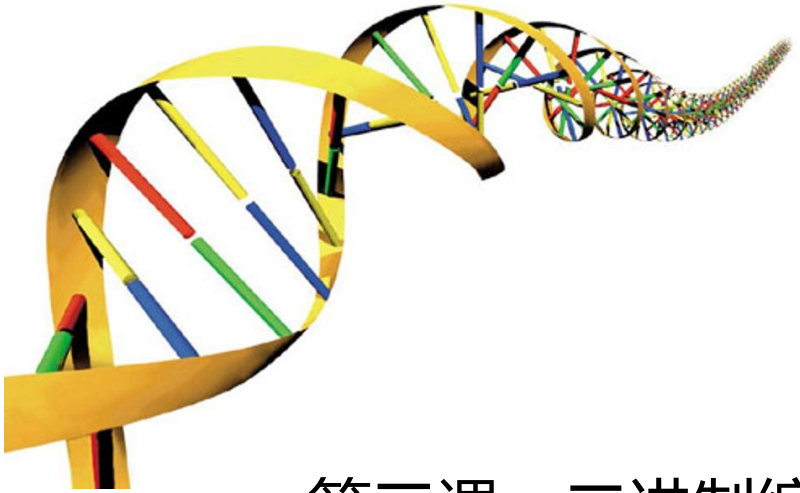
进化优化算法

基于仿生和种群的计算机智能方法



第三课：二进制编码遗传算法

第一次作业

- 作业内容：示例2背包问题
- 交作业时间：2024年10月21日之前
- 作业要求：
  1. 编程语言不限
  2. 电子版作业报告一份：包含问题描述、编程思路及方法、源代码（添加适当的注释）
  3. 源代码一份，将在实验室50次运行测试程序效率及准确性
  4. 程序自己编写，不要拷贝或抄袭现有的程序
  5. 每人交一份作业

第一次作业

j	1	2	3	4	5	6	7	8	9	10
价值c_j	91	72	90	46	55	8	35	75	61	15
重量w_j	84	83	43	4	44	6	82	92	25	83

j	11	12	13	14	15	16	17	18	19	20
价值c_j	77	40	63	75	29	75	17	78	40	44
重量w_j	56	18	58	14	48	70	96	32	68	92

限重 W=879

答案：总价值1025，总重量871

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)

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

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

二进制编码和解码

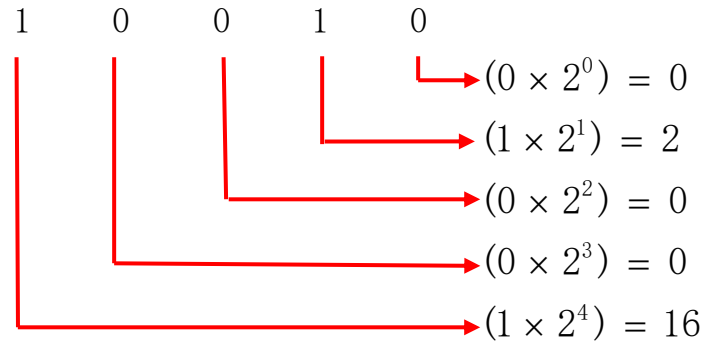
实变量编码成二进制变量（0和1）

如果一个二进制编码具有n位，那么在变量的取值范围内有2^n个不同的解

假设用一个5位二进制编码来代表变量x，其中x的取值范围$[x_{\min}, x_{\max}]$

二进制编码[0 0 0 0 0]映射为$x_{\min}$，二进制编码[1 1 1 1 1]映射为$x_{\max}$

DV : *decoded value*



二进制编码可以通过下式解码

$$DV = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0$$

其中二进制编码为$[b_{n-1} \quad b_{n-2} \quad \ldots \quad b_1 \quad b_0]$

最大值对应于（$2^n - 1$），最小值对应于0

二进制编码和解码

实变量编码成二进制变量（0和1）

如果一个二进制编码具有n位，那么在变量的取值范围内有2^n个不同的解

如果二进制编码有n位,取值范围$[x_{\min}, x_{\max}]$的变量x可以写成

$$x = x_{\min} + \left(\frac{x_{\max} - x_{\min}}{2^n - 1}\right) DV \qquad \text{精度} = \left(\frac{x_{\max} - x_{\min}}{2^n - 1}\right)$$

例：假设用一个4位二进制编码来代表变量x，其中x的取值范围[5, 30]

因此在5和30之间共有16（$= 2^4$）个离散值

对应于二进制编码$s = [0 \quad 1 \quad 1 \quad 0]$, $\Rightarrow DV = (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6$

对应于$DV = 6$

$$x = 5 + \left(\frac{30 - 5}{2^4 - 1}\right) \times 6 = 15$$

$$\text{精度} = \left(\frac{30 - 5}{2^4 - 1}\right) = 1.67$$

意味着15的下一个数是16.67，无法得到15与16.67之间的数值

若想提高精度，需要增大编码位数n

二进制编码转换为实数

$x_{\min} = 5$ $x_{\max} = 30$

$$x = x_{\min} + \left(\frac{x_{\max} - x_{\min}}{2^n - 1}\right)DV$$

$n = 3$

精度 $= \left(\frac{30-5}{2^3-1}\right) = 3.57$

二进制	解码	实际值
000	0	5
001	1	8.57
010	2	12.14
011	3	15.71
100	4	19.29
101	5	22.86
110	6	26.43
111	7	30

$n = 4$

精度 $= \left(\frac{30-5}{2^4-1}\right) = 1.67$

二进制	解码	实际值	二进制	解码	实际值
0000	0	5	1000	8	18.33
0001	1	6.67	1001	9	20
0010	2	8.33	1010	10	21.67
0011	3	10	1011	11	23.33
0100	4	11.67	1100	12	25
0101	5	13.33	1101	13	26.67
0110	6	15	1110	14	28.33
0111	7	16.67	1111	15	30

例题

变量x的定义域为[-2, 5]，要求精度不低于10^{-6}

$x_{\min} = -2$ $x_{\max} = 5$

$$\frac{x_{\max} - x_{\min}}{10^{-6}} = \frac{5 - (-2)}{10^{-6}} = 7000000$$

需要将[-2, 5]分成至少7000000个等长的小区域

$4{,}194{,}304 = 2^{22} < 7000000 < 2^{23} = 8{,}388{,}608$

23位的二进制数（$b_{22}b_{21}...b_0$）都对应着0到8,388,608中的一个数

将二进制数（$b_{22}b_{21}...b_0$）转化为十进制整数，先将二进制解码

$$DV = \sum_{i=0}^{22} b_i \times 2^i$$

计算对应变量x的值：

$$x = -2 + \frac{5 - (-2)}{2^{23} - 1} DV$$

这样就把[-2, 5]范围内的数进行了二进制编码而且精度高于10^{-6}

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

种群的初始化

- ✓ 假设种群大小N_p=6，问题的维度为D=2
- ✓ 用5位二进制编码代表变量（n=5）
- ✓ 种群矩阵的尺寸为$N_p \times nD$　(6×10)
- ✓ 初始种群由0或1随机生成

$$p = \begin{bmatrix} \overbrace{0\ 0\ 0\ 1\ 0}^{X_1} & \overbrace{1\ 1\ 1\ 0\ 0}^{X_2} \\ 1\ 0\ 0\ 1\ 1 & 1\ 1\ 0\ 1\ 0 \\ 1\ 0\ 0\ 0\ 0 & 0\ 1\ 0\ 1\ 0 \\ 1\ 0\ 0\ 1\ 1 & 1\ 1\ 1\ 0\ 0 \\ 0\ 0\ 1\ 1\ 1 & 0\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 1 & 0\ 1\ 0\ 1\ 0 \end{bmatrix}$$

