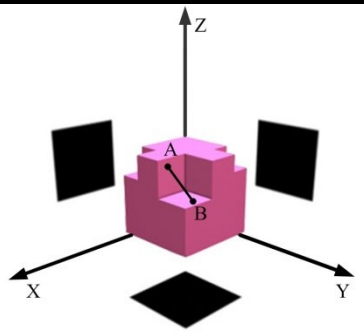
Settings: .

Rotation angles: random floating numbers between 0 to 2π

The number of rotations: $N_{rot} = 10000$

The number of bins of a histogram: $N_{his} = 1024$

[1] ***Zhouhui Lian***, Afzal Godil, Paul L. Rosin, Xianfang Sun. A New Convexity Measurement for 3D Meshes, CVPR 2012



经验分享

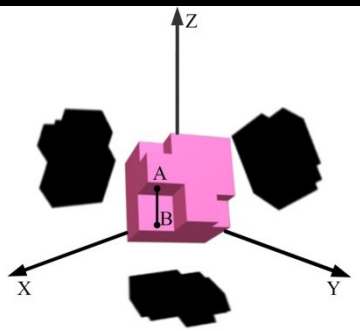
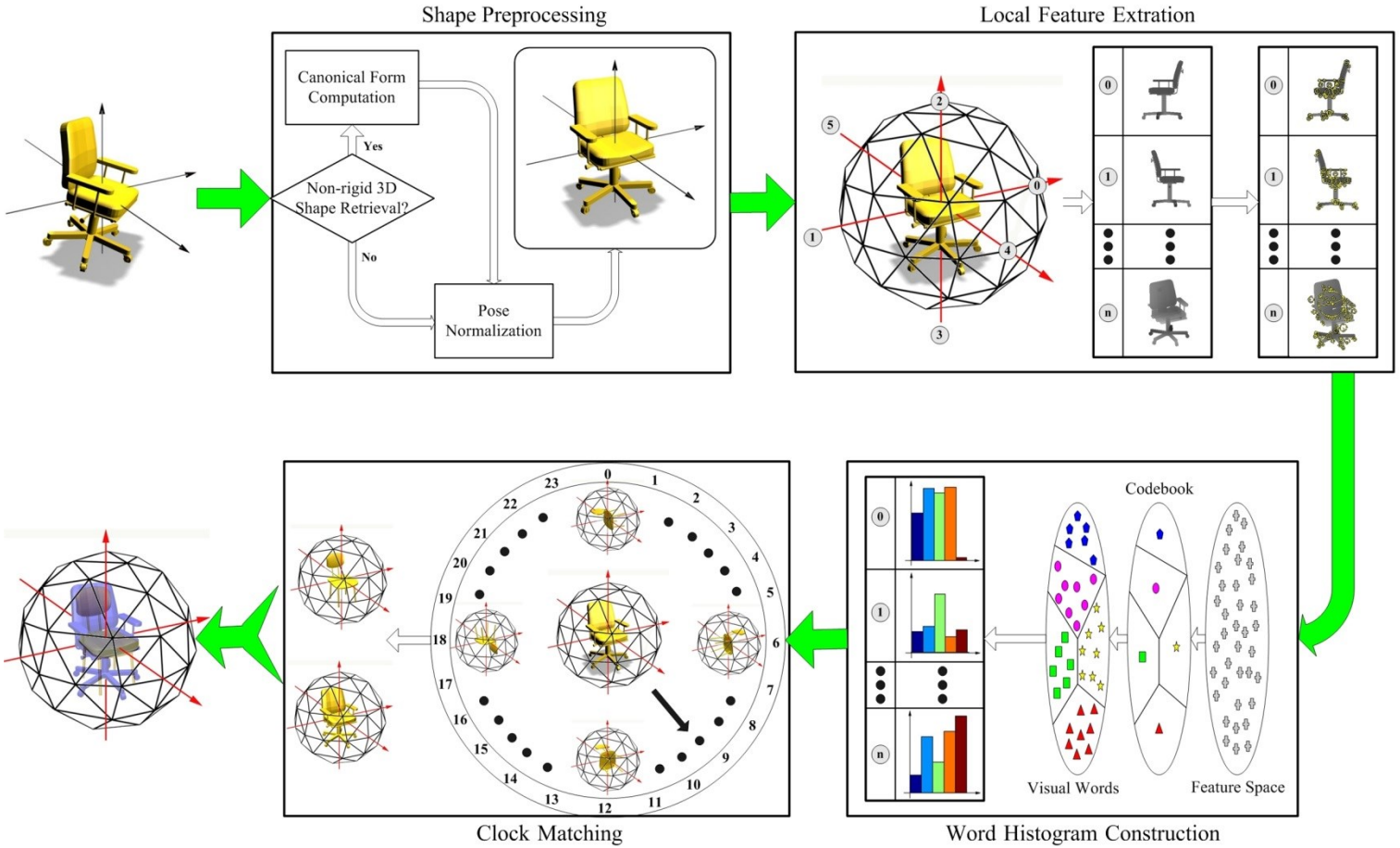


Table 1. Retrieval performance of our methods (CD and $C(M)$) and other convexity measures evaluated on the McGill database.

	NN	1-Tier	2-Tier	DCG
CD	57.3%	41.3%	67.1%	72.9%
$C(M)$	25.9%	26.3%	45.9%	60.4%
$C_1(M)$	27.8%	29.6%	51.9%	62.4%
$C_2(M)$	37.3%	26.0%	43.7%	59.8%

[1] ***Zhouhui Lian***, Afzal Godil, Paul L. Rosin, Xianfang Sun. A New Convexity Measurement for 3D Meshes, CVPR 2012

经验分享



[2] Z. Lian, A. Godil, X. Sun, J. Xiao. CM-BOF: Visual Similarity based 3D Shape Retrieval Using Clock Matching and Bag-of-Features, Machine Vision and Applications (MVAP), vol. 24, no. 8, pp. 1685-1704, 2013

小作业

- 论文讲解(姓名排序随机生成)
- 编程实现: 用线性同余法产生周期为65536的长度为一个周期的0-1均匀分布伪随机数序列, 再用该伪随机数序列生成$N(2,9)$的正态分布伪随机数序列, 最后用生成的伪随机数来计算π, 近似值需在3.14158 – 3.14160之间
- 在教学网提交 http://course.pku.edu.cn/