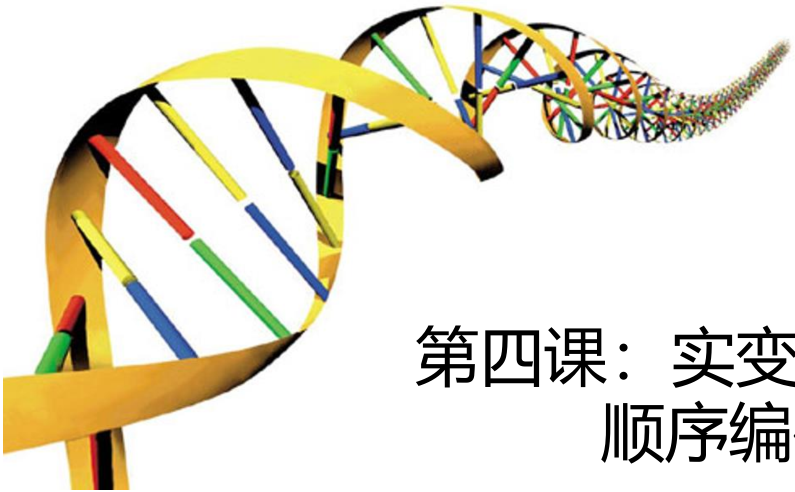
进化优化算法

基于仿生和种群的计算机智能方法



第四课：实变量编码遗传算法、
顺序编码遗传算法

示例2：0-1背包问题 （Knapsack problem）

- 0-1背包问题，就是给定一个背包和许多物品，然后从中挑选出一些物品放入背包
- 假设有n个不同物品，每个物品的价值为c_j，重量为w_j
- 背包能够容纳的总重量为W
- 问题：背包尽可能装入总价值最多的物品，但不超过背包的承重限制
- 背包问题是一类具有单约束的纯整数规划问题，可用于许多工业建模场合的应用，最典型的应用包括资本运算、货物装卸和存储分配等，具有非常重要的研究意义

示例2：0-1背包问题 （Knapsack problem）

背包问题中一些常见的符号：

(1) j ：物品的索引，其中$j = 1, \ 2, . . . , n$

(2) n ：物品的数量；

W ：背包容量；

c_j ：第j个物品的价值；

w_j ：第j个物品的重量

(3) 决策变量

x_j ：0，1决策变量，且满足：$x_j = \begin{cases} 1, & \text{若选择第}j\text{个物品} \\ 0, & \text{否则} \end{cases}$

示例2：0-1背包问题 （Knapsack problem）

0 - 1背包问题的数学公式如下：

$$\max \quad f(x) = \sum_j c_j \times x_j$$

$$s.t. \quad g(x) = \sum_j w_j \times x_j \le W$$

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

$$其中 x_j = \begin{cases} 1, & 若选择第j个物品 \\ 0, & 否则 \end{cases}$$

二进制字符串是0 - 1背包问题解的很自然的表示方式，比如对于物品总数为7个的背包问题决策变量取值如下： $x = [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]$

它表示选出物品2和4并放入背包

示例2：0-1背包问题 （Knapsack problem）

决策变量取值如下： $x = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$

j	c_j	w_j
1	40	40
2	60	50
3	10	30
4	10	10
5	3	10
6	20	40
7	20	30

背包承重W=100

总的价值

$$f(x) = 40x_1 + 60x_2 + 10x_3 + 10x_4$$
$$+ 3x_5 + 20x_6 + 60x_7$$
$$= 60 + 10 = 70$$

总的重量

$$g(x) = 40x_1 + 50x_2 + 30x_3 + 10x_4$$
$$+ 10x_5 + 40x_6 + 30x_7$$
$$= 50 + 10 = 60 \le W = 100$$

二进制表示方式可能产生不可行解：两种处理方法（1.罚函数法 2.解码方法）

不可行解的处理方法：罚函数法

在二进制表示方式中，可能产生不可行解，

*Gordon*和*Whitney*提出罚函数法，

将染色体的惩罚值设置为超出背包容量的总量

对于最大值问题，罚函数可设计为

$$\delta = \min\left\{W,\ \left|\sum_{j=1}^{n} w_j - W\right|\right\}$$

$$p(x) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta}$$

不可行解的处理方法：罚函数法

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30

背包承重W=100，所有物品的重量之和为210

对于最大值问题，罚函数可设计为

$$\delta = \min\left\{W, \left|\left|\sum_{j=1}^{n} w_j - W\right|\right|\right\} = \min\{100, |210 - 100|\} = 100$$

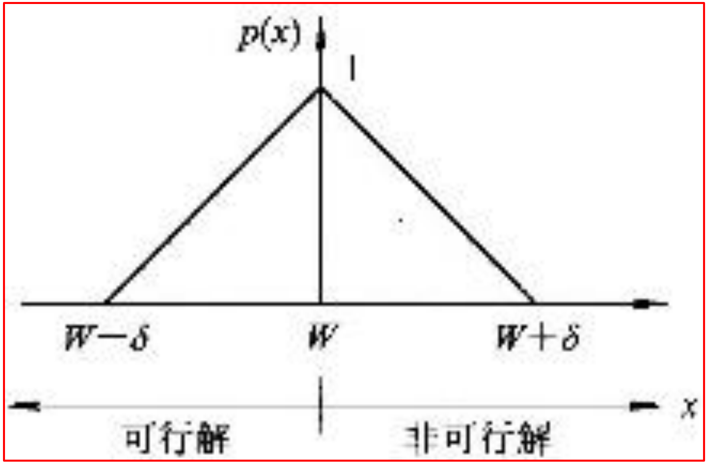
$$p(x) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta} = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - 100\right|}{100}$$

得出$p(x)$的定义后，就可以将优化问题转化为： $\max \quad eval(x) = f(x)p(x)$

不可行解的处理方法：罚函数法

$$\delta = \min\left\{W,\ \left|\sum_{j=1}^{n} w_j - W\right|\right\}$$

$$p(x) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta}$$



得出$p(x)$的定义后，就可以得到如下的适应度函数：

$$eval(x) = f(x)p(x)$$

可以看出，只有当选择物品重量恰为背包容量时才不会产生惩罚，其他情况都会产生不同程度的惩罚

不可行解的处理方法：解码方法

解码方法：

输入：所有物品、物品数量、每个物品重量、每个物品价值

输出：放入背包中的物品

解码方法的运算步骤：

(1) 根据价值重量比c_j/w_j将$x_j = 1$的物品按降序排列

(2) 按照价值重量比次序选择物品，直到背包不能再放入物品

(3) 输出选择的物品并停止运算

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30
c_j/w_j比率	1.0	1.2	0.33	1.0	0.3	0.5	0.67

背包承重W=100

不可行解的处理方法：解码方法

假设给定的染色体如下：

选择的基因

j	1	2	3	4	5	6	7
染色体	0	1	1	0	0	0	1

(1) 根据价值重量比c_j/w_j将$x_j=1$的物品按降序排列，排序后对应的染色体如下

j	2	7	3
降序排列	1	1	1

其中2号、7号和3号基因的c_j/w_j值分别为1.2、 0.67和0.33

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30
c_j/w_j比率	1.0	1.2	0.33	1.0	0.3	0.5	0.67

背包承重W=100

不可行解的处理方法：解码方法

(2) 按照价值重量比次序选择物品，直到背包不能再放入物品

首先选择出2号基因，对于重量$g(x) = 50 \le W$， $f(x) = 60$，符合要求；

其次，选择出7号基因，此时$g(x) = 80 \le W$， $f(x) = 80$，符合要求；

最后，判断3号基因，此时$g(x) = 110 > W$， $f(x) = 90$，此时总重量超出背包最大容量，不再符合要求；

(3) 从而符合背包容量要求的物品为2号和7号物体，此时总重量和相应的价值为$g(x) = 80 \le W, f(x) = 80$

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30
c_j/w_j比率	1.0	1.2	0.33	1.0	0.3	0.5	0.67

背包承重W=100

示例：0-1背包问题 （Knapsack problem）

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30

$$\max \quad f(x) = \sum_j c_j \times x_j$$

$$s.t. \quad g(x) = \sum_j w_j \times x_j \le W \qquad x_j = 0\text{或}1, \quad j = 1, \quad 2, \ldots, n$$

$$\text{其中}x_j = \begin{cases} 1, & \text{若选择第}j\text{个物品} \\ 0, & \text{否则} \end{cases}$$

背包承重W=100

对于最大值问题，罚函数可设计为

$$\delta = \min\left\{W, \quad \left|\sum_{j=1}^{n} w_j - W\right|\right\} = \min\{100, \quad |210 - 100|\} = 100$$

$$p(x) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta} = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - 100\right|}{100}$$

得出$p(x)$的定义后，就可以将优化问题转化为： $\max \qquad eval(x) = f(x)p(x)$

示例：0-1背包问题 （Knapsack problem）

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30

第一步：确定种群规模, 迭代次数，交叉概率，变异概率

$N_p = 4$ $T = 10$ $p_c = 0.8$ $p_m = 0.3$

示例：0-1背包问题 （Knapsack problem）

j	1	2	3	4	5	6	7
价值c_j	40	60	10	10	3	20	20
重量w_j	40	50	30	10	10	40	30

第一步：确定种群规模, 迭代次数，交叉概率，变异概率

$N_p = 4 \qquad T = 10 \qquad p_c = 0.8 \qquad p_m = 0.3$

第二步：随机生成初始种群, 并且计算适应度函数$f(x)p(x)$

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 113 \\ 10 \\ 100 \\ 73 \end{bmatrix} \qquad p = \begin{bmatrix} 0.7 \\ 0.3 \\ 0.7 \\ 0.8 \end{bmatrix} \qquad eval = \begin{bmatrix} 79.1 \\ 3 \\ 70 \\ 58.4 \end{bmatrix}$$

选择：轮盘赌选择

第三步：利用轮盘赌方式选择一对父代解，轮盘赌运行2次

$$P = \begin{bmatrix} \boxed{1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0} \\ 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \boxed{0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0} \\ 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \end{bmatrix} \qquad eval = \begin{bmatrix} \boxed{79.1} \\ 3 \\ \boxed{70} \\ 58.4 \end{bmatrix} \qquad \text{选择的概率}P = \begin{bmatrix} 0.376 \\ 0.014 \\ 0.333 \\ 0.277 \end{bmatrix}$$

假设运行两次轮盘赌得到的第一对父代解为

$$Parent_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$Parent_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

交叉：单点交叉

第三步：利用轮盘赌方式选择一对父代解，轮盘赌运行2次

$$P = \begin{bmatrix} \boxed{1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0} \\ 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \boxed{0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0} \\ 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \end{bmatrix} \qquad eval = \begin{bmatrix} \boxed{79.1} \\ 3 \\ \boxed{70} \\ 58.4 \end{bmatrix} \qquad \text{选择的概率}P = \begin{bmatrix} 0.376 \\ 0.014 \\ 0.333 \\ 0.277 \end{bmatrix}$$

假设运行两次轮盘赌得到的第一对父代解为

$$Parent_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$Parent_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

第四步：随机生成一个随机数决定是否进行交叉

假设$r = 0.3$

$r < p_c = 0.8 \rightarrow$ 进行交叉操作

第五步：随机选择一个交叉点

假设$r = 4$

$$Parent_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & \boxed{1 \quad 0 \quad 0} \end{bmatrix} \qquad offspring_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & \boxed{0 \quad 1 \quad 0} \end{bmatrix}$$
$$Parent_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & \boxed{0 \quad 1 \quad 0} \end{bmatrix} \qquad offspring_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & \boxed{1 \quad 0 \quad 0} \end{bmatrix}$$

选择：轮盘赌选择

第六步：利用轮盘赌方式选择第二对父代解，轮盘赌运行2次

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad eval = \begin{bmatrix} 79.1 \\ 3 \\ 70 \\ 58.4 \end{bmatrix} \quad 选择的概率P = \begin{bmatrix} 0.376 \\ 0.014 \\ 0.333 \\ 0.277 \end{bmatrix}$$

假设运行两次轮盘赌得到的第二对父代解为

$$Parent_3 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$Parent_4 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

交叉：单点交叉

第六步：利用轮盘赌方式选择第二对父代解，轮盘赌运行2次

$$P=\begin{bmatrix}1&1&1&0&1&0&0\\0&0&1&0&0&0&0\\\boxed{0\ 1\ 1\ 1\ 0\ 1\ 0}\\\boxed{1\ 0\ 1\ 0\ 1\ 1\ 0}\end{bmatrix}\qquad eval=\begin{bmatrix}79.1\\3\\\boxed{70}\\\boxed{58.4}\end{bmatrix}\qquad \text{选择的概率}P=\begin{bmatrix}0.376\\0.014\\0.333\\0.277\end{bmatrix}$$