种群的初始化

初始种群的产生：遗传算法中初始种群中的个体可以是随机产生的，但最好采用如下策略设定

- 根据问题固有知识，设法把握最优解所占空间在整个问题空间中所占的范围，然后在此范围内设定初始种群

- 先随机产生一定数目的个体，然后从中挑选最好的个体加入到初始种群中。这个过程不断迭代，直到初始种群中个体的数目达到预先设定的规模

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

适应度函数评价

如何评价一个个体的适应度（fitness）？

- 一个简单的策略是以目标函数值作为评价标准
  - ✓ 对于只需要考虑单一目标函数的问题，比较简单
  - ✓ 对于两个或以上的目标函数，需要不同的处理
    - 不同目标函数可能具有不同的等级
    - 在目标函数的适应度计算上具有不同的重要性

适应度函数评价

- 在遗传算法中，需要产生下一代种群
  - ✓ 下一代种群应该向全局最优解靠近
  - ✓ 随机产生种群不是好的策略
  - ✓ 更好的策略是模仿生物学过程：选择（Selection）
- 选择（Selection）包括
  - ✓ 最优者幸存
  - ✓ 生存竞争
- 适应度函数评价是评价当前种群中每一个个体的生命力

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

遗传算法的选择机制

- 在完成编码操作之后，遗传算法的一个重要操作是如何在种群中选择（Selection）父代，并生成子代

- 选择（Selection）操作的目的是重视种群中适应性最强的个体，并且希望它们可以生成适应性更好的子代
  - ✓ 生成配对池(Creating a mating pool)
  - ✓ 选择一对解(Select a pair)
  - ✓ 再生/繁殖（Reproduce）

- 选择操作降低了种群的多样性

常用的选择机制

常用的选择机制：

- 轮盘赌选择（Roulette Wheel Selection）
- 随机遍历选择（Stochastic Universal Selection）
- 基于排名选择（Rank-based Selection）
- 锦标赛选择（Tournament Selection）
- 种马进化算法(Stud Evolutionary Algorithms)

轮盘赌选择（Roulette Wheel Selection）

- 基本思想是基于概率的随机选择：一个个体被选择进配对池的概率正比于这个个体的适应度值
- 通过轮盘选择的方式完成

个体 1: 适应度 = 10, 个体 3: 适应度 = 30,
个体 2: 适应度 = 20, 个体 4: 适应度 = 40.

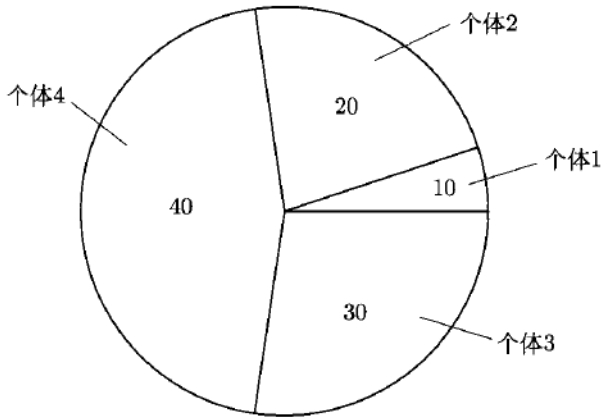


图 3.2 上面的饼图说明有 4 个成员的种群的遗传算法的轮盘赌选择. 每一个个体按其适应度分得一片. 每一个个体被选作父代的概率与它在轮盘中的片成比例.

轮盘赌选择（Roulette Wheel Selection）

- 轮盘的表面面积被分为N部分，每一部分的大小正比于每个个体的适应度值f_1，f_2，f_3，f_4，....f_N
- 轮盘沿着某一方向（顺时针或逆时针）旋转，一个固定指针停下时选择获胜区域
- 轮盘上第i个区域的面积代表第i个解被选择的概率，

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

- 具有较高适应度值的解可能被选择多次
- 轮盘转N_p次，每次选择一个解

个体 1: 适应度 = 10, 个体 3: 适应度 = 30,
个体 2: 适应度 = 20, 个体 4: 适应度 = 40.

$$p_1 = 0.1$$
$$p_2 = 0.2$$
$$p_3 = 0.3$$
$$p_4 = 0.4$$

轮盘赌选择（Roulette Wheel Selection）

算法 3.1 利用图 3.2 中的轮盘赌选择一个父代.

生成一个服从均匀分布的随机数 $r \in [0, 1]$

If $r < 0.1$ then

    Parent = 个体 1

else if $r < 0.3$ then

    Parent = 个体 2

else if $r < 0.6$ then

    Parent = 个体 3

else

    Parent = 个体 4

End if

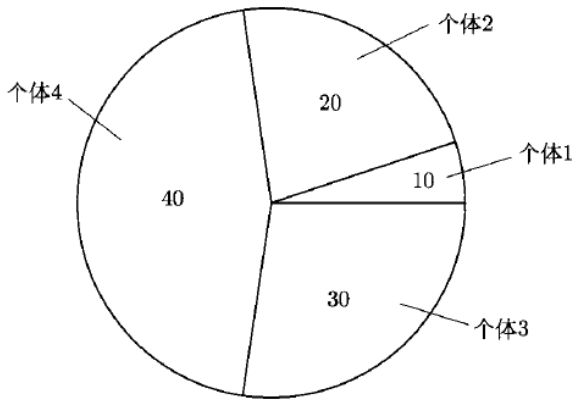


图 3.2 上面的饼图说明有 4 个成员的种群的遗传算法的轮盘赌选择. 每一个个体按其适应度分得一片. 每一个个体被选作父代的概率与它在轮盘中的片成比例.

轮盘赌选择（RWS）伪代码

```
/* Roulette wheel selection
* 输出参数：
* 选中的染色体
*/
procedure RWS
1   m ← 0;
2   r ← Random(0, 1);
3   for i = 1 to N
4       m ← m + p_i;
5       if r <= m
6           return i;
7       end if
8   end for
end procedure
```

个体 1: 适应度 = 10, 个体 3: 适应度 = 30,
个体 2: 适应度 = 20, 个体 4: 适应度 = 40.

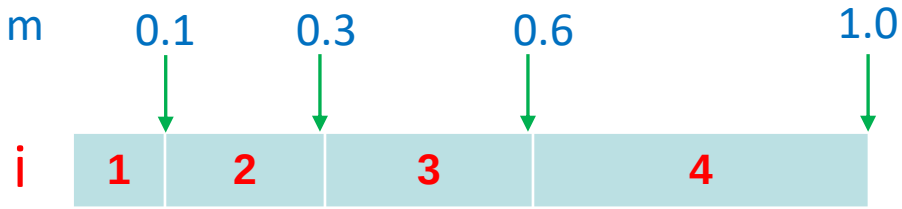
$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$

0至1的随机数



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4        m ← m + p_i;
5        if  r <= m
6             return  i;
7        end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

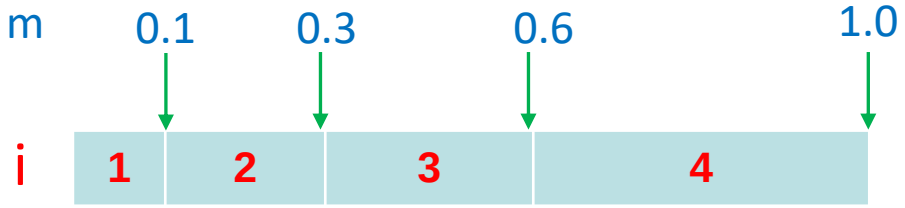
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4        m ← m + p_i;
5        if  r <= m
6             return  i;
7        end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

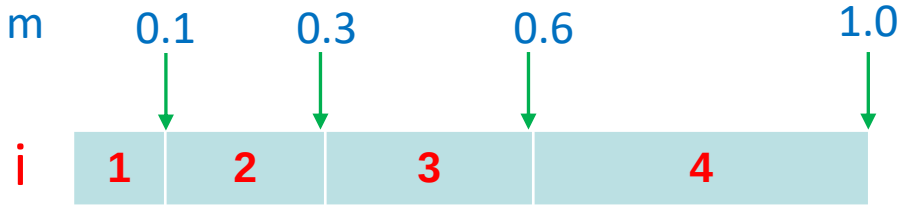
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

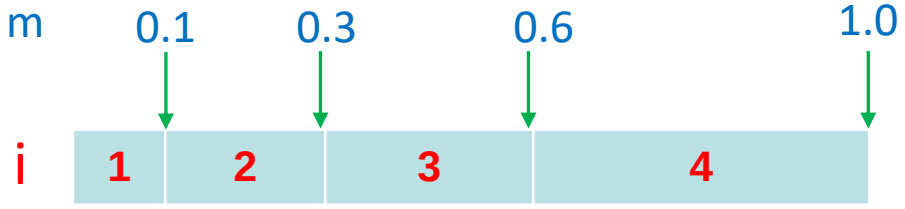
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

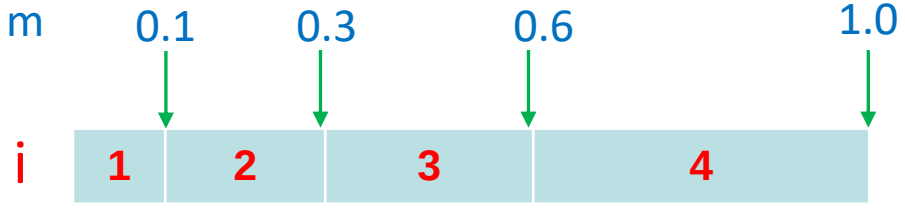
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

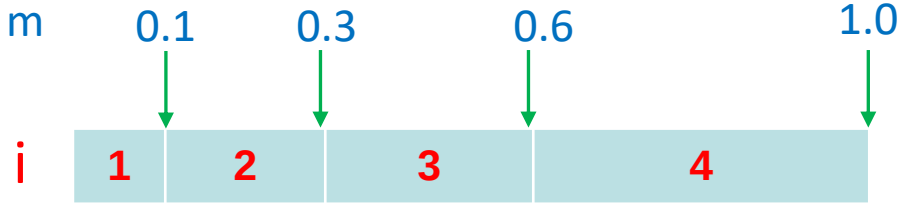
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

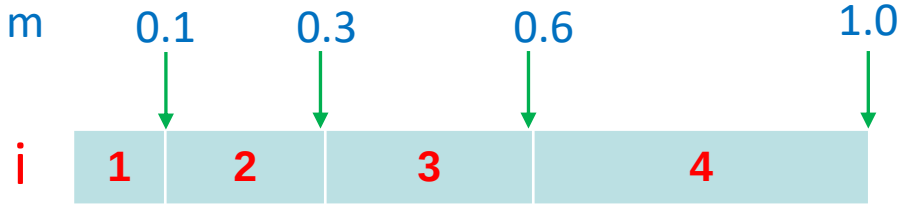
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

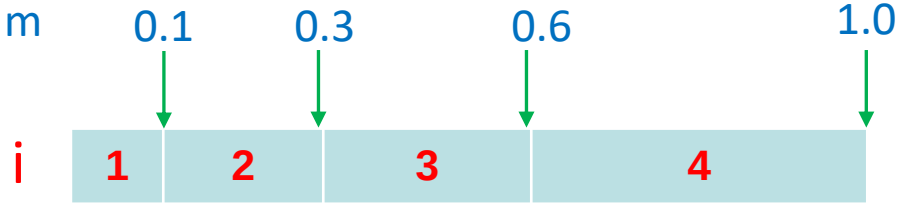
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4        m ← m + p_i;
5        if  r <= m
6             return  i;
7        end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

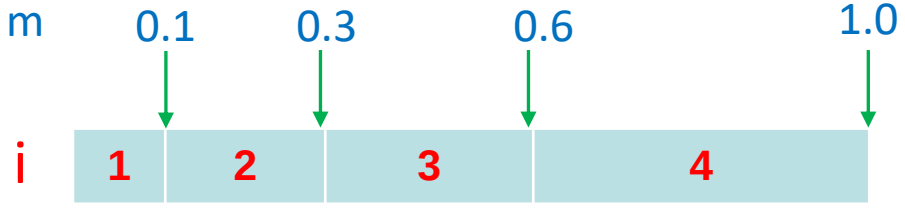
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

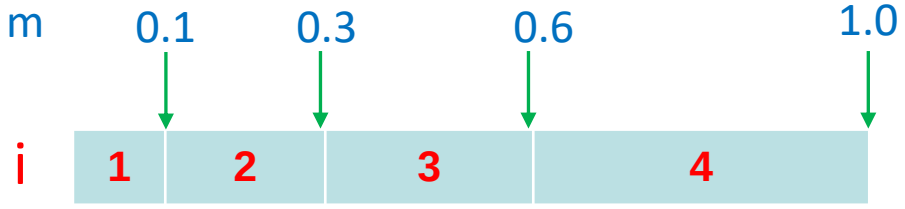
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure RWS
1   m ← 0;
2   r ← Random(0, 1);
3   for i = 1 to N
4       m ← m + p_i;
5       if r <= m
6           return i;
7       end if
8   end for
end procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

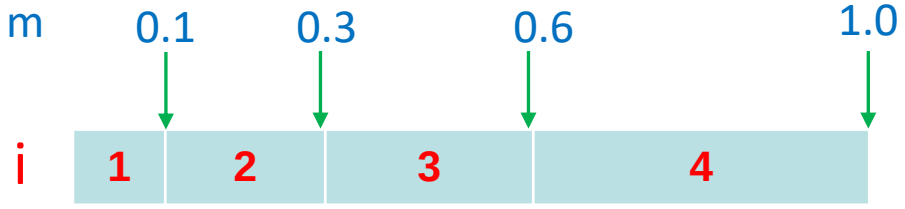
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4         m ← m + p_i;
5         if  r <= m
6               return  i;
7         end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

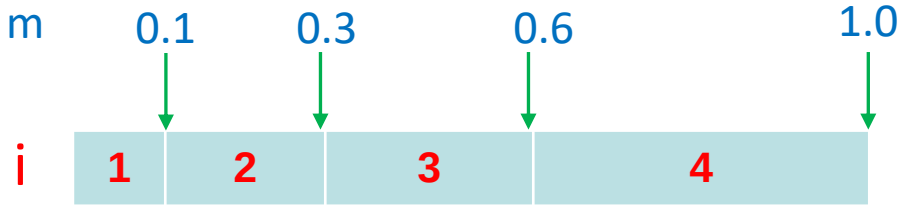
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择（RWS）伪代码

```
procedure  RWS
1   m ← 0;
2   r ← Random(0,  1);
3   for  i = 1  to  N
4        m ← m + p_i;
5        if  r <= m
6             return  i;
7        end  if
8   end  for
end  procedure
```

$m = 0$

假设$r = 0.5$

$i = 1$

$m = 0 + p_1 = 0.1$

不满足$r \le m$

$i = 2$

$m = 0.1 + p_2 = 0.3$

不满足$r \le m$

$i = 3$

$m = 0.3 + p_3 = 0.6$

满足$r \le m$

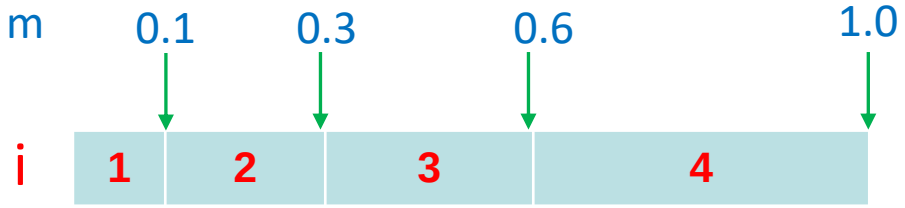
选择$i = 3$

$p_1 = 0.1$

$p_2 = 0.2$

$p_3 = 0.3$

$p_4 = 0.4$



轮盘赌选择的局限

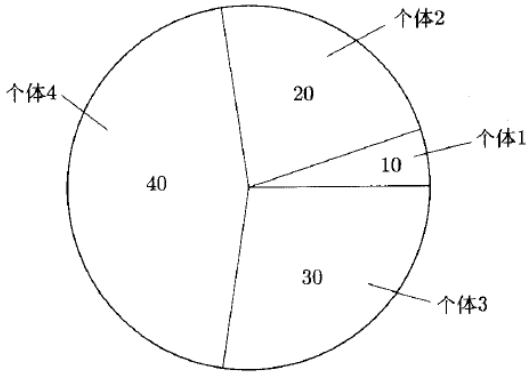
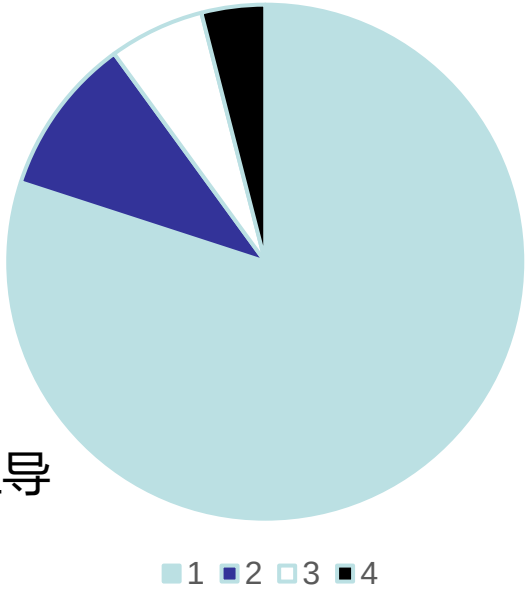