假设运行两次轮盘赌得到的第二对父代解为

$$Parent_3=\begin{bmatrix}0&1&1&1&0&1&0\end{bmatrix}$$
$$Parent_4=\begin{bmatrix}1&0&1&0&1&1&0\end{bmatrix}$$

第七步：随机生成一个随机数决定是否进行交叉

假设$r=0.5$

$r<p_c=0.8\rightarrow$ 进行交叉操作

第八步：随机选择一个交叉点

假设$r=2$

$$Parent_3=\begin{bmatrix}0&1&\boxed{1\ 1\ 0\ 1\ 0}\end{bmatrix}\qquad offspring_3=\begin{bmatrix}0&1&\boxed{1\ 0\ 1\ 1\ 0}\end{bmatrix}$$
$$Parent_4=\begin{bmatrix}1&0&\boxed{1\ 0\ 1\ 1\ 0}\end{bmatrix}\qquad offspring_4=\begin{bmatrix}1&0&\boxed{1\ 1\ 0\ 1\ 0}\end{bmatrix}$$

变异：bit-wise 变异

第九步：对所有子代进行变异操作 $p_m = 0.3$

	Offspring	Random number for mutation	New offspring
O_1	[1 1 1 0 0 1 0]	[0.1 0.4 0.5 0.8 0.6 0.7 0.6]	[0 1 1 0 0 1 0]
O_2	[0 1 1 1 1 0 0]	[0.4 0.6 0.7 0.5 0.9 0.4 0.1]	[0 1 1 1 1 0 1]
O_3	[0 1 1 0 1 1 0]	[0.7 0.1 0.9 0.4 0.6 0.5 0.2]	[0 0 1 0 1 1 1]
O_4	[1 0 1 1 0 1 0]	[0.8 0.6 0.4 0.8 0.7 0.4 0.6]	[1 0 1 1 0 1 0]

$$O = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 90 \\ 143 \\ 53 \\ 80 \end{bmatrix} \quad p = \begin{bmatrix} 0.7 \\ 0.7 \\ 0.6 \\ 0.8 \end{bmatrix} \quad eval = \begin{bmatrix} 63 \\ 100.1 \\ 31.8 \\ 64 \end{bmatrix}$$

幸存策略

第十步：合并所有的解，选择最佳的N_p（$N_p = 4$）个解

$$P = \begin{bmatrix} \boxed{1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0} \\ 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \\ \boxed{0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0} \\ 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \end{bmatrix} \quad f = \begin{bmatrix} 113 \\ 10 \\ 100 \\ 73 \end{bmatrix} \quad p = \begin{bmatrix} 0.7 \\ 0.3 \\ 0.7 \\ 0.8 \end{bmatrix} \quad eval = \begin{bmatrix} \boxed{79.1} \\ 3 \\ \boxed{70} \\ 58.4 \end{bmatrix}$$

$$O = \begin{bmatrix} 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \\ \boxed{0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1} \\ 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \\ \boxed{1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0} \end{bmatrix} \quad f = \begin{bmatrix} 90 \\ 143 \\ 53 \\ 80 \end{bmatrix} \quad p = \begin{bmatrix} 0.7 \\ 0.7 \\ 0.6 \\ 0.8 \end{bmatrix} \quad eval = \begin{bmatrix} 63 \\ \boxed{100.1} \\ 31.8 \\ \boxed{64} \end{bmatrix}$$

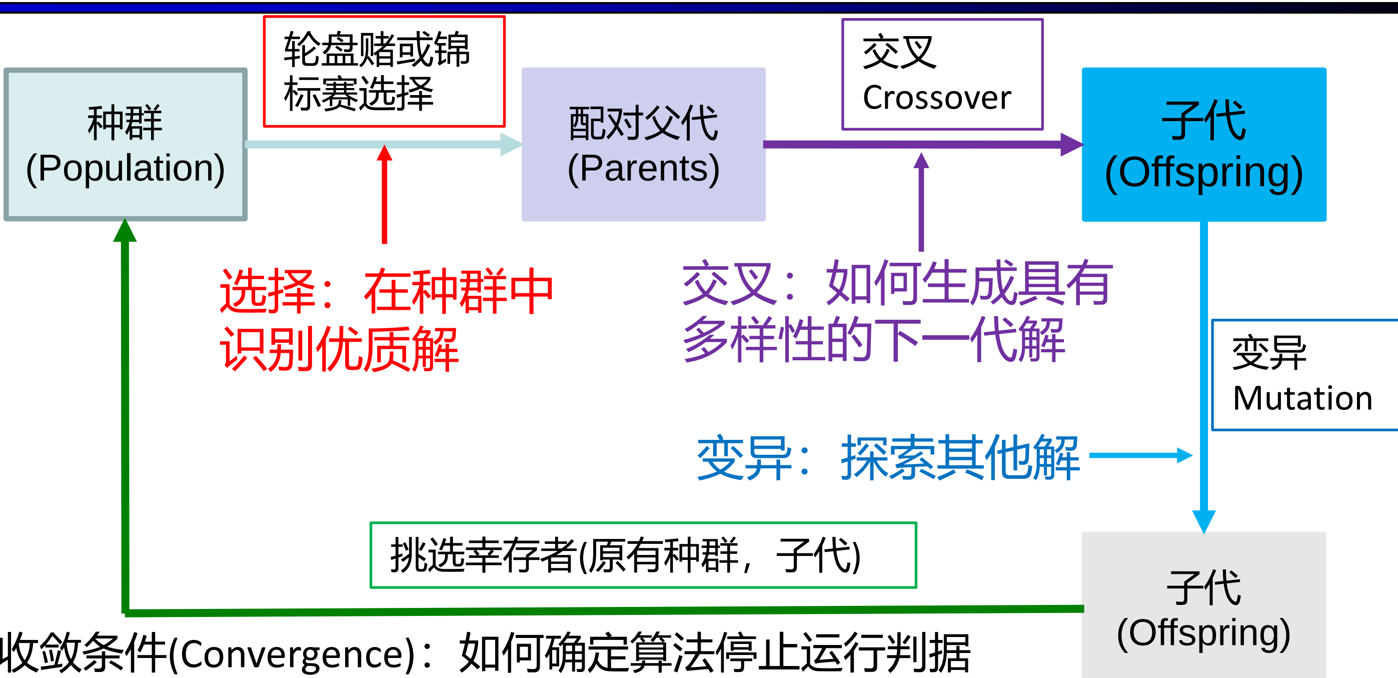
下一代种群

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 113 \\ 100 \\ 143 \\ 80 \end{bmatrix} \quad p = \begin{bmatrix} 0.7 \\ 0.7 \\ 0.7 \\ 0.8 \end{bmatrix} \quad eval = \begin{bmatrix} 79.1 \\ 70 \\ 100.1 \\ 64 \end{bmatrix}$$

二进制编码遗传算法

- 二进制编码遗传算法
- 举例说明二进制编码遗传算法的工作机理
  1. Sphere function
  2. 0-1背包问题 （Knapsack problem）
- 二进制编码遗传算法的优点和局限

遗传算法的基本流程



收敛条件(Convergence)：如何确定算法停止运行判据

遗传算法最大优点：我们不需要知道如何求解问题，我们只需要知道如何评价生成解的品质，通过选择、交叉和变异操作，我们就可以得到优质解

- ✓ 进化&选择过程对所有问题几乎一致
- ✓ 适应度函数&染色体设计：每个特定的优化问题不一样

二进制编码遗传算法

- 一个种群的二进制编码字符串（称为染色体）
- “种群”采用某种“自然选择”机制，结合受遗传学启发的“交叉操作”及“变异操作”等遗传操作，进行进化
- “染色体”的每一位称为“基因”，“基因”由0或1组成
- “选择操作”在种群中选择染色体进行复制，通常适应度值越高的染色体比适应度值低的染色体有更大的机会被选择
- “交叉操作”对两个染色体中的子部分进行交换
- “变异操作”对染色体中某些位置的“基因”进行随机更改

二进制遗传算法的优点和局限

优点：

- 编码解码操作简单易行、便于适应度值的计算
- 交叉、变异等遗传操作便于实现
- 在很多组合优化问题中，目标函数和约束函数均为离散函数，采用二进制编码往往具有直接意义，可以将问题空间的特征与位串的基因相对比，其可应用在整数规划、归纳学习、机器人控制、生产计划等问题中

二进制遗传算法的优点和局限

局限：

- 二进制遗传算法将搜索空间离散化
- 无法实现任意精度
  - 如果采用n位二进制数代表决策变量，那么在决策变量的取值范围内有2^n个不同的值
  - 为了提高精度，必须增加n，对于复杂问题编码过长
  - 增加n会导致变量维度的增加以及种群规模的增加

二进制遗传算法的优点和局限

局限：

- 汉明悬崖(hamming cliffs):二进制编码的一个缺点，就是在某些相邻整数的二进制代码之间有很大的汉明距离（例如：01111=15　10000=16）两个相邻的数字在二进制转换过程中需要同时改变很多位（bits）

14：　0 1 1 1 0

15：　0 1 1 1 1　　1位改变

16：　1 0 0 0 0　　5位改变

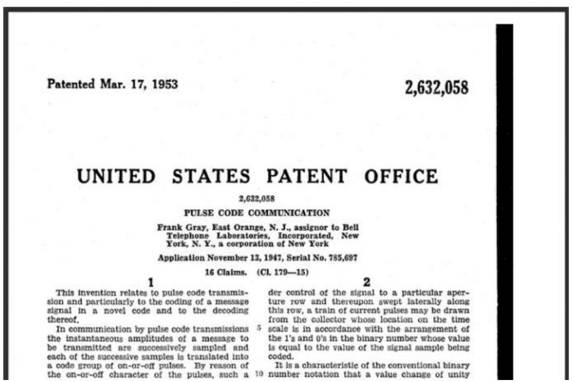
二进制遗传算法的优点和局限

局限：

- 对于一些连续函数优化问题，其随机性使得其局部搜索能力较差，如对于一些高精度的问题，当解迫近于最优解后，由于其变异后表现型变化很大，不连续，因此会远离最优解，达到不稳定。而Gray码能有效地防止这类现象出现

格雷编码

格雷码（Gray Code）是由贝尔实验室的弗兰克·格雷（Frank Gray，1887-1969）在20世纪40年代提出，并在1953年取得美国专利“Pulse Code Communication”。最初目的是在使用PCM（Pusle Code Modulation）方法传输数字信号的过程中降低错误可能。



Patented Mar. 17, 1953 2,632,058

UNITED STATES PATENT OFFICE

2,632,058

PULSE CODE COMMUNICATION

Frank Gray, East Orange, N. J., assignor to Bell Telephone Laboratories, Incorporated, New York, N. Y., a corporation of New York

Application November 13, 1947, Serial No. 785,697

16 Claims. (Cl. 179—15)

1

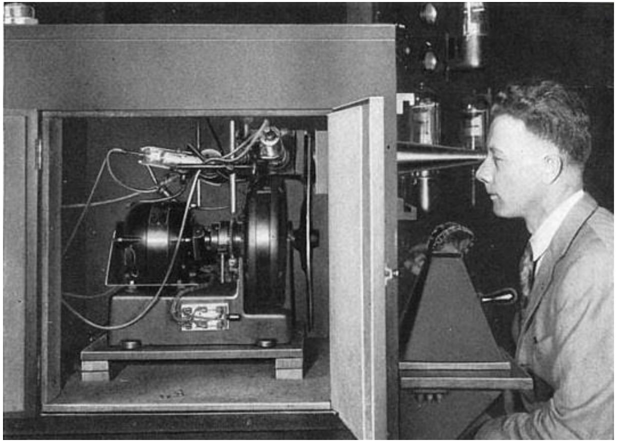
This invention relates to pulse code transmission and particularly to the coding of a message signal in a novel code and to the decoding thereof.

In communication by pulse code transmissions the instantaneous amplitudes of a message to be transmitted are successively sampled and each of the successive samples is translated into a code group of on-or-off pulses. By reason of the on-or-off character of the pulses, such a

2

der control of the signal to a particular aperture row and thereupon swept laterally along this row, a train of current pulses may be drawn from the collector whose location on the time scale is in accordance with the arrangement of the 1's and 0's in the binary number whose value is equal to the value of the signal sample being coded.

It is a characteristic of the conventional binary number notation that a value change of unity



格雷编码

- 也称为反射二进制编码（Reflected Binary Code），格雷编码中相邻的数只有一位不同

十进制	二进制编码	格雷编码	十进制	二进制编码	格雷编码
0	0 0 0 0	0 0 0 0	8	1 0 0 0	1 1 0 0
1	0 0 0 1	0 0 0 1	9	1 0 0 1	1 1 0 1
2	0 0 1 0	0 0 1 1	10	1 0 1 0	1 1 1 1
3	0 0 1 1	0 0 1 0	11	1 0 1 1	1 1 1 0
4	0 1 0 0	0 1 1 0	12	1 1 0 0	1 0 1 0
5	0 1 0 1	0 1 1 1	13	1 1 0 1	1 0 1 1
6	0 1 1 0	0 1 0 1	14	1 1 1 0	1 0 0 1
7	0 1 1 1	0 1 0 0	15	1 1 1 1	1 0 0 0

二进制编码转换为格雷编码

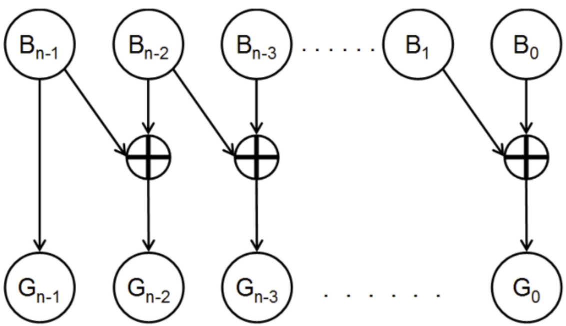
步骤：

1. 记录最重要的位（Most Significant Bit, MSB）保持不变
2. 将二进制编码MSB加到二进制编码下一位，记录它们的和，忽略进位
3. 重复上述过程

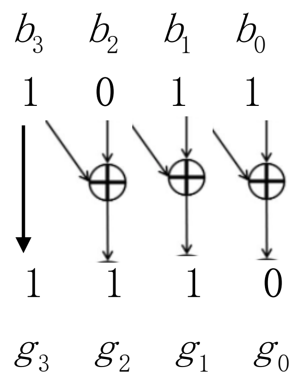
普通二进制码 $\longrightarrow n$ 位格雷码：

$$\begin{cases} G_{n-1} = B_{n-1} \\ G_i = B_i \oplus B_{i+1} \quad ,0 \le i \le n-2 \end{cases}$$

其中 $\oplus$ 表示异或运算（即模2加法），$0 \oplus 0 = 0$，$0 \oplus 1 = 1$，$1 \oplus 0 = 1$，$1 \oplus 1 = 0$。



我们考虑一个二进制编码



格雷编码

格雷编码转换为二进制编码

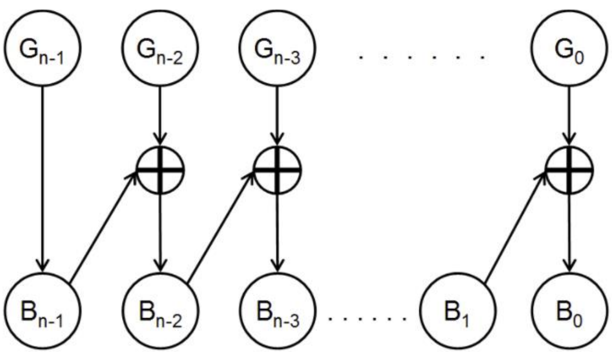
步骤：

1. 记录最重要的位（MSB）保持不变
2. 将二进制编码MSB加到格雷编码下一位，记录它们的和，忽略进位
3. 重复上述过程

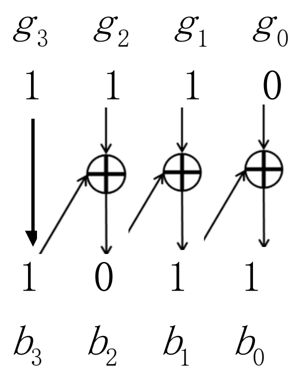
n 位格雷码 $\rightarrow$ 普通二进制码：

$$\begin{cases} B_{n-1} = G_{n-1} \\ B_i = G_i \oplus B_{i+1} \quad ,0 \le i \le n-2 \end{cases}$$

其中 $\oplus$ 表示异或运算（即模2加法），$0 \oplus 0 = 0$，$0 \oplus 1 = 1$，$1 \oplus 0 = 1$，$1 \oplus 1 = 0$。



我们考虑一个格雷编码



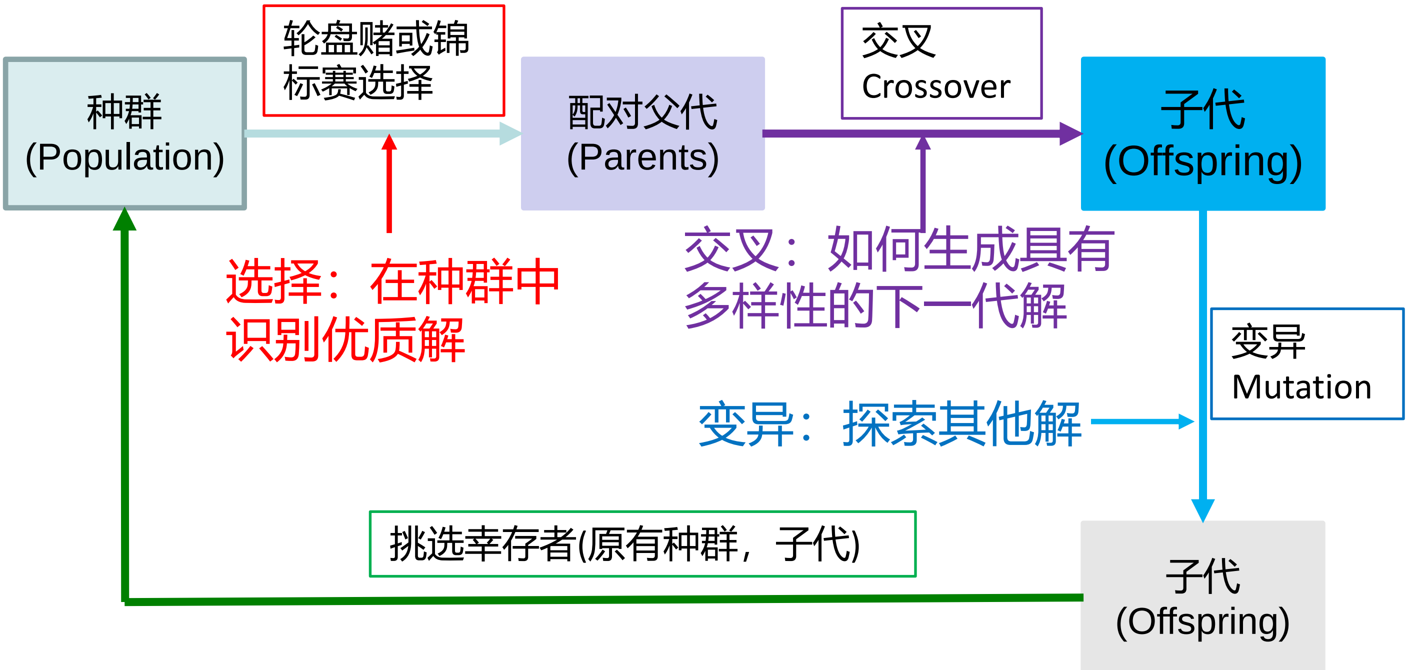
二进制编码

1. 二进制编码遗传算法（Binary encoding genetic algorithm）
2. 实变量编码遗传算法(Real-number encoding genetic algorithm)
3. 顺序编码遗传算法(Order encoding genetic algorithm)

实变量编码遗传算法

- 实变量编码遗传算法
- 举例说明实变量编码遗传算法的工作机理
- 实变量编码遗传算法的优点和局限

遗传算法的基本流程



- 实变量遗传算法的交叉操作和变异操作需要结构性的变化
- 选择操作不需要改变，因为选择操作只需要适应度函数值

实变量编码遗传算法

- 不需要将实变量转变为二进制编码
- 决策变量可以直接用于计算目标函数值
- 二进制编码遗传算法的选择操作可以用于实变量遗传算法
- 交叉操作（例如单点交叉）效果可能不理想
  - 搜索范围只局限于当前决策变量数值
  - 依赖变异操作得到新的决策变量数值
- 对搜索空间的开发依赖于变异操作的修改

随机选取交叉点的位置 = 3

$$parent_1 = [3.5\quad 1.8\quad 9.1\quad 6.4\quad 7.3] \qquad offspring_1 = [3.5\quad 1.8\quad 9.1\quad 4.8\quad 1.7]$$

$$parent_2 = [8.2\quad 2.6\quad 0.3\quad 4.8\quad 1.7] \qquad offspring_2 = [8.2\quad 2.6\quad 0.3\quad 6.4\quad 7.3]$$

实变量编码中的交叉操作

- 线性交叉 (Linear crossover)
- 混合交叉 (Blend crossover)
- 模拟二进制交叉（ Simulated Binary crossover)

实变量编码中的线性交叉

- Wright在1991年提出
- 线性交叉利用父代染色体的线性函数生成子代染色体

例子：

假设P_1和P_2是两个父代染色体的参数值，那么相应的子代染色体的参数值利用下面公式得到

$$C_i = \alpha_i P_1 + \beta_i P_2$$

其中$i = 1, 2, ...n$(子代数量)

α_i和β_i是用户采用的常数值

线性交叉：例子

例子：假设$P_1 = 15.65$和$P_2 = 18.83$，

$\alpha_1 = \beta_1 = 0.5$

$\alpha_2 = 1.5 \qquad \beta_2 = -0.5$

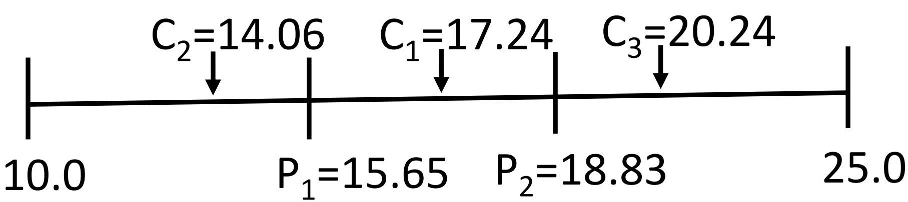
$\alpha_3 = -0.5 \qquad \beta_3 = 1.5$

那么

$$C_1 = \alpha_1 P_1 + \beta_1 P_2 = 0.5 \times 15.65 + 0.5 \times 18.83 = 17.24$$

$$C_2 = \alpha_2 P_1 + \beta_2 P_2 = 1.5 \times 15.65 - 0.5 \times 18.83 = 14.06$$

$$C_3 = \alpha_3 P_1 + \beta_3 P_2 = -0.5 \times 15.65 + 1.5 \times 18.83 = 20.24$$



- ✓ 我们为下一代保留这三个后代中适应性最强的那一个，或适应性最强的前两个，是保留一个或是保留两个取决于具体的进化算法

线性交叉的优点和局限

优点

- 计算简单，运算速度快
- 两个父代可以生成大量的子代
- 可以实现大范围的变化

局限

- 需要确定α_i和β_i
- 对于缺乏经验的人员很难确定正确的α_i和β_i值
- 如果α_i和β_i值选择不合适，问题解可能会陷入局部最优解

实变量编码中的交叉操作

- 线性交叉 (Linear crossover)
- 混合交叉 (Blend crossover)
- 模拟二进制交叉（ Simulated Binary crossover)

实变量编码中的混合交叉

- Eshelman和Schaffer在1993年提出

1）假设P_1和P_2是两个父代染色体的参数值，$P_1 < P_2$

2）混合交叉机制在下面范围内生成子代解

$\left\langle \{P_1 - \alpha(P_2 - P_1)\} \ldots \{P_2 + \alpha(P_2 - P_1)\} \right\rangle$ 其中α是用户采用的常数值

3）利用α和0到1之间随机数r得到参数γ

$$\gamma = (1 + 2\alpha)r - \alpha$$

4)子代解C_1和C_2由下列公式生成

$$C_1 = (1 - \gamma)P_1 + \gamma P_2$$
$$C_2 = (1 - \gamma)P_2 + \gamma P_1$$

混合交叉：例子

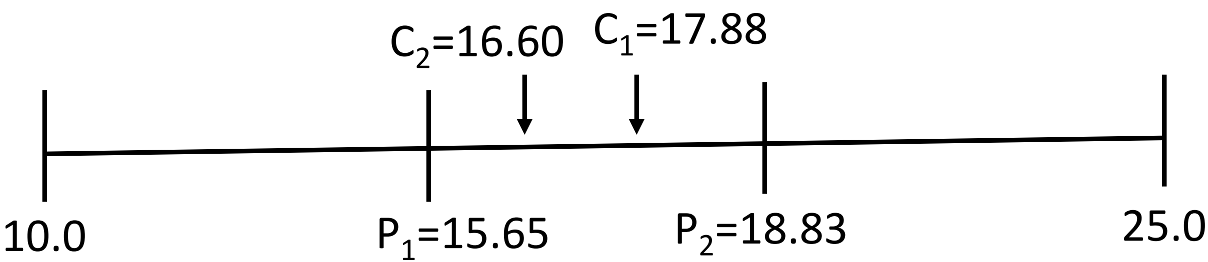
例子：假设$P_1 = 15.65$和$P_2 = 18.83$,

$\alpha = 0.5$和$r = 0.6$

那么$\gamma = (1+2\alpha)r - \alpha = (1 + 2 \times 0.5) \times 0.6 - 0.5 = 0.7$

$C_1 = (1-\gamma)P_1 + \gamma P_2 = (1-0.7) \times 15.65 + 0.7 \times 18.83 = 17.88$