图 8.10　有 4 个个体的种群的轮盘赌选择的说明. 每一个个体按其适应度分得一片. 每一个个体被选作父代的可能与它在轮盘中片的面积成比例.

- 另一个潜在问题是有很大可能漏掉最好的个体
- 例如，假设旋转图8.10的轮盘4次，选择4个父代来重组。个体#4最好但在4次选择中都没被选中的概率等于$(0.6)^4$=13%。最好的个体不被选中的机会大约有1/7。失去种群中最好个体的信息的概率高达1/7，这让人难以接受
- 采用随机遍历选择克服这一局限

轮盘赌选择的局限

个体（i）	适应度函数值	轮盘（面积）
1	0.4	80%
2	0.05	10%
3	0.03	6%
4	0.02	4%



- 适应度函数值特别高的“超常”个体占主导地位，这些超常个体会因竞争力突出，其他解被选入配对池的概率极小
- 这容易导致多样性较少，减少了探索其他解的机会，造成不成熟收敛或者过早收敛到局部最优解，影响到算法的全局优化性能
- 采用基于排名选择克服这一局限

常用的选择机制

常用的选择机制：

- 轮盘赌选择（Roulette Wheel Selection）
- 随机遍历选择（Stochastic Universal Selection）
- 基于排名选择（Rank-based Selection）
- 锦标赛选择（Tournament Selection）
- 种马进化算法(Stud Evolutionary Algorithms)

随机遍历选择（Stochastic Universal Selection）

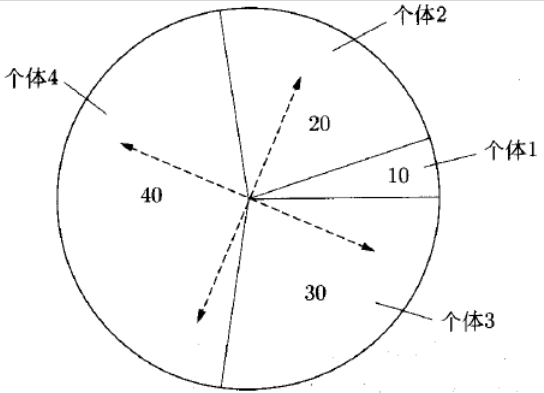


图 8.11　有 4 个个体的种群的随机遍历采样. 每一个个体按其适应度分得一片. 采用带有均匀分布的 4 个指针的旋转器, 旋转一次就得到 4 个父代.

- 为了克服轮盘赌选择可能漏掉最好的个体的局限，提出了随机遍历选择机制
- 随机遍历选择仍然采用轮盘赌方法，但它能解决这个问题。采用带有均匀分布的4个(N个)指针的旋转器，将它放在轮盘上并旋转一次，就能选出4个（N个）父代
- 保证其中至少一个是个体#3及至少一个是个体#4，因为它们的适应度占适应度值总和的份额都大于25%

随机遍历选择（Stochastic Universal Selection）

算法 8.7 利用随机遍历采样从 N 个个体中选择 N 个父代的伪代码. 此代码假定对于所有 $i \in [1, N]$, $f_i \geqslant 0$.

$\boldsymbol{x}_i =$ 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow \boldsymbol{x}_i$ 的适应度, $i \in [1, N]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

$f_{sum} = 100$

随机数$r \in [0, \quad 100/4] = [0, \quad 25]$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \boldsymbol{x}_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

$$f_1 = 10$$
$$f_2 = 20$$
$$f_3 = 30$$
$$f_4 = 40$$

i	1	2	3	4

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

f_{accum} r=20

i	1	2	3	4

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

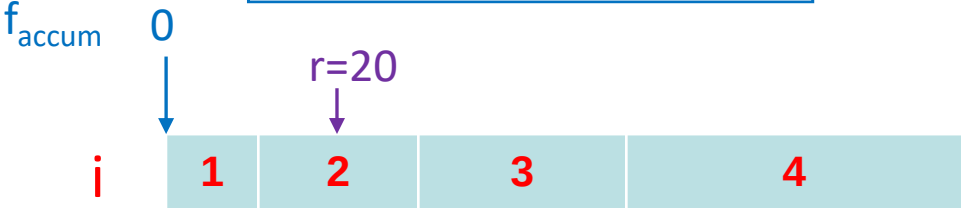
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

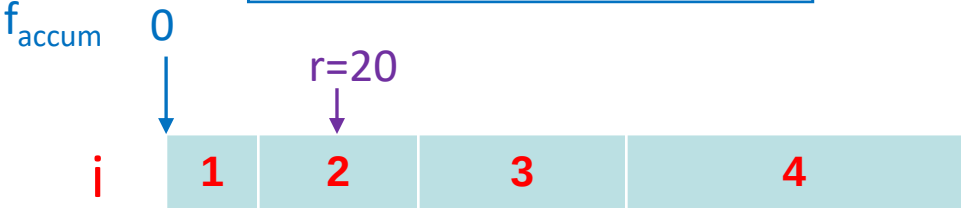
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

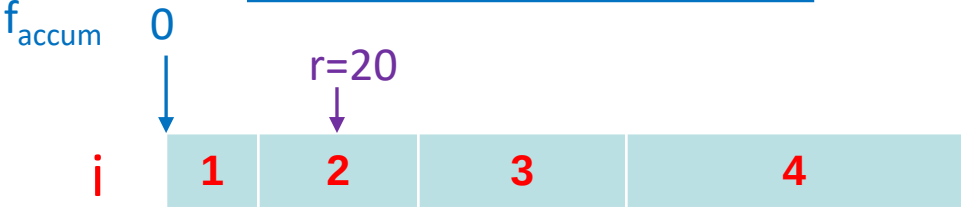
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

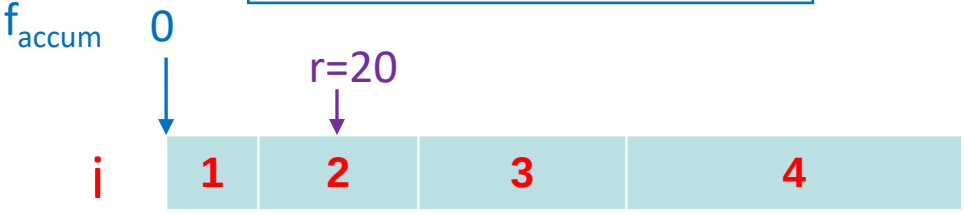
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

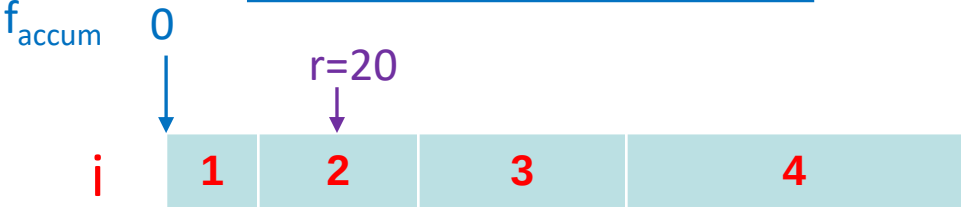
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