$C_2 = (1-\gamma)P_2 + \gamma P_1 = (1-0.7) \times 18.83 + 0.7 \times 15.65 = 16.60$



混合交叉的优点和局限

优点

- 计算简单，运算速度快
- 两个父代可以生成大量的子代
- 可以实现大范围的变化

局限

- 需要确定α
- 对于缺乏经验的人员很难确定正确的α值
- 如果α值选择不合适，问题解可能会陷入局部最优解

实变量编码中的交叉操作

- 线性交叉 (Linear crossover)
- 混合交叉 (Blend crossover)
- 模拟二进制交叉（ Simulated Binary crossover)

二进制遗传算法单点交叉操作的性质

- 性质1：二进制编码在单点交叉操作前后，二进制编码的解码值的平均值保持不变

$$\text{随机选取交叉点的位置} = 3$$

$$\text{parent}_1 = [1 \quad 0 \quad 1 \quad 0 \quad 1] \qquad \text{offspring}_1 = [1 \quad 0 \quad 1 \quad 1 \quad 1]$$

$$\text{parent}_2 = [0 \quad 1 \quad 1 \quad 1 \quad 1] \qquad \text{offspring}_2 = [0 \quad 1 \quad 1 \quad 0 \quad 1]$$

$$DV_{\text{parent}_1} = 21 \qquad DV_{\text{offspring}_1} = 23$$

$$DV_{\text{parent}_2} = 15 \qquad DV_{\text{offspring}_2} = 13$$

$$AVG_{\text{parent}} = \frac{21+15}{2} = 18 \qquad AVG_{\text{offspring}} = \frac{23+13}{2} = 18$$

二进制遗传算法单点交叉操作的性质

- 性质2：扩展因子β定义为子代解的扩展程度与父代解的扩展程度的比值

扩展因子 $\beta = \dfrac{|o_2 - o_1|}{|p_2 - p_1|}$

三种情况：

1. 收缩交叉， β<1
   - ✓ 子代解的扩展程度小于父代解的扩展程度，子代解被父代解围住

二进制遗传算法单点交叉操作的性质

扩展因子 $\beta = \dfrac{|o_2 - o_1|}{|p_2 - p_1|}$

2. 扩展交叉，β>1
   - ✓ 子代解的扩展程度大于父代解的扩展程度，父代解被子代解围住
3. 静态交叉，β=1
   - ✓ 子代解的扩展程度等于父代解的扩展程度，子代解等于父代解

模拟二进制交叉算子

- Deb和Agrawal于1995年提出，K.Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. Complex Systems, 9(2):115-148, 1995

- 设计思想：在实变量编码遗传算法中模拟二进制编码的单点交叉
  1. 平均值不变性质：二进制编码在单点交叉操作前后，其解码值的平均值保持不变
  2. 扩展因子性质：扩展因子β定义为子代解的扩展程度与父代解的扩展程度的比值

模拟二进制交叉算子

- 利用平均值保持不变的性质，子代解通过下式生成

$$O_a = 0.5\left[\left(P_a^{'} + P_b^{'}\right) + \beta\left(P_a^{'} - P_b^{'}\right)\right] = 0.5\left[\left(1 + \beta\right)P_a^{'} + \left(1 - \beta\right)P_b^{'}\right]$$
$$O_b = 0.5\left[\left(P_a^{'} + P_b^{'}\right) - \beta\left(P_a^{'} - P_b^{'}\right)\right] = 0.5\left[\left(1 - \beta\right)P_a^{'} + \left(1 + \beta\right)P_b^{'}\right]$$

$P_a^{'}$为父代1　　　　O_a为子代1

$P_b^{'}$为父代2　　　　O_b为子代2

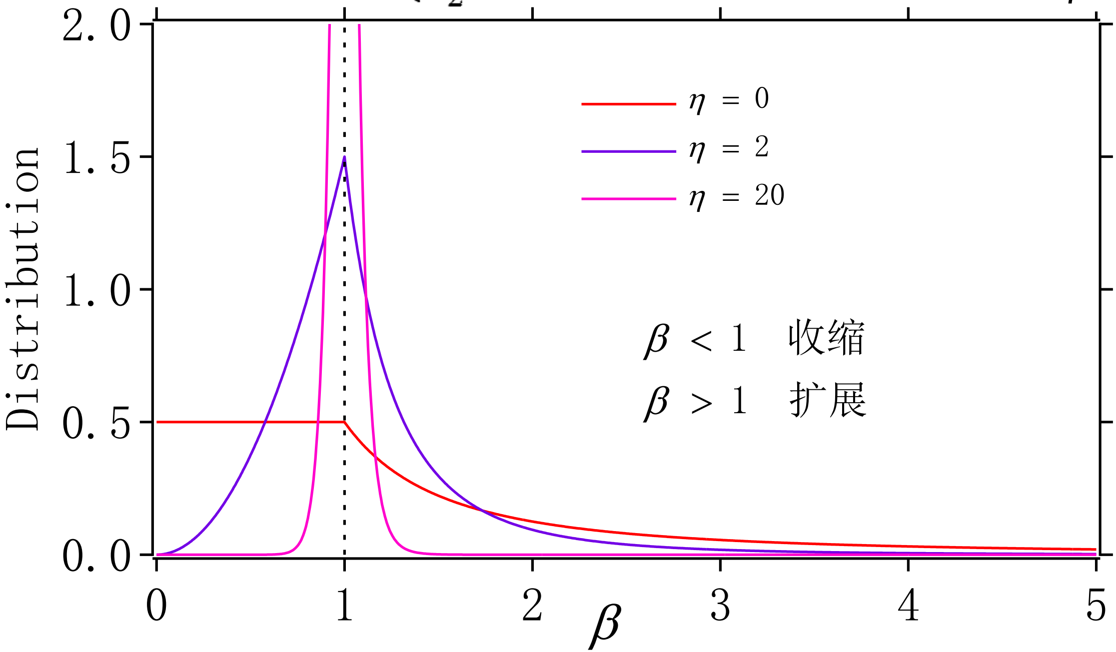
- 上式保证了子代解的平均值等于父代解的平均值

模拟二进制交叉算子

其中 β_k 是由下面的概率密度函数生成的随机数:

$$\mathrm{PDF}(\beta)=\begin{cases}\frac{1}{2}(\eta+1)\beta^{\eta}, & \text{如果 } 0\leqslant\beta\leqslant 1,\\ \frac{1}{2}(\eta+1)\beta^{-(\eta+2)}, & \text{如果 } \beta>1,\end{cases} \tag{8.56}$$

η是用户采用的常数



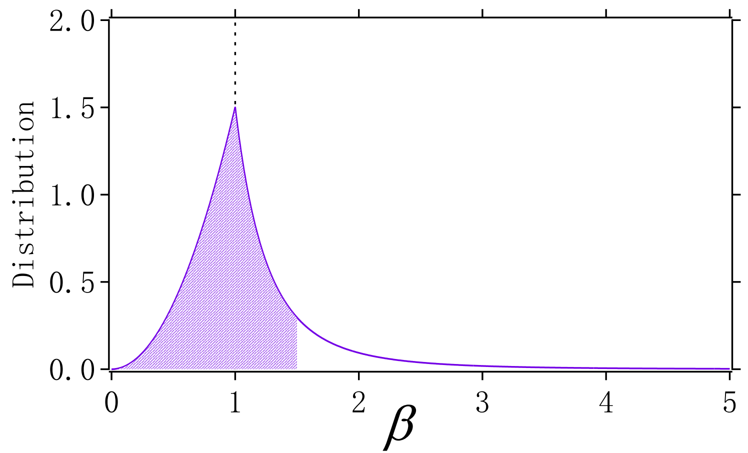
如何计算β?

通过计算PDF函数下面的面积等于u（u是 $\in [0,1]$的随机数）

如果$u \le 0.5$

$$\int_0^{\beta} \frac{1}{2}(\eta+1)\beta^{\eta} d\beta = \frac{1}{2}\beta^{\eta+1} = u$$

$$\Rightarrow \beta = (2u)^{1/(\eta+1)}$$



otherwise

$$0.5 + \int_1^{\beta} \frac{1}{2}(\eta+1)\beta^{-(\eta+2)} d\beta = 0.5 - \frac{1}{2}\beta^{-(\eta+1)} + 0.5 = u$$

$$\Rightarrow \beta = \left(\frac{1}{2(1-u)}\right)^{1/(\eta+1)}$$

模拟二进制交叉算子

- 设计思想：模拟二进制编码的单点交叉
- 需要两个父代产生两个子代
- 两个子代之间的差距与两个父代之间的差距成正比

$$O_a - O_b = \beta(P_a^{'} - P_b^{'})$$

- 计算β

$$\beta = \begin{cases} (2u)^{1/(\eta+1)} & \text{如果随机数} u \le 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta+1)} & otherwise \end{cases}$$

其中η是用户指定的常数

模拟二进制交叉算子

$$\beta = \begin{cases} (2u)^{1/(\eta+1)} & \text{如果随机数} u \le 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta+1)} & otherwise \end{cases}$$

其中η是用户指定的常数

β和u是D维向量，每个变量对应一个β_j和随机数u_j

$Case1$：收缩交叉 $\rightarrow \beta < 1$

$Case2$：扩展交叉 $\rightarrow \beta > 1$

$Case3$：静态交叉 $\rightarrow \beta = 1$

从父代解产生子代解的概率分布是多项式分布

- 生成子代

$$O_a = 0.5\left[(1+\beta)P_a^{'} + (1-\beta)P_b^{'}\right]$$ $P_a^{'}$为父代1 O_a为子代1

$$O_b = 0.5\left[(1-\beta)P_a^{'} + (1+\beta)P_b^{'}\right]$$ $P_b^{'}$为父代2 O_b为子代2

模拟二进制交叉算子

- 交叉操作以较高的概率发生
- 两个子代关于父代对称
- 在一次交叉操作中避免了子代偏向于任何特定的父代
- 取β为常数
  - 如果父代的距离较大，产生的子代的距离也较大
  - 如果父代比较接近，产生的子代也比较接近

$$Case1：假设 P_a^{'} = 2 \quad P_b^{'} = 8 \quad \beta = 0.8$$

$$O_1 = 0.5\left[(1+0.8)\times 2 + (1-0.8)\times 8\right] = 2.6$$

$$O_2 = 0.5\left[(1-0.8)\times 2 + (1+0.8)\times 8\right] = 7.4$$

$$Case2：假设 P_a^{'} = 4 \quad P_b^{'} = 5 \quad \beta = 0.8$$

$$O_1 = 0.5\left[(1+0.8)\times 4 + (1-0.8)\times 5\right] = 4.1$$

$$O_2 = 0.5\left[(1-0.8)\times 4 + (1+0.8)\times 5\right] = 4.9$$

$$O_a - O_b = \beta(P_a^{'} - P_b^{'})$$

$$O_a = 0.5\left[(1+\beta)P_a^{'} + (1-\beta)P_b^{'}\right]$$

$$O_b = 0.5\left[(1-\beta)P_a^{'} + (1+\beta)P_b^{'}\right]$$

示例：β的影响

假设$P_a^{'} = 2 \qquad P_b^{'} = 5$

$Case1$：收缩交叉（$\beta < 1$） $\Rightarrow$ 子代更近

$\beta = 0.6$

$O_1 = 0.5\left[\left(1 + 0.6\right) \times 2 + \left(1 - 0.6\right) \times 5\right] = 2.6$

$O_2 = 0.5\left[\left(1 - 0.6\right) \times 2 + \left(1 + 0.6\right) \times 5\right] = 4.4$

$Case2$：静态交叉（$\beta = 1$） $\Rightarrow$ 子代和父代一致

$O_1 = 0.5\left[\left(1 + 1\right) \times 2 + \left(1 - 1\right) \times 5\right] = 2$

$O_2 = 0.5\left[\left(1 - 1\right) \times 2 + \left(1 + 1\right) \times 5\right] = 5$

$Case3$：扩展交叉（$\beta > 1$） $\Rightarrow$ 子代更远

$\beta = 1.4$

$O_1 = 0.5\left[\left(1 + 1.4\right) \times 2 + \left(1 - 1.4\right) \times 5\right] = 1.4$

$O_2 = 0.5\left[\left(1 - 1.4\right) \times 2 + \left(1 + 1.4\right) \times 5\right] = 5.6$

$O_a - O_b = \beta(P_a^{'} - P_b^{'})$

$O_a = 0.5\left[\left(1 + \beta\right)P_a^{'} + \left(1 - \beta\right)P_b^{'}\right]$

$O_b = 0.5\left[\left(1 - \beta\right)P_a^{'} + \left(1 + \beta\right)P_b^{'}\right]$

模拟二进制交叉的优点和局限

优点

- 两个父代可以生成大量的子代
- 结果准确，通常可以找到全局最优
- 迭代次数更少
- 交叉操作与染色体的长度无关

局限

- 计算量大
- 如果参数选择不合适，可能导致早熟收敛，问题解可能会陷入局部最优解

实变量编码遗传算法的变异操作

实变量编码遗传算法中变异操作存在许多变形：

- 随机变异（Random mutation）
- 多项式变异（Polynomial mutation）

随机变异（Random mutation）

变异解采用下列公式得到

$$O_{\text{mutated}} = O_{original} + (r - 0.5) \times \Delta$$

其中r为0到1之间的随机数，Δ是用户选择的最大扰动值

例如：

$$O_{original} = 15.6$$

$$r = 0.7$$

$$\Delta = 2.5$$

$$\Rightarrow O_{\text{mutated}} = O_{original} + (r - 0.5) \times \Delta$$

$$= 15.6 + (0.7 - 0.5) \times 2.5 = 16.1$$

实变量编码遗传算法的变异操作

实变量编码遗传算法中变异操作存在许多变形：

- 随机变异（Random mutation）
- 多项式变异（Polynomial mutation）

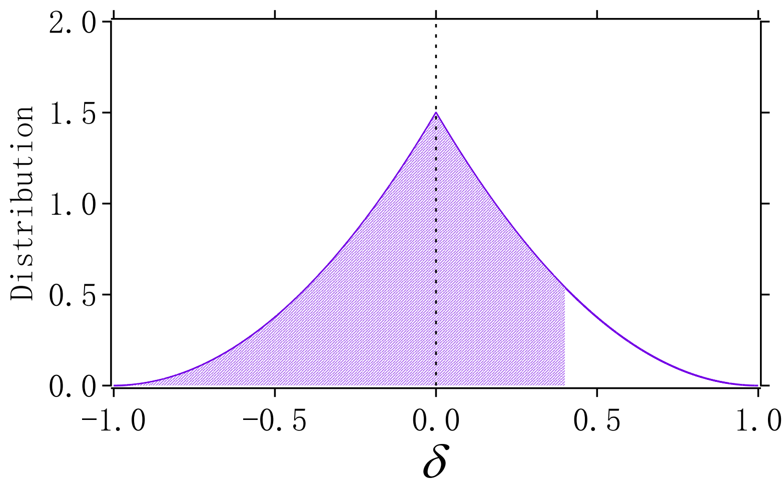
多项式变异（Polynomial mutation）

δ采用如下多项式概率密度分布函数 $P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|)^{\eta_m}$

计算 δ：概率密度分布函数曲线下面积等于r（r是 $\in [0,1]$的随机数）

$$\delta = \begin{cases} (2r)^{1/(\eta_m+1)} - 1 & \text{如果} r < 0.5 \\ 1 - [2(1-r)]^{1/(\eta_m+1)} & \text{如果} r \geq 0.5 \end{cases}$$

η_m是用户采用的常数



多项式变异（Polynomial mutation）

生成子代

$$O_{mutated} = O_{original} + (ub - lb)\delta$$

其中O是子代，ub是上限，lb是下限

或

$$O_{\mathrm{mutated}} = O_{original} + \delta \times \Delta$$

其中Δ是用户选择的最大扰动值

一个子代生成一个新的子代

多项式变异

假设$O_{original} = 15.6$，$r = 0.7$，$\eta_m = 2$，$\Delta = 1.2$

$r \geq 0.5$

$$\Rightarrow \delta = 1 - \left[2 \times \left(1 - r\right)\right]^{\frac{1}{\eta_m + 1}} = 1 - \left[2 \times \left(1 - 0.7\right)\right]^{\frac{1}{3}} = 0.1566$$

$$O_{\mathrm{mutated}} = O_{original} + \delta \times \Delta = 15.6 + 0.1566 \times 1.2 = 15.8$$

实变量编码遗传算法

- 实变量编码遗传算法
- 举例说明实变量编码遗传算法的工作机理
- 实变量编码遗传算法的优点和局限

例题： Sphere function

$$\min \quad f(x)=\sum_{i=1}^{4} x_i^2 \qquad 0 \le x_i \le 10 \qquad i=1,\ 2,\ 3,\ 4$$

决策变量：x_1, x_2, x_3, x_4 $\qquad f(x)=x_1^2+x_2^2+x_3^2+x_4^2$

第一步：确定种群规模, 交叉概率，变异概率，最大迭代次数，交叉和变异的 *distribution index*

$N_p=6$, $p_c=0.8$, $p_m=0.2$, $T=10$, $\eta_c=20$, $\eta_m=20$

例题：Sphere function

$$\min \quad f(x) = \sum_{i=1}^{4} x_i^2 \qquad 0 \le x_i \le 10 \qquad i = 1,\ 2,\ 3,\ 4$$

决策变量：x_1, x_2, x_3, x_4 $\qquad f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2$

第一步：确定种群规模, 交叉概率，变异概率，最大迭代次数，交叉和变异的 *distribution index*

$N_p = 6,\quad p_c = 0.8,\quad p_m = 0.2,\quad T = 10,\quad \eta_c = 20,\quad \eta_m = 20$

第二步：在决策变量的取值范围内生成随机解, 计算适应度函数值

$$P = \begin{bmatrix} 4 & 0 & 0 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \\ 5 & 8 & 1 & 3 \end{bmatrix} \qquad f = \begin{bmatrix} 80 \\ 140 \\ 35 \\ 102 \\ 113 \\ 99 \end{bmatrix}$$

选择：锦标赛选择

第三步：随机选择两个解参加锦标赛

假设选择的两个解是

$$P_3 = \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix} \qquad f_3 = 35$$

$$P_2 = \begin{bmatrix} 3 & 1 & 9 & 7 \end{bmatrix} \qquad f_2 = 140$$

$$P = \begin{bmatrix} 4 & 0 & 0 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 6 & 2 & 8 & 3 \\ 5 & 8 & 1 & 3 \end{bmatrix} \qquad f = \begin{bmatrix} 80 \\ 140 \\ 35 \\ 102 \\ 113 \\ 99 \end{bmatrix}$$

第四步：比较适应度函数值，选择胜利者

进入父代群（*Mating Pool*）

$$f_3 < f_2 \rightarrow f_3$$

P_3（35）

P_2（140）

P_3（35）

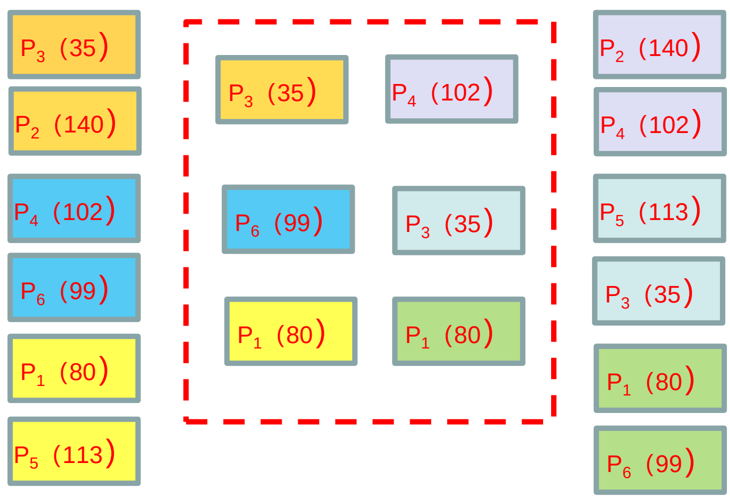
选择：锦标赛选择

第五步：经过六次锦标赛

最好的解在父代群中有两个

最差的解没有进入父代群

$$P=\begin{bmatrix}4&0&0&8\\3&1&9&7\\0&3&1&5\\2&1&4&9\\6&2&8&3\\5&8&1&3\end{bmatrix}\qquad f=\begin{bmatrix}80\\140\\35\\102\\113\\99\end{bmatrix}$$



模拟二进制交叉算子的步骤

输入：P， D， p_c, η_c

1. 从父代群中随机选择两个父代（e.g. P_a’和P_b’）
2. 生成0到1之间的随机数
3. 如果$r \geq p_c$, 将父代解拷贝到子代解
4. 如果$r<p_c$, 生成D个随机数(u)，每个变量对应一个随机数u_j
5. 针对每个变量确定β
6. 生成两个子代（O_a和O_b）

$$O_a - O_b = \beta(P_a' - P_b')$$

$$O_a = 0.5\left[\left(1+\beta\right)P_a' + \left(1-\beta\right)P_b'\right]$$

$$O_b = 0.5\left[\left(1-\beta\right)P_a' + \left(1+\beta\right)P_b'\right]$$

$$\beta = \begin{cases} (2u)^{1/(\eta_c+1)} & \text{如果} u \leq 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta_c+1)} & otherwise \end{cases}$$

交叉

第六步：随机选择两个父代进行交叉

假设选择的两个解是

$$Parent_1 = \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix} \qquad Parent_3 = \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix}$$

第七步：产生一个随机数决定是否进行交叉

假设$r = 0.2$

$\mathrm{r} < \mathrm{p_c} \rightarrow$ 进行交叉操作

$$Parent = \begin{bmatrix} \boxed{0 \quad 3 \quad 1 \quad 5} \\ 5 \quad 8 \quad 1 \quad 3 \\ \boxed{4 \quad 0 \quad 0 \quad 8} \\ 2 \quad 1 \quad 4 \quad 9 \\ 0 \quad 3 \quad 1 \quad 5 \\ 4 \quad 0 \quad 0 \quad 8 \end{bmatrix}$$

$\boxed{p_c = 0.8 \qquad \eta_c = 20}$

交叉

第八步：为每个变量生成一个随机数执行交叉操作

假设 $u = \begin{bmatrix} \boxed{0.2} & 0.6 & 0.1 & \boxed{0.8} \end{bmatrix}$

$(u \leq 0.5)$

$$\beta = (2 \times 0.2)^{1/(20+1)} = 0.96$$

$(u > 0.5)$

$$\beta = \left(\frac{1}{2 \times (1 - 0.8)}\right)^{1/(20+1)} = 1.04$$

$$\beta = \begin{cases} (2u)^{1/(\eta_c+1)} & \text{如果} u \leq 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta_c+1)} & otherwise \end{cases}$$

交叉

$$\beta = \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}$$

$$\boxed{O_a = 0.5\left[(1+\beta)P_a^{'} + (1-\beta)P_b^{'}\right]}$$

第九步：生成两个子代

$$offspring_1 = 0.5 \times \begin{bmatrix} (1 + \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix} + \\ (1 - \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \end{bmatrix}$$

$$Parent = \begin{bmatrix} 0 & 3 & 1 & 5 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 2 & 1 & 4 & 9 \\ 0 & 3 & 1 & 5 \\ 4 & 0 & 0 & 8 \end{bmatrix}$$

$$\boxed{p_c = 0.8 \qquad \eta_c = 20}$$

交叉

$$\beta = \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}$$

第九步：生成两个子代

$$\boxed{O_a = 0.5\left[(1+\beta)P_a^{'} + (1-\beta)P_b^{'}\right]}$$

$$offspring_1 = 0.5 \times \begin{bmatrix} (1 + \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix} + \\ (1 - \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \end{bmatrix}$$

$$\boxed{O_b = 0.5\left[(1-\beta)P_a^{'} + (1+\beta)P_b^{'}\right]}$$

$$offspring_2 = 0.5 \times \begin{bmatrix} (1 - \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix} - \\ (1 + \begin{bmatrix} 0.96 & 1.01 & 0.93 & 1.04 \end{bmatrix}) \times \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 3.92 & -0.02 & 0.03 & 8.06 \end{bmatrix}$$

$$Parent = \begin{bmatrix} 0 & 3 & 1 & 5 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 2 & 1 & 4 & 9 \\ 0 & 3 & 1 & 5 \\ 4 & 0 & 0 & 8 \end{bmatrix}$$

$$\boxed{p_c = 0.8 \qquad \eta_c = 20}$$

交叉

$$offspring_1 = \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \end{bmatrix}$$

$$offspring_2 = \begin{bmatrix} 3.92 & -0.02 & 0.03 & 8.06 \end{bmatrix}$$

第十步：检查是否满足变量取值范围

$$offspring_2 = \begin{bmatrix} 3.92 & -0.02 & 0.03 & 8.06 \end{bmatrix} \rightarrow \begin{bmatrix} 3.92 & 0 & 0.03 & 8.06 \end{bmatrix}$$