    $k \leftarrow k + 1$

    $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

    While $f_{\text{accum}} > r$

        Parents $\leftarrow$ Parents $\cup \ x_k$

        $r \leftarrow r + f_{\text{sum}}/N$

    End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

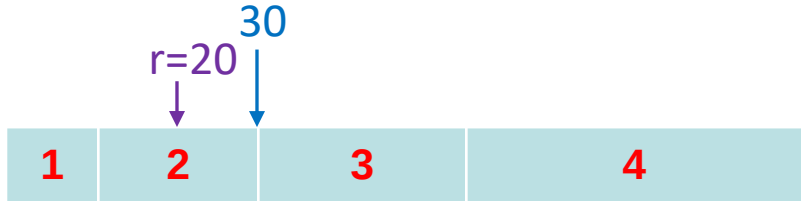
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

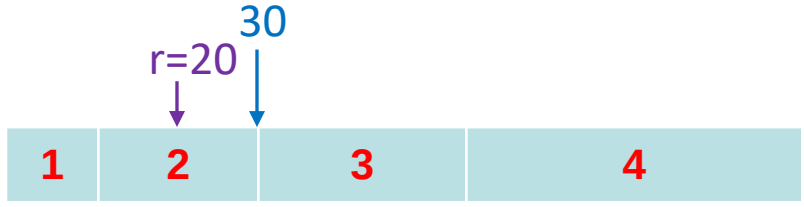
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

随机数 $r \in [0, \quad 100/4] = [0, \quad 25]$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup\ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

假设 $r = 20$

$f_{accum} = 0$

父代为空集(数量为0)

$k = 0$

父代数量 $= 0 < 4$

$k = 0 + 1 = 1$

$f_{accum} = 0 + f_1 = 10$

不满足 $f_{accum} > r$

父代数量 $= 0 < 4$

$k = 1 + 1 = 2$

$f_{accum} = 10 + f_2 = 30$

满足 $f_{accum} > r = 20$, 选择 x_2

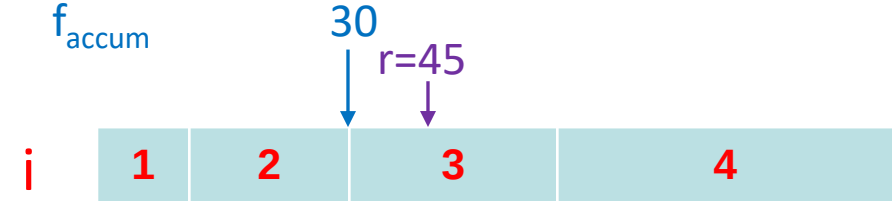
$r = 20 + 25 = 45$ 不满足 $f_{accum} > r$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

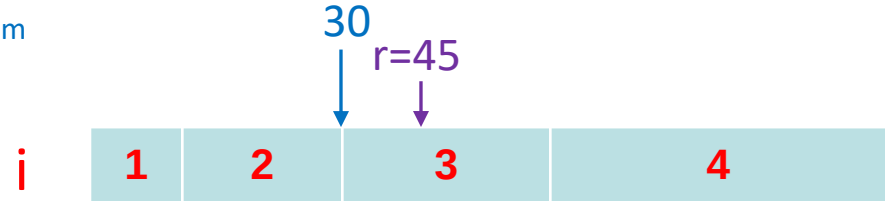
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

    $k \leftarrow k + 1$

    $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

    While $f_{\text{accum}} > r$

        Parents $\leftarrow$ Parents $\cup \ x_k$

        $r \leftarrow r + f_{\text{sum}}/N$

    End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

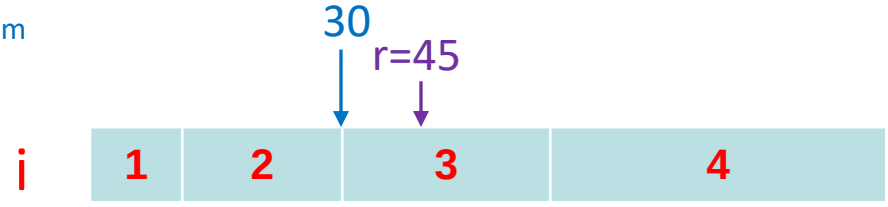
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 $= 1 < 4$

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 $= 2 < 4$

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

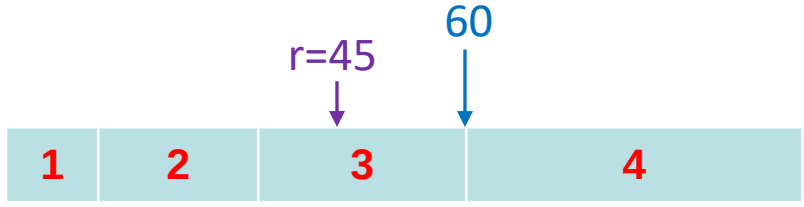
父代（x_2, x_3, x_4, x_4）数量 $= 4$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \quad 100/4] = [0, \quad 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup\, x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

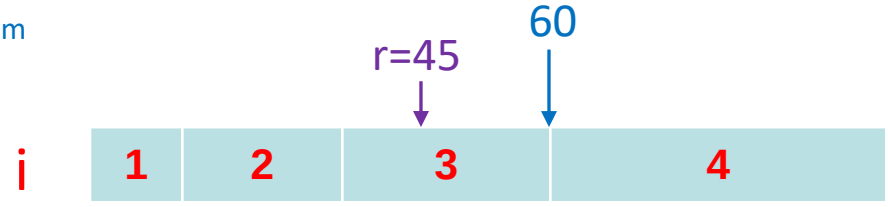
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

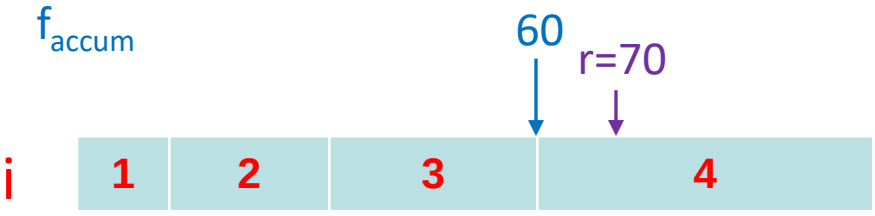
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

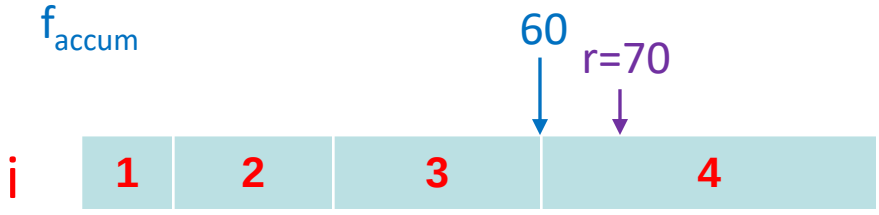
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \quad 100/4] = [0, \quad 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

    $k \leftarrow k + 1$

    $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

    While $f_{\text{accum}} > r$

        Parents $\leftarrow$ Parents $\cup\, x_k$

        $r \leftarrow r + f_{\text{sum}}/N$

    End while

下一个父代

父代（x_2）数量 $= 1 < 4$

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 $= 2 < 4$

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

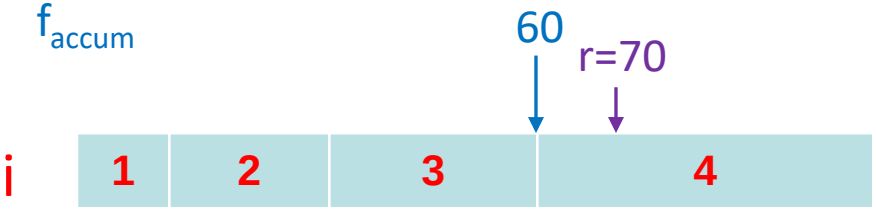
父代（x_2, x_3, x_4, x_4）数量 $= 4$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 $= 1 < 4$

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 $= 2 < 4$

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

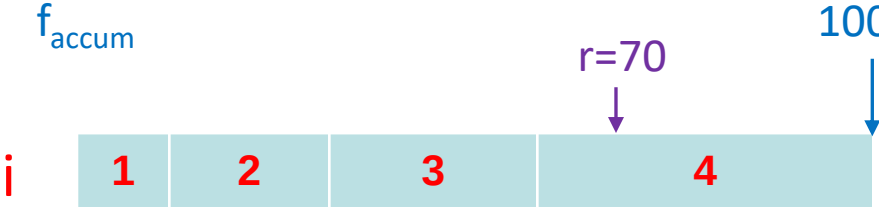
父代（x_2, x_3, x_4, x_4）数量 $= 4$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

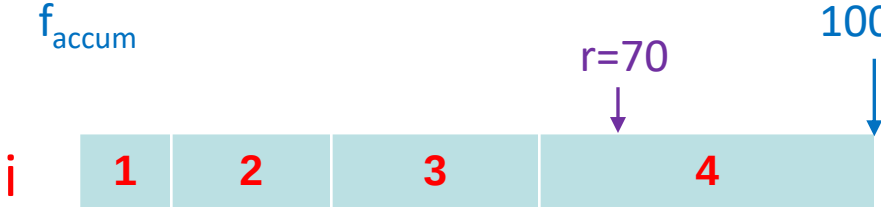
父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 $= 1 < 4$

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 $= 2 < 4$

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

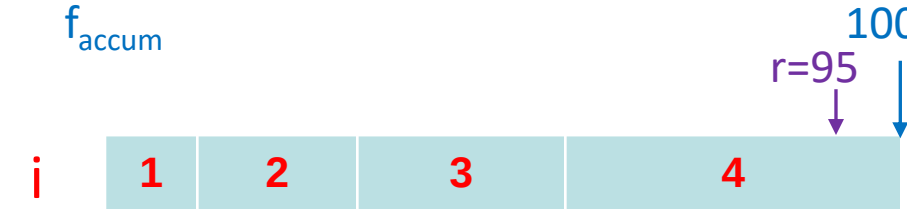
父代（x_2, x_3, x_4, x_4）数量 $= 4$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$



随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 $= 1 < 4$

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 $= 2 < 4$

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

父代（x_2, x_3, x_4, x_4）数量 $= 4$

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

随机遍历选择（Stochastic Universal Selection）

x_i = 种群中第 i 个个体, $i \in [1, N]$

$f_i \leftarrow x_i$ 的适应度, $i \in [1, N]$

$f_{sum} = 100$

随机数 $r \in [0, \ 100/4] = [0, \ 25]$

$$f_{\text{sum}} \leftarrow \sum_{i=1}^{N} f_i$$

生成均匀分布随机数 $r \in [0, f_{\text{sum}}/N]$

$f_{\text{accum}} \leftarrow 0$

Parents $\leftarrow \varnothing$

$k \leftarrow 0$

While | Parents | $< N$

  $k \leftarrow k + 1$

  $f_{\text{accum}} \leftarrow f_{\text{accum}} + f_k$

  While $f_{\text{accum}} > r$

    Parents $\leftarrow$ Parents $\cup \ x_k$

    $r \leftarrow r + f_{\text{sum}}/N$

  End while

下一个父代

父代（x_2）数量 = 1 < 4

$k = 2 + 1 = 3$

$f_{accum} = 30 + f_3 = 60$

满足 $f_{accum} > r = 45$, 选择 x_3

$r = 45 + 25 = 70$ 不满足 $f_{accum} > r$

父代（x_2, x_3）数量 = 2 < 4

$k = 3 + 1 = 4$

$f_{accum} = 60 + f_4 = 100$

满足 $f_{accum} > r = 70$, 选择 x_4

$r = 70 + 25 = 95$

满足 $f_{accum} > r = 95$, 选择 x_4

$r = 95 + 25 = 120$, 不满足 $f_{accum} > r$

父代（x_2, x_3, x_4, x_4）数量 = 4

$f_1 = 10$

$f_2 = 20$

$f_3 = 30$

$f_4 = 40$

f_{accum}

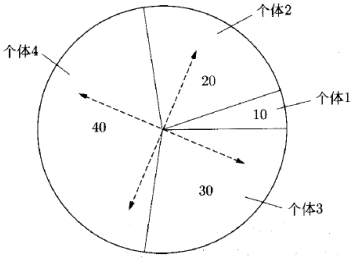
100

r=120

i	1	2	3	4

随机遍历选择（Stochastic Universal Selection）

将随机遍历采样应用于图 8.11 的适应度值，选出的父代如下：



$$\begin{aligned} &\text{个体\#1, \#2, \#3, 和\#4,} \\ \text{或者}\quad &\text{个体\#1, \#3, \#4, 和\#4,} \\ \text{或者}\quad &\text{个体\#2, \#3, \#3, 和\#4,} \\ \text{或者}\quad &\text{个体\#2, \#3, \#4, 和\#4.} \end{aligned} \tag{8.14}$$

算法 8.7 是随机遍历采样的伪代码. 不妨将它与轮盘赌选择算法 3.2 中的代码比较. 算法 8.7 保证个体 $\boldsymbol{x}_i$ 被选择的次数在 $N_{i,\min}$ 和 $N_{i,\max}$ 之间，其中

$$N_{i,\min} = \left\lfloor \frac{N f_i}{f_{\text{sum}}} \right\rfloor, \quad N_{i,\max} = \left\lceil \frac{N f_i}{f_{\text{sum}}} \right\rceil. \tag{8.15}$$

这里，$\lfloor \alpha \rfloor$ 是小于或等于 α 的最大整数，$\lceil \alpha \rceil$ 是大于或等于 α 的最小整数.

$$N_{1,\min} = \left\lfloor \frac{40}{100} \right\rfloor = 0 \qquad N_{1,\max} = \left\lceil \frac{40}{100} \right\rceil = 1 \qquad N_{2,\min} = \left\lfloor \frac{80}{100} \right\rfloor = 0 \qquad N_{2,\max} = \left\lceil \frac{80}{100} \right\rceil = 1$$

$$N_{3,\min} = \left\lfloor \frac{120}{100} \right\rfloor = 1 \qquad N_{3,\max} = \left\lceil \frac{120}{100} \right\rceil = 2 \qquad N_{4,\min} = \left\lfloor \frac{160}{100} \right\rfloor = 1 \qquad N_{4,\max} = \left\lceil \frac{160}{100} \right\rceil = 2$$

常用的选择机制

常用的选择机制：

- 轮盘赌选择（Roulette Wheel Selection）
- 随机遍历选择（Stochastic Universal Selection）
- 基于排名选择（Rank-based Selection）
- 锦标赛选择（Tournament Selection）
- 种马进化算法(Stud Evolutionary Algorithms)

基于排名选择（Rank-based Selection）

为了克服轮盘赌选择“适应度函数值特别高的个体占主导地位，导致多样性少”的局限，提出了基于排名选择机制

步骤：

1）将种群中的每一个个体按照它的适应度函数值进行递增排序。

2)适应度函数值最小的个体的排序为1，其他个体根据适应度函数值依次排序。

3）根据分配的排序，执行按照比例进行选择：

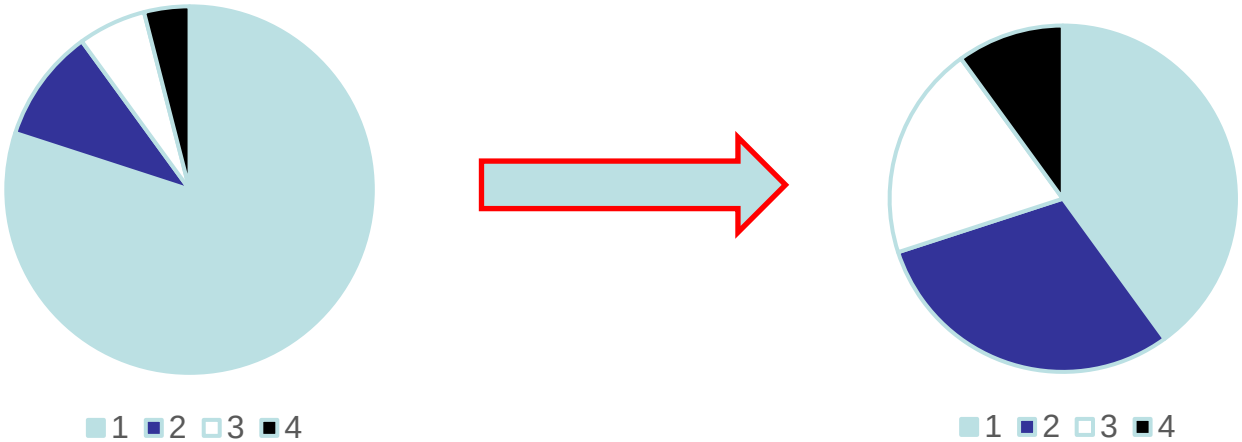
第i个个体占据的%面积 $= \dfrac{r_i}{\sum_{i=1}^{N} r_i}$ 其中r_i为第i个个体的排名