$offspring_1$满足取值范围

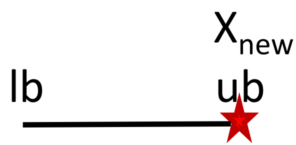
$$\boxed{0 \le x_i \le 10}$$

保证新解在取值范围内

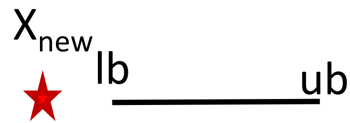
X_{new}在取值范围内

不需要修改任何值

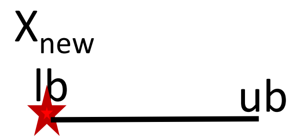
X_{new}违反取值范围的上限



将X_{new}移到取值范围的上限



X_{new}违反取值范围的下限



将X_{new}移到取值范围的下限

交叉

第十一步：随机选择两个父代进行交叉

$$p_c = 0.8 \qquad \eta_c = 20$$

假设选择的两个解是

$$Parent_2 = \begin{bmatrix} 5 & 8 & 1 & 3 \end{bmatrix} \qquad Parent_6 = \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix}$$

$$Parent = \begin{bmatrix} 0 & 3 & 1 & 5 \\ \boxed{5 \quad 8 \quad 1 \quad 3} \\ 4 & 0 & 0 & 8 \\ 2 & 1 & 4 & 9 \\ 0 & 3 & 1 & 5 \\ \boxed{4 \quad 0 \quad 0 \quad 8} \end{bmatrix}$$

第十二步：产生一个随机数决定是否进行交叉

假设$r = 0.9$ $\qquad r > p_c \rightarrow$ 不进行交叉操作

第十三步：将两个父代解拷贝到子代解

$$Offspring_3 = Parent_2 = \begin{bmatrix} 5 & 8 & 1 & 3 \end{bmatrix}$$

$$Offspring_4 = Parent_6 = \begin{bmatrix} 4 & 0 & 0 & 8 \end{bmatrix}$$

$$Offspring = O = \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \end{bmatrix}$$

交叉

第十四步：随机选择两个父代进行交叉

$p_c = 0.8 \qquad \eta_c = 20$

假设选择的两个解是

$Parent_4 = \begin{bmatrix} 2 & 1 & 4 & 9 \end{bmatrix} \qquad Parent_5 = \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix}$

第十五步：产生一个随机数决定是否进行交叉

假设$r = 0.4$

$r < p_c \rightarrow$ 进行交叉操作

$$Parent = \begin{bmatrix} 0 & 3 & 1 & 5 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 2 & 1 & 4 & 9 \\ 0 & 3 & 1 & 5 \\ 4 & 0 & 0 & 8 \end{bmatrix}$$

交叉

第十六步：为每个变量生成一个随机数执行交叉操作

假设$u = \begin{bmatrix} 0.3 & 0.1 & 0.8 & 0.6 \end{bmatrix}$

$\beta = \begin{bmatrix} 0.98 & 0.93 & 1.04 & 1.01 \end{bmatrix}$

$p_c = 0.8 \qquad \eta_c = 20$

第十七步：生成两个子代

$Offspring_5 = \begin{bmatrix} 1.98 & 1.07 & 4.06 & 9.02 \end{bmatrix}$

$Offspring_6 = \begin{bmatrix} 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix}$

$$Parent = \begin{bmatrix} 0 & 3 & 1 & 5 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 2 & 1 & 4 & 9 \\ 0 & 3 & 1 & 5 \\ 4 & 0 & 0 & 8 \end{bmatrix}$$

$$O_a - O_b = \beta(P_a' - P_b')$$

$$O_a = 0.5\left[(1+\beta)P_a' + (1-\beta)P_b'\right]$$

$$O_b = 0.5\left[(1-\beta)P_a' + (1+\beta)P_b'\right]$$

$$\beta = \begin{cases} (2u)^{1/(\eta_c+1)} & 如果u \le 0.5 \\ \left(\dfrac{1}{2(1-u)}\right)^{1/(\eta_c+1)} & otherwise \end{cases}$$

多项式变异的步骤

输入： P， D， p_m, η_m

1. 生成0到1之间的随机数u
2. 如果$u \geq p_m$, 该子代解不进行变异
3. 如果$u<p_m$, 生成D个随机数(r)，每个变量对应一个随机数r
4. 确定每个变量的δ

$$\delta = \begin{cases} (2r)^{1/(\eta_m+1)} - 1 & \text{如果} r < 0.5 \\ 1 - [2(1-r)]^{1/(\eta_m+1)} & \text{如果} r \geq 0.5 \end{cases}$$

5. 利用下式对子代解进行变异

$$O_{mutated} = O_{original} + (ub - lb)\delta$$ 其中O是子代，ub是上限，lb是下限

变异

第十八步：选择第一个子代进行变异

$$Offspring_1 = [0.08 \quad 3.01 \quad 0.97 \quad 4.94]$$

$$O = \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 1.98 & 1.07 & 4.06 & 9.02 \\ 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix}$$

第十九步：产生一个随机数决定是否进行变异

假设$r = 0.1$

$r < p_m \rightarrow$ 进行变异操作

$$p_m = 0.2 \qquad q_m = 20$$

第二十步：为每个变量生成一个随机数执行变异操作

假设$r = [0.6 \quad 0.1 \quad 0.2 \quad 0.8]$

$(r \geq 0.5)$ $\qquad \delta_1 = 1 - (2 \times (1 - 0.6))^{1/(20+1)} = 0.01$

$(r < 0.5)$ $\qquad \delta_3 = (2 \times 0.2)^{1/(20+1)} - 1 = -0.04$

$$\delta = [0.01 \quad -0.07 \quad -0.04 \quad 0.04]$$

$$\delta = \begin{cases} (2r)^{1/(\eta_m+1)} - 1 & \text{如果} r < 0.5 \\ 1 - [2(1-r)]^{1/(\eta_m+1)} & \text{如果} r \geq 0.5 \end{cases}$$

变异

第二十一步：产生一个新的子代

$0 \le x_i \le 10 \qquad i = 1, \ 2, \ 3, \ 4$

$$Offspring_1 = [0.08 \quad 3.01 \quad 0.97 \quad 4.94] + ([10 \quad 10 \quad 10 \quad 10] - [0 \quad 0 \quad 0 \quad 0]) \times [0.01 \quad -0.07 \quad -0.04 \quad 0.04]$$

$$= [0.18 \quad 2.31 \quad 0.57 \quad 5.34]$$

$$\delta = [0.01 \quad -0.07 \quad -0.04 \quad 0.04]$$

$$O_1 = O_1 + (ub - lb)\delta$$

$$O = \begin{bmatrix} 0.08 & 3.01 & 0.97 & 4.94 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 1.98 & 1.07 & 4.06 & 9.02 \\ 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix} \Rightarrow O = \begin{bmatrix} 0.18 & 2.31 & 0.57 & 5.34 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 1.98 & 1.07 & 4.06 & 9.02 \\ 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix}$$

变异

第二十二步：选择第二个子代进行变异操作

$$Offspring_2 = \begin{bmatrix} 3.92 & 0 & 0.03 & 8.06 \end{bmatrix}$$

第二十三步：生成一个随机数决定是否执行变异操作

假设$r = 0.2$

$r = p_m \rightarrow$ 不执行变异操作

第二十四步：子代保持不变

$$O = \begin{bmatrix} 0.18 & 2.31 & 0.57 & 5.34 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 8 & 1 & 3 \\ 4 & 0 & 0 & 8 \\ 1.98 & 1.07 & 4.06 & 9.02 \\ 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix}$$

变异

第二十五步：对所有剩余的子代进行变异 $p_m = 0.2 \quad \eta_m = 20$

	Offspring	r	δ	New offspring
O_3	[5 8 1 3]	0.1	[0.5 0.1 0.6 0.3]	[5 7.3 1.1 2.8]
O_4	[4 0 0 8]	0.6	$r>p_m$不执行变异操作	[4 0 0 8]
O_5	[1.98 1.07 4.06 9.02]	0.3	$r>p_m$不执行变异操作	[1.98 1.07 4.06 9.02]
O_6	[0.02 2.93 0.94 4.98]	0.8	$r>p_m$不执行变异操作	[0.02 2.93 0.94 4.98]

变异

第二十五步：对所有剩余的子代进行变异 $p_m = 0.2 \quad \eta_m = 20$

	Offspring	r	δ	New offspring
O_3	[5 8 1 3]	0.1	[0.5 0.1 0.6 0.3]	[5 7.3 1.1 2.8]
O_4	[4 0 0 8]	0.6	$r>p_m$不执行变异操作	[4 0 0 8]
O_5	[1.98 1.07 4.06 9.02]	0.3	$r>p_m$不执行变异操作	[1.98 1.07 4.06 9.02]
O_6	[0.02 2.93 0.94 4.98]	0.8	$r>p_m$不执行变异操作	[0.02 2.93 0.94 4.98]

第二十六步：计算子代的适应度函数值

$$O = \begin{bmatrix} 0.18 & 2.31 & 0.57 & 5.34 \\ 3.92 & 0 & 0.03 & 8.06 \\ 5 & 7.3 & 1.1 & 2.8 \\ 4 & 0 & 0 & 8 \\ 1.98 & 1.07 & 4.06 & 9.02 \\ 0.02 & 2.93 & 0.94 & 4.98 \end{bmatrix} \qquad f_O = \begin{bmatrix} 34.21 \\ 80.33 \\ 87.34 \\ 80 \\ 102.91 \\ 34.27 \end{bmatrix}$$

幸存

第二十七步：合并所有的解选择最佳的$N_p(N_p = 6)$个解

$$P=\begin{bmatrix}4&0&0&8\\3&1&9&7\\0&3&1&5\\2&1&4&9\\6&2&8&3\\5&8&1&3\end{bmatrix}\qquad f=\begin{bmatrix}80\\140\\35\\102\\113\\99\end{bmatrix}$$

$$O=\begin{bmatrix}0.18&2.31&0.57&5.34\\3.92&0&0.03&8.06\\5&7.3&1.1&2.8\\4&0&0&8\\1.98&1.07&4.06&9.02\\0.02&2.93&0.94&4.98\end{bmatrix}\qquad f_O=\begin{bmatrix}34.21\\80.33\\87.34\\80\\102.91\\34.27\end{bmatrix}$$

下一代种群

$$P=\begin{bmatrix}0.18&2.31&0.57&5.34\\0.02&2.93&0.94&4.98\\0&3&1&5\\4&0&0&8\\4&0&0&8\\3.92&0&0.03&8.06\end{bmatrix}\qquad f=\begin{bmatrix}34.21\\34.27\\35\\80\\80\\80.33\end{bmatrix}$$

伪代码

```
Input : Fitness function, lb, ub, N_p, T, p_c, p_m, η_c  η_m  k
1.   Initialize a random population (P)
2.   Evaluate the objective function value(f) of P
     for t = 1 to T
       Perform tournament selection of tournament size, k
       for i = 1 to N_p / 2
         Randomly choose two parents
           if r < p_c
             Generate two offspring using SBX crossover
             Bound the offspring
           else
             Copy the selected parents as offspring
           end
       end
       for i = 1 to N_p
           if r < p_m
              Perform polynomial mutation of ith offspring
              Bound the mutated offspring
           else
              No change in ith offspring
           end
       end
     Evaluate the fitness
     Combine population and offspring to perform(μ + λ)
     end
```

实变量编码遗传算法

- 实变量编码遗传算法
- 举例说明实变量编码遗传算法的工作机理
- 实变量编码遗传算法的优点和局限

实变量编码遗传算法的优点和局限

优点：

- 适合于在遗传算法中表示范围较大的数
- 适用于精度要求较高的遗传算法
- 便于较大空间的遗传搜索
- 改善了遗传算法的计算复杂性，提高了运算效率
- 便于遗传算法与经典优化方法的混合使用
- 便于处理复杂的决策变量约束条件