具有相同的适应度函数值的两个或更多的个体具有相同的排名

基于排名选择（Rank-based Selection）

个体(i)	适应度函数值	轮盘(面积)	排名(Rank)	排名选择(面积)
1	0.4	80%	4	40%
2	0.05	10%	3	30%
3	0.03	6%	2	20%
4	0.02	4%	1	10%

- 与轮盘赌选择方式相比较，适应度函数值小的个体被选择的几率得到了提高



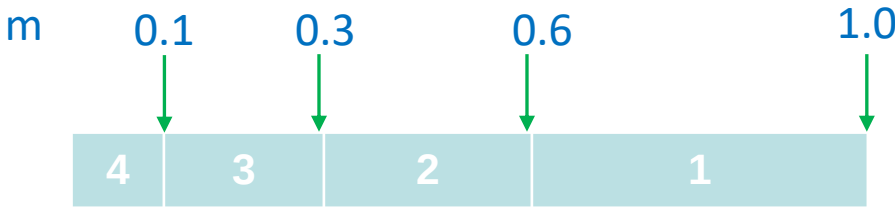
基于排名选择伪代码

```
/* Roulette wheel selection
* 输出参数：
* 选中的染色体
*/
procedure RWS
1  m ← 0;
2  r ← Random(0, 1);
3  for i = 1 to N
4        m ← m + p_i;
5        if r <= m
6              return i;
7        end if
8  end for
end procedure
```

个体 (i)	适应度函数值	轮盘 (面积)	排名 (Rank)	排名选择 (面积)
1	0.4	80%	4	40%
2	0.05	10%	3	30%
3	0.03	6%	2	20%
4	0.02	4%	1	10%

0至1的随机数

$$p_4 = 0.1$$
$$p_3 = 0.2$$
$$p_2 = 0.3$$
$$p_1 = 0.4$$



常用的选择机制

常用的选择机制：

- 轮盘赌选择（Roulette Wheel Selection）
- 随机遍历选择（Stochastic Universal Selection）
- 基于排名选择（Rank-based Selection）
- 锦标赛选择（Tournament Selection）
- 种马进化算法(Stud Evolutionary Algorithms)

锦标赛选择（Tournament Selection）

- 锦标赛的抽样空间大小（Pool Size）：用于选择参加锦标赛的成员数量（通常为N_p）
- 锦标赛的规模(k)：参加锦标赛的成员的数量（通常k=2）
- 锦标赛在k个成员之间进行，适应度函数表现最好的成员入选
- 锦标赛进行N_p轮

假设$k=3$，种群规模为$N_p=6$，选择的解分别是{4， 2， 6}，它们对应的适应度函数值分别是$f_4=27$，$f_2=89$，$f_6=12$

对于最大化问题：解2会赢得锦标赛

对于最小化问题：解6会赢得锦标赛

锦标赛选择（Tournament Selection）

输入：种群的适应度函数值（种群规模为N）

输出：配对池的规模为N_p（$N_p \leq N$）

步骤：

1）随机选择k个个体（$k \leq N$）

2）从k个个体中，选择具有最佳的适应度函数值的个体作为获胜者

3）将选出的获胜者加入到配对池中，配对池初始状态为空集

4）重复步骤1到3, 直至配对池中包含N_p个个体

5）结束

锦标赛选择（Example）

最大化问题，N=8;　　k=2;　　N_p=8

个体	1	2	3	4	5	6	7	8
适应度函数值	1.0	2.1	3.1	4.0	4.6	1.9	1.8	4.5

锦标赛	个体	胜利者（Selected）
1	2，4	4
2	3，8	8
3	1，3	3
4	4，5	5
5	1，6	6
6	1，2	2
7	4，2	4
8	8，3	8

- 表现好的解可能被选入配对池多次
- 锦标赛选择在计算速度上常常优于轮盘赌选择和基于排名选择

锦标赛选择（Tournament Selection）

- 当k=2, 如果锦标赛采用不放回抽样方式开展，每个解会有两次机会参与锦标赛
  - 最好的解会赢得两次锦标赛，会被选择两次
  - 最差的解永远不会胜出，不会被选择
  - 其他的解会有0，1或2次胜出机会
- 随着锦标赛规模k的增加，每个解的选择压力也会增加
- K一般取2~4

k	会发生的情况
2	最差的解不会赢得锦标赛，不会被选择进入父代池
3	最差及次差的解不会赢得锦标赛，不会被选择进入父代池
n	最差的n-1个解不会赢得锦标赛，不会被选择进入父代池

常用的选择机制

常用的选择机制：

- 轮盘赌选择（Roulette Wheel Selection）
- 随机遍历选择（Stochastic Universal Selection）
- 基于排名选择（Rank-based Selection）
- 锦标赛选择（Tournament Selection）
- 种马进化算法(Stud Evolutionary Algorithms)

种马进化算法(Stud Evolutionary Algorithms)

- 在种马进化算法中，每次重组操作始终都选每一代最好的个体，每一代最好的个体被称为种马
- 然后以正常的方式（例如，基于适应度选择、基于排名选择、锦标赛选择）选出其他父代与种马组合生成后代
- 在遗传算法中增加种马逻辑后就将标准遗传算法3.3变成了种马遗传算法8.9

种马进化算法(Stud Evolutionary Algorithms)

算法 3.3 简单遗传算法的伪代码.

Parents ← {随机生成的种群}
While not (终止准则)
    计算种群中每个父代的适应度
    Children ← ∅
    While |Children| < |Parents|
        用适应度根据概率选出一对交配的父代
        父代交配生成子代 c_1 和 c_2
        Children ← Children∪$\{c_1, c_2\}$
    Loop
    一些子代随机变异
    Parents ← Children
下一代

算法 8.9 下面的伪代码概述种马遗传算法.

Parents ← { 随机生成的种群 }
While not (终止准则)
    计算种群中每一个父代的适应度
    Children ← ∅
    $\boldsymbol{x}_1$ ← 适应性最强的父代
    While | Children | < | Parents |
        利用适应度依概率选择第二个父代 $\boldsymbol{x}_2 : \boldsymbol{x}_2 \neq \boldsymbol{x}_1$
        $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 交配生成子代 $\boldsymbol{c}_1$ 和 $\boldsymbol{c}_2$
        Children ← Children ∪$\{\boldsymbol{c}_1, \boldsymbol{c}_2\}$
    Loop
    随机变异某些子代
    Parents ← Children
下一代

种马进化算法(Stud Evolutionary Algorithms)

表 8.1　例 8.11 的结果显示种马方案和无种马方案的遗传算法的相对性能. 此表列出由这两个遗传算法版本找到的归一化后的最小值在 50 次蒙特卡罗仿真上的平均. 基准函数的定义参见附录 C.