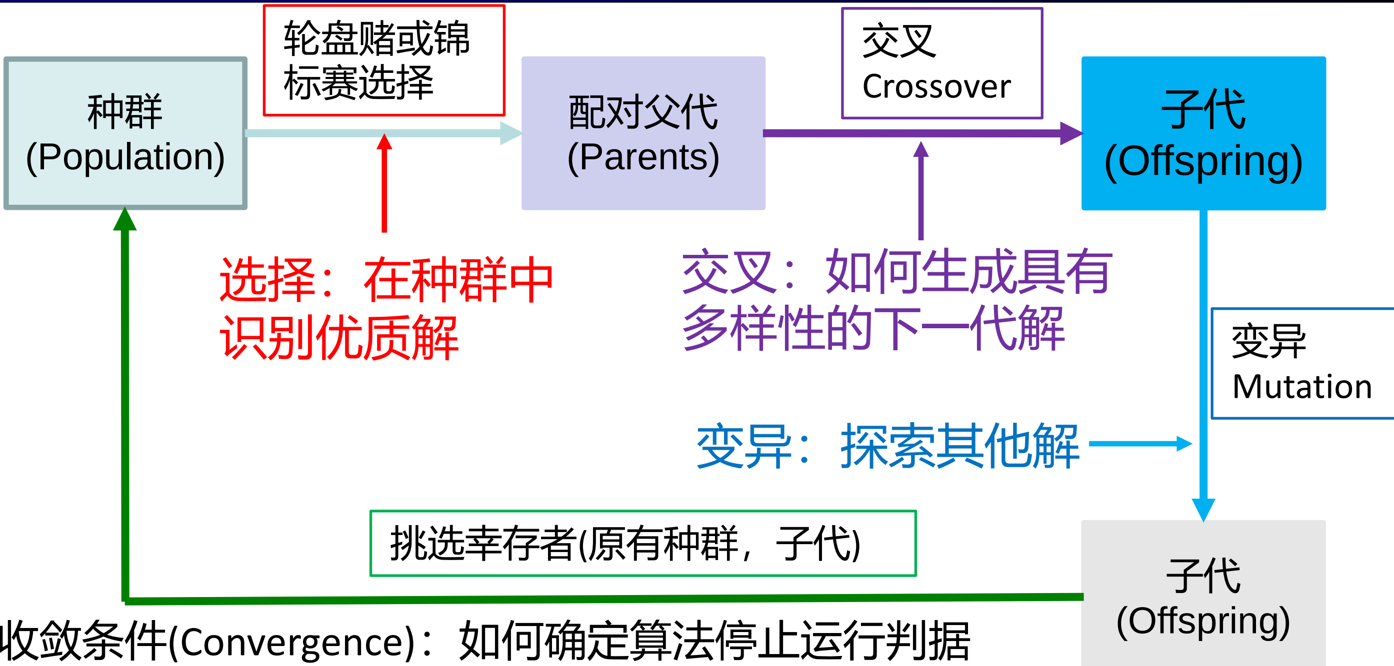
局限：

- 适用范围有限，只能用于连续变量问题

1. 二进制编码遗传算法（Binary encoding genetic algorithm）
2. 实变量编码遗传算法(Real-number encoding genetic algorithm)
3. 顺序编码遗传算法(Order encoding genetic algorithm)[次序编码、排序编码]

- 顺序编码遗传算法
- 举例说明顺序编码遗传算法的工作机理
- 遗传算法参数经验设置
- 遗传算法的优点和局限

遗传算法的基本流程



收敛条件(Convergence)：如何确定算法停止运行判据

遗传算法最大优点：我们不需要知道如何求解问题，我们只需要知道如何评价生成解的品质，通过选择、交叉和变异操作，我们就可以得到优质解

- ✓ 进化&选择过程对所有问题几乎一致
- ✓ 适应度函数&染色体设计：每个特定的优化问题不一样

顺序遗传算法中的交叉操作

- 二进制交叉技术不适用于顺序遗传算法

例如：旅行商问题，考虑两个顺序编码染色体

父代1：	5	7	2	8	1	6	3	4
父代2：	6	1	3	5	4	2	8	7

子代1：	5	7	2	8	1	2	8	7
子代2：	6	1	3	5	4	6	3	4

➢ <u>子代不是有效的染色体</u>

顺序编码遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉（Partial mapped crossover)
- 基于位置的交叉(Position based crossover)
- 边重组交叉(Edge recombination crossover)

单点排序交叉 (Single-point order crossover)

假设P_1和P_2是两个父代染色体, 染色体的长度为L

步骤：

1）随机产生一个交叉点K，满足$1 \le K \le L-1$

2）将父代染色体P_1在交叉点K左侧的片段，复制到C_1（初始为空），

将父代染色体P_2在交叉点K左侧的片段，复制到C_2（初始为空）

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7

子代1（C_1）：	5	7	2					
子代2（C_2）：	6	1	3					

单点排序交叉 (Single-point order crossover)

步骤：

3)对于C_1的右侧，从父代染色体P_2中复制，基因值出现的顺序保持不变，但已经出现在C_1的左侧的基因不进行拷贝

4）对于C_2的右侧，从父代染色体P_1中复制，基因值出现的顺序保持不变，但已经出现在C_2的左侧的基因不进行拷贝

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7

子代1（C_1）：	5	7	2					
子代2（C_2）：	6	1	3					

单点排序交叉 (Single-point order crossover)

步骤：

3)对于C_1的右侧，从父代染色体P_2中复制，基因值出现的顺序保持不变，但已经出现在C_1的左侧的基因不进行拷贝

4）对于C_2的右侧，从父代染色体P_1中复制，基因值出现的顺序保持不变，但已经出现在C_2的左侧的基因不进行拷贝

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7

子代1（C_1）：	5	7	2	6	1	3	4	8
子代2（C_2）：	6	1	3	5	7	2	8	4

排序遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉（Partial mapped crossover)
- 基于位置的交叉(Position based crossover)
- 边重组交叉(Edge recombination crossover)

两点排序交叉 (Two-point order crossover)

假设P_1和P_2是两个父代染色体, 染色体的长度为L

步骤：

1）随机产生两个交叉点K_1和K_2，满足$1 \le K_1,\ K_2 \le L-1$

2）将父代染色体P_1位于交叉点K_1和K_2之间的片段，复制到C_1（初始为空），将父代染色体P_2位于交叉点K_1和K_2之间的片段，复制到C_2（初始为空）

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7
子代1（C_1）：			2	8	1			
子代2（C_2）：			3	5	4			

两点排序交叉 (Two-point order crossover)

步骤：

3）子代C_1和C_2剩余的基因片段分别从P_2和P_1复制，基因顺序保持不变，已经在子代C_1和C_2中包含的基因不再复制

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7
子代1（C_1）：			2	8	1			
子代2（C_2）：			3	5	4			

两点排序交叉 (Two-point order crossover)

步骤：

3）子代C_1和C_2剩余的基因片段分别从P_2和P_1复制，基因顺序保持不变，已经在子代C_1和C_2中包含的基因不再复制

父代1（P_1）：	5	7	2	8	1	6	3	4
父代2（P_2）：	6	1	3	5	4	2	8	7
子代1（C_1）：	6	3	2	8	1	5	4	7
子代2（C_2）：	7	2	3	5	4	8	1	6

两点排序交叉的伪代码

procedure：OX 交叉

input：染色体 v_1，v_2，染色体长度 l

output：子代染色体 v'

```
begin
  w←1
    s←random[1:l−1];
    t←random[s+1:l];
    v′←v1[s:t];
    for i=1 to s−1
        for j=w to l
            fg←0;
            for k=s to t
                if v2[j]=v1[k] then
                    fg←1; break;
```

e.g. s=3, t=6

	s				t				
parent1	1	2	3	4	5	6	7	8	9
parent2	5	7	4	9	1	3	6	2	8
v′ proto-child			3	4	5	6			

标签f_g:父代2基因与选择的位置的基因相同为1，不相同的为0

1	0	1	0	0	1	1	0	0

其中，v_1是父代染色体 1，v_2是父代染色体 2，l 是染色体的长度，v'是子代染色体，w 为工作数据，f_g为标签，s 为子字符串的起始位置，t 为子字符串的终止位置。

两点排序交叉的伪代码

```
procedure：OX 交叉
input：染色体 v1，v2，染色体长度 l
output：子代染色体 v′
begin
  w←1
    s←random[1:l−1];
    t←random[s+1:l];
    v′←v1[s:t];
    for i=1 to s−1
      for j=w to l
        fg←0;
        for k=s to t
          if v2[j]=v1[k] then
            fg←1; break;
```

e.g. s=3, t=6

parent1	1	2	3	4	5	6	7	8	9
parent2	5	7	4	9	1	3	6	2	8
v′ proto-child			3	4	5	6			

标签f_g:父代2基因与选择的位置的基因相同为1，不相同的为0

1	0	1	0	0	1	1	0	0

其中，v_1是父代染色体 1，v_2是父代染色体 2，l 是染色体的长度，v'是子代染色体，w 为工作数据，f_g为标签，s 为子字符串的起始位置，t 为子字符串的终止位置。

两点排序交叉的伪代码

procedure：OX 交叉

input：染色体 v_1，v_2，染色体长度 l

output：子代染色体 v'

begin

$w \leftarrow 1$

$s \leftarrow \mathrm{random}[1:l-1]$;

$t \leftarrow \mathrm{random}[s+1:l]$;

$v' \leftarrow v_1[s:t]$;

for $i=1$ **to** $s-1$

for $j=w$ **to** l

$f_g \leftarrow 0$;

for $k=s$ **to** t

if $v_2[j]=v_1[k]$ **then**

$f_g \leftarrow 1$; break;

e.g. s=3, t=6

parent1	1	2	3	4	5	6	7	8	9
parent2	5	7	4	9	1	3	6	2	8
v' proto-child			3	4	5	6			

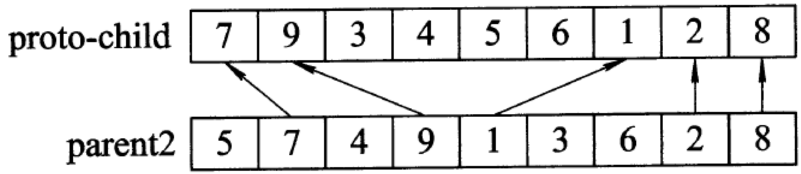
标签f_g:父代2基因与选择的位置的基因相同为1，不相同的为0

1	0	1	0	0	1	1	0	0

其中，v_1是父代染色体 1，v_2是父代染色体 2，l 是染色体的长度，v'是子代染色体，w 为工作数据，f_g为标签，s 为子字符串的起始位置，t 为子字符串的终止位置。

两点排序交叉的伪代码

```
        if f_g = 0 then
            v'[i] ← v2[j]
            w ← j+1; break;
    for i = t+1 to l
        for j = w to l
            f_g ← 0;
            for k = s to t
                if v2[j] = v1[k] then
                    f_g ← 1; break;
            if f_g = 0 then
                v'[i] ← v2[j];
                w ← j+1; break;
    output 子代染色体 v';
end
```



标签f_g:								
1	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**

依次遍历父代2中所有基因位置，将标签f_g为0的基因拷贝到子代中的第1到s-1位置以及t+1到l位置

其中，v_1是父代染色体1，v_2是父代染色体2，l是染色体的长度，v'是子代染色体，w为工作数据，f_g为标签，s为子字符串的起始位置，t为子字符串的终止位置。

排序遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉（Partial mapped crossover)
- 基于位置的交叉(Position based crossover)
- 边重组交叉(Edge recombination crossover)

部分映射交叉（Partial mapped crossover)

1985年，*Goldberg*等针对*TSP*提出了部分映射交叉操作。

假设P_1和P_2是两个父代染色体, 染色体的长度为L

步骤：

1）先根据均匀随机分布产生父代个体的两个交叉点，定义这两点之间的区域为一匹配区域

2）使用位置交叉操作交换两个父代个体的匹配区域

3）对两子串匹配区域以外出现的遍历重复，依据匹配区域内的位置映射关系，逐一进行交换，此时得到的两个子代个体所表示的路径为有效路径

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9
子代1：					7	5	6			
子代2：					8	1	5			

映射关系7↔8， 1↔5， 5↔6

- 匹配区域包含1,5,6,7,8
- 不在匹配区域内的是2,3,4,9,10, 这些城市码子代解从父代解中直接继承

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9
子代1：	2		4	3	7	5	6	10		9
子代2：		2		10	8	1	5	4	3	9

映射关系7↔8， 1↔5， 5↔6

- 匹配区域包含1,5,6,7,8
- 不在匹配区域内的是2,3,4,9,10, 这些城市码子代解从父代解中直接继承

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9
子代1：	2	1	4	3	7	5	6	10		9
子代2：		2		10	8	1	5	4	3	9

映射关系7↔8， 1↔5， 5↔6

6↔5 ↔1

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9

子代1：	2	1	4	3	7	5	6	10	8	9
子代2：		2		10	8	1	5	4	3	9

映射关系7⟷8， 1⟷5， 5⟷6

7⟷8

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9
子代1：	2	1	4	3	7	5	6	10	8	9
子代2：	7	2		10	8	1	5	4	3	9

映射关系7↔8， 1↔5， 5↔6

8↔7

部分映射交叉（Partial mapped crossover)

父代1：	2	6	4	3	8	1	5	10	7	9
父代2：	8	2	1	10	7	5	6	4	3	9
子代1：	2	1	4	3	7	5	6	10	8	9
子代2：	7	2	6	10	8	1	5	4	3	9

映射关系7↔8， 1↔5， 5↔6

1↔5 ↔6

部分映射交叉伪代码

（1）部分映射交叉 PMX。首先给出 PMX 交叉算子的运算过程。

procedure：PMX 交叉

input：染色体 $\mathbf{v}_1$，$\mathbf{v}_2$，染色体长度 l

output：子代染色体 $\mathbf{v}_1'$，$\mathbf{v}_2'$

begin

$R \leftarrow \phi$

$s \leftarrow \mathrm{random}[1:l-1]$；

$t \leftarrow \mathrm{random}[s+1:l]$；

$\mathbf{v}_1' \leftarrow \mathbf{v}_1[1:s-1]//\mathbf{v}_2[s:t]//\mathbf{v}_1[t+1, l]$；

$\mathbf{v}_2' \leftarrow \mathbf{v}_2[1:s-1]//\mathbf{v}_1[s:t]//\mathbf{v}_2[t+1, l]$；

$R \leftarrow \mathrm{relation}(\mathbf{v}_1[s:t], \mathbf{v}_2[s:t])$；

$\mathrm{legalize}(\mathbf{v}_1', \mathbf{v}_2', R)$；

output 子代 $\mathbf{v}_1'$，$\mathbf{v}_2'$；

end

其中，v_1是父代染色体 1，v_2是父代染色体 2，l 是染色体的长度，v_1'是子代染色体 1，v_2'是子代染色体 2，R 为两个父代染色体之间的映射关系，s 为涉及交叉操作的子字符串的起始位置，t 为涉及交叉操作的子字符串的终止位置，其中 relation(v_1，v_2)用于搜索 v_1 和 v_2 之间的关系，legalize(v_1，v_2，R)则基于映射关系 R 改变 v_1 和 v_2 的基因值。

排序遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉 （Partial mapped crossover)
- 基于位置的交叉(Position based crossover)
- 边重组交叉(Edge recombination crossover)

基于位置的交叉(Position based crossover)

假设P_1和P_2是两个父代染色体, 染色体的长度为L

步骤:

1）随机选择n个交叉点，j_1，j_2,. . .j_n, 其中n << L

2）将P_1的第j_1^{th}，j_2^{th},. . .j_n^{th}基因复制到子代C_1和C_2的相同位置

		j_1 ↓		j_2 ↓			j_3 ↓	
父代1:	5	7	2	8	1	6	3	4
父代2:	6	1	3	5	4	2	8	7
子代1:		7		8			3	
子代2:		1		5			8	

基于位置的交叉(Position based crossover)

3)子代C_1剩余位置的基因从P_2按照顺序复制，已经在子代C_1中出现的基因不再复制

4）子代C_2剩余位置的基因从P_1按照顺序复制，已经在子代C_2中出现的基因不再复制

		j_1		j_2			j_3	
父代1:	5	7	2	8	1	6	3	4
父代2:	6	1	3	5	4	2	8	7
子代1:		7		8			3	
子代2:		1		5			8	

基于位置的交叉(Position based crossover)

3)子代C_1剩余位置的基因从P_2按照顺序复制，已经在子代C_1中出现的基因不再复制

4）子代C_2剩余位置的基因从P_1按照顺序复制，已经在子代C_2中出现的基因不再复制

		j_1		j_2			j_3	
父代1:	5	7	2	8	1	6	3	4
父代2:	6	1	3	5	4	2	8	7
子代1:	6	7	1	8	5	4	3	2
子代2:	7	1	2	5	6	3	8	4

基于位置的交叉伪代码

procedure：PBX 交叉

input：染色体 v_1，v_2，染色体长度 l

output：子代染色体 v'

begin

$S \leftarrow \phi$，$T \leftarrow \phi$，$w \leftarrow 1$；

$N \leftarrow \mathrm{random}[1:l]$；

for $i=1$ **to** N

$j \leftarrow \mathrm{random}[1:l]$；

$v'[j] \leftarrow v_1[j]$；

$T \leftarrow T \cup j$；

$S \leftarrow S \cup v_1[j]$；

for $i=1$ **to** l

$f_{g1} \leftarrow 0$；

for $j=1$ **to** N

if $i=t[j]$ **then** $f_{g1} \leftarrow 1$；

N是随机选择位置的数量 .e.g 4

	j			j	j			j
parent1: 1	2	3	4	5	6	7	8	9

v' proto-child

	2			5	6			9

$T = \{2,5,6,9\}$位置合集

$S = \{2,5,6,9\}$基因合集

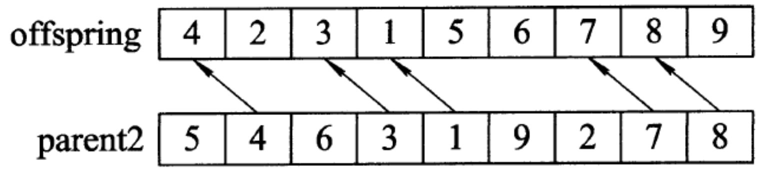
标签f_{g1}:选择的位置为1，没选择的为0

0	1	0	0	1	1	0	0	1

基于位置的交叉伪代码

```
    v′[j]←v₁[j] ;
    T←T∪j ;
    S←S∪v₁[j] ;
for i=1 to l
    f_g1←0;
    for j =1 to N
        if i=t[j] then f_g1←1;
    if f_g1=1 then continue;
    for k=w to l
        f_g2←0;
        for m=1 to N
            if v₂[k]=s[m] then
                f_g2←1; break;
        if f_g2=0 then
            v′[i]←v₂[k ];
            w←k +1 ; break ;
output 子代染色体 v′;
end
```

其中，v_1 是父代染色体 1，v_2 是父代染色体 2，l 是染色体的长度，v' 是子代染色体，N 为选择的位置总数，$T=\{t[j]\}$，$j=1,2,\cdots,N$ 为选择的位置集合，$S=\{s[m]\}$，$m=1,2,\cdots,N$ 为选择位置的基因值集合，f_{g1} 为标签 1，f_{g2} 为标签 2，w 为工作数据。



标签f_{g2}:父代2中与父代1中选择位置基因相同的位置为1，不相同的位置为0

1	0	1	0	0	1	1	0	0

依次遍历父代2中所有基因位置，将标签f_{g2}为0的基因拷贝到子代中的第i个位置（标签f_{g1}为0的位置）

排序遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉（Partial mapped crossover)
- 基于位置的交叉(Position based crossover)
- 边重组交叉(Edge recombination crossover)

边重组交叉(Edge recombination crossover)

- 前面介绍的TSP交叉操作基本上考虑的是城市的位置和顺序，未考虑城市间的连接
- Grefenstette认为遗传算法应用于TSP，其遗传操作不仅要考虑城市的位置，而且有必要考虑城市间的关系，城市间的关系定义为边，让子个体继承父个体中边的信息，设计围绕边的遗传操作很有意义
- 1989年，Whitley等提出了一种被称为边重组交叉操作，使子个体能够从父个体继承边95%~99%的信息
- 边重组操作根据继承两个父个体定义的旅程中城市间的相邻状况生成子个体

边重组交叉(Edge recombination crossover)

- 例如一条路径

3	1	2	8	7	4	6	9	5

可以定义边有
(3, 1) (1, 2) (2, 8) (8, 7) (7, 4) (4, 6) (6, 9) (9, 5) (5, 3)

- TSP中，最小化的目标函数是由合法路径中所有边的总和构成的，从这个角度考虑，路径中城市的位置不是特别重要

- 因此，边重组操作中，对于每一城市，将任意一父个体中与之邻接的其他城市列出构成列表，然后利用两个父个体对应的路径中的边表来设计交叉操作

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

根据两个父代染色体生成每一个城市的边表，其中两个父个体中均有的边，数字打上标记“”，边重组操作中产生子个体时优先考虑打有标记的城市码

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 “2” 4
2	~~“1”~~ 3 8
3	2 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 ~~1~~ 6 3

父代染色体为P_1和P_2，子代染色体为C_1（初始为空）

步骤：

1）从父代染色体P_1的初始城市（X）开始

2）将X复制到子代染色体C_1

3)从边表右侧列表中删除所有的X

开始城市1

1								

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	9 “2” 4
2	~~“1”~~ 3 8
3	2 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

下一城市选择2									
	1	2							

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 8
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 ~~2~~
9	8 ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

下一城市选择2

1	2							

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 8
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 ~~2~~
9	8 ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市2

下一城市在3和8之间选择，8的连接更少，选择8

1	2	8						

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” ~~“8”~~
8	“7” 9 ~~2~~
9	~~8~~ ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市2

下一城市在3和8之间选择，8的连接更少，选择8

1	2	8						

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~"2"~~ 4
2	~~"1"~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 "5" ~~1~~
5	"4" 6 3
6	5 "7" 9
7	"6" ~~"8"~~
8	"7" 9 ~~2~~
9	~~8~~ ~~1~~ 6 3

步骤:

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市2
下一城市在3和8之间选择，8的连接更少，选择8

1	2	8						

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 “7” 9
7	“6” ~~“8”~~
8	“7” 9 ~~2~~
9	~~8~~ ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市8

下一城市优先选择7

1	2	8	7					

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 ~~“7”~~ 9
7	“6” ~~“8”~~
8	~~“7”~~ 9 ~~2~~
9	~~8~~ ~~1~~ 6 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市8

下一城市优先选择7

1	2	8	7					

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” 6 3
6	5 ~~“7”~~ 9
7	“6” ~~“8”~~
8	~~“7”~~ 9 ~~2~~
9	~~8~~ ~~1~~ 6 3

步骤:

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市7

下一城市选择6

1	2	8	7	6				

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” ~~6~~ 3
6	5 ~~“7”~~ 9
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ 9 ~~2~~
9	~~8~~ ~~1~~ ~~6~~ 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市7
下一城市选择6

1	2	8	7	6				

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	9 ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 9
4	3 “5” ~~1~~
5	“4” ~~6~~ 3
6	5 ~~“7”~~ 9
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ 9 ~~2~~
9	~~8~~ ~~1~~ ~~6~~ 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市6

下一城市在5和9之间选择，9的连接更少，选择9

1	2	8	7	6	9			

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ ~~“2”~~ 4
2	~~“1”~~ 3 ~~8~~
3	~~2~~ 4 5 ~~9~~
4	3 “5” ~~1~~
5	“4” ~~6~~ 3
6	5 ~~“7”~~ ~~9~~
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ 3

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市6

下一城市在5和9之间选择，9的连接更少，选择9

1	2	8	7	6	9			

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

步骤：

4）从与城市*X*相邻的城市列表中选择下一个城市*Y*，优先选择具有“”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

城市	连接
1	~~9~~ “~~2~~” 4
2	“~~1~~” 3 ~~8~~
3	~~2~~ 4 5 ~~9~~
4	3 “5” ~~1~~
5	“4” ~~6~~ 3
6	5 “~~7~~” ~~9~~
7	“~~6~~” “~~8~~”
8	“~~7~~” ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ 3

当前城市9
下一城市选择3

1	2	8	7	6	9	3		

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ ~~"2"~~ 4
2	~~"1"~~ ~~3~~ ~~8~~
3	~~2~~ 4 5 ~~9~~
4	~~3~~ "5" ~~1~~
5	"4" ~~6~~ ~~3~~
6	5 ~~"7"~~ ~~9~~
7	~~"6"~~ ~~"8"~~
8	~~"7"~~ ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ ~~3~~

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市9
下一城市选择3

1	2	8	7	6	9	3		

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ ~~“2”~~ 4
2	~~“1”~~ ~~3~~ ~~8~~
3	~~2~~ 4 5 ~~9~~
4	~~3~~ “5” ~~1~~
5	“4” ~~6~~ ~~3~~
6	5 ~~“7”~~ ~~9~~
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ ~~3~~

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市3

下一城市在4和5之间选择，4和5的连接数相同，随机选择一个，假设选择5

1	2	8	7	6	9	3	5	

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ ~~“2”~~ 4
2	~~“1”~~ ~~3~~ ~~8~~
3	~~2~~ 4 ~~5~~ ~~9~~
4	~~3~~ ~~“5”~~ ~~1~~
5	“4” ~~6~~ ~~3~~
6	~~5~~ ~~“7”~~ ~~9~~
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ ~~3~~

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市3
下一城市在4和5之间选择，4和5的连接数相同，随机选择一个，假设选择5

1	2	8	7	6	9	3	5	

边重组交叉(Edge recombination crossover)

父代1：	1	2	3	4	5	6	7	8	9
父代2：	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ ~~“2”~~ 4
2	~~“1”~~ ~~3~~ ~~8~~
3	~~2~~ 4 ~~5~~ ~~9~~
4	~~3~~ ~~“5”~~ ~~1~~
5	“4” ~~6~~ ~~3~~
6	~~5~~ ~~“7”~~ ~~9~~
7	~~“6”~~ ~~“8”~~
8	~~“7”~~ ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ ~~3~~

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市5
下一城市选择4

1	2	8	7	6	9	3	5	4

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	~~9~~ “~~2~~” 4
2	“~~1~~” ~~3~~ ~~8~~
3	~~2~~ 4 ~~5~~ ~~9~~
4	~~3~~ “~~5~~” ~~1~~
5	“4” ~~6~~ ~~3~~
6	~~5~~ “~~7~~” ~~9~~
7	“~~6~~” “~~8~~”
8	“~~7~~” ~~9~~ ~~2~~
9	~~8~~ ~~1~~ ~~6~~ ~~3~~

步骤：

4）从与城市X相邻的城市列表中选择下一个城市Y，优先选择具有“ ”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

当前城市4，得到C1

同样可以得到C2

C1:	1	2	8	7	6	9	3	5	4