基准	非种马遗传算法	种马遗传算法
Ackley	1.44	1
Fletcher	3.26	1
Griewank	3.96	1
Penalty #1	1.05×10^5	1
Penalty #2	160.8	1
Quartic	9.14	1
Rastrigin	1.92	1
Rosenbrock	3.89	1
Schwefel 1.2	1.24	1
Schwefel 2.21	1.65	1
Schwefel 2.22	3.70	1
Schwefel 2.26	2.56	1
Sphere	4.47	1
Step	4.23	1

- 种马进化算法显示了它的优越性，研究文献中的绝大部分（也许全部）遗传算法都使用种马进化算法

选择机制的有效性

- 通常，选择机制遵循达尔文的“适者生存”原则
- 遗传算法中的选择机制倾向于将当前种群中表现好的个体选入配对池中（因此优质基因可以遗传到子代中），搜索趋近于全局最优

选择机制的有效性取决于两个因素

1. 种群的多样性(Population diversity)：类似于探索，种群多样性高意味者在开发当前已经发现的较好基因的同时，允许继续探索新的搜索空间
2. 选择压力（Selection pressure）：具有较好适应度值的候选解“生存”至下一代或被选为父代的可能性，随着选择压力的增加而逐渐提高，而具有较差适应度的候选解则相应地难以“生存”至下一代或被选为父代

选择机制的有效性

- 选择操作将遗传搜索的方向引向最优解所在区域，选择的作用使得群体向最优解所在区域移动。因此，合适的选择压力很重要

- 如果选择压力高
  1. 当前搜索专注于表现好的个体（以适应度函数值来判断）
  2. 失去了种群多样性
  3. 更高的收敛速率。常常导致未成熟的收敛到局部最优解

选择机制的有效性

- 如果选择压力低
  1. 可能无法适当地推动搜索，导致搜索停滞
  2. 收敛速率低，遗传算法可能需要极长的时间找到最优解
  3. 解的精度提高（因为在搜索中考察了更多的基因）

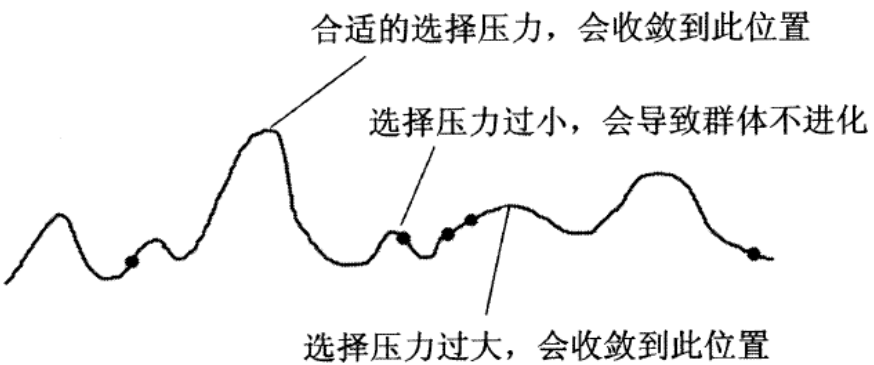


图 2－20　不同选择压力群体进化方向示意图

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

配对池（交叉操作之前）

- 随机从配对池中选择一对父代解。如果配对池的规模是N，那么选择N/2对父代解（随机交叉）

- 通过随机概率的方式（0到1之间的随机数）决定选择的这对父代解是否进行交叉操作，交叉操作的概率为p_c。如果生成的随机数小于p_c，则这对父代解进行交叉操作。否则在种群中保持不变

- 通常，p_c设置为较大的数值（e.g. 0.6~0.9），父代解通常较大概率会执行交叉操作

交叉操作(Crossover Operation)

- 配对池形成后，每一对父代解将经历交叉操作
- 在交叉操作中，会对父代解的性能（属性）进行交换，从而产生两个子代解
- 单点交叉操作交叉点的位置由1到L-1之间的随机整数生成器决定，其中L代表染色体的长度

➢ 存在很多属性交换的机制，因此也有很多交叉操作策略

二进制编码中的交叉操作

交叉操作的机制有很多种，重要的一些如下所示：

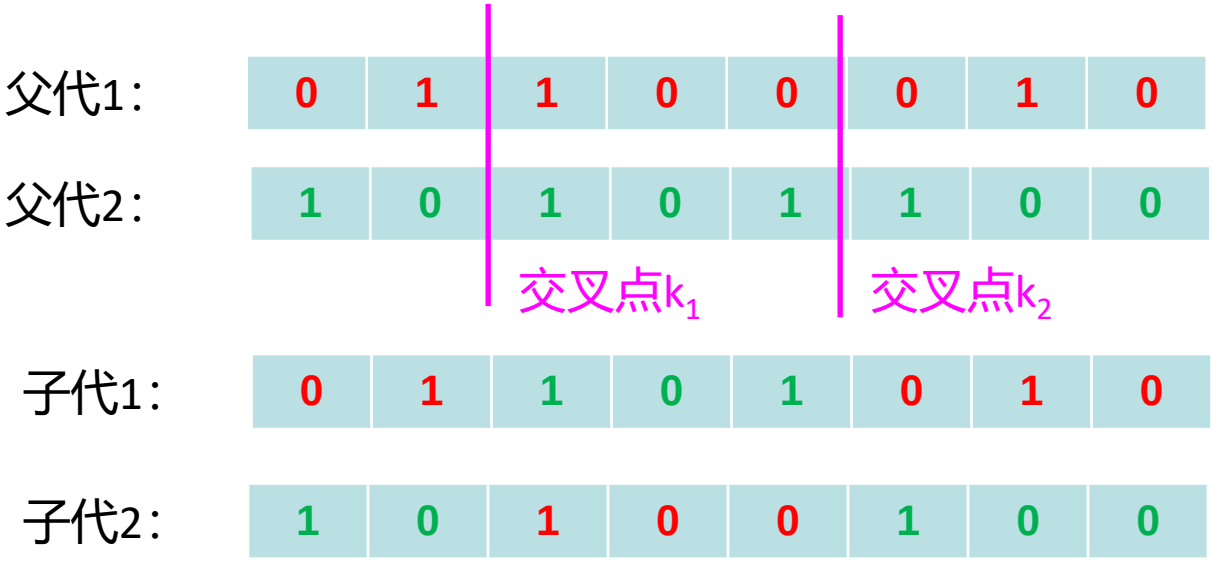
- 单点交叉 (Single point crossover)
- 两点交叉 (Two-point crossover)
- 多点交叉（也称n点交叉）(Multi-point crossover)
- 均匀交叉 (Uniform crossover, UX)
- 半均匀交叉 (Half-uniform crossover, HUX)
- 洗牌交叉 (Shuffle crossover)
- 三亲交叉 （Three-parent crossover）

单点交叉

1. 在1和L-1之间随机选择一个随机整数k
2. 在两个父代染色体中选择第k个位置
3. 两个父代染色体中第k个位置之后的部分进行交换
4. 交换后得到的新的两个染色体就是子代解
5. P_c的典型范围（$0.6 \le P_c \le 0.9$）

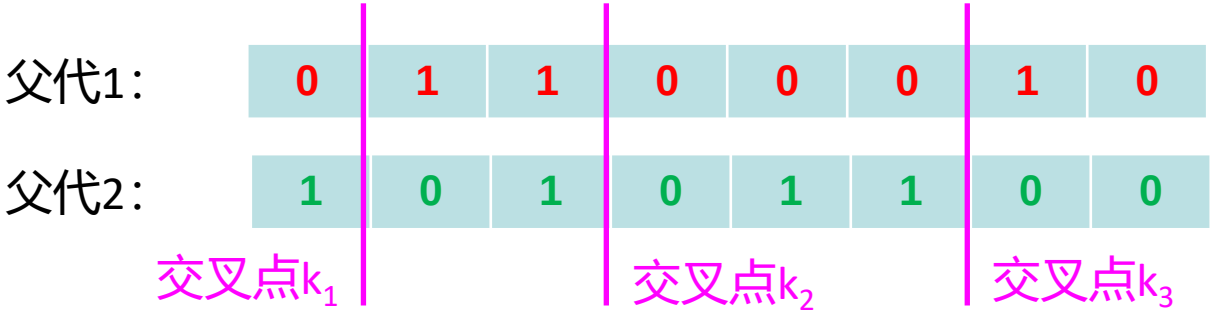
两点交叉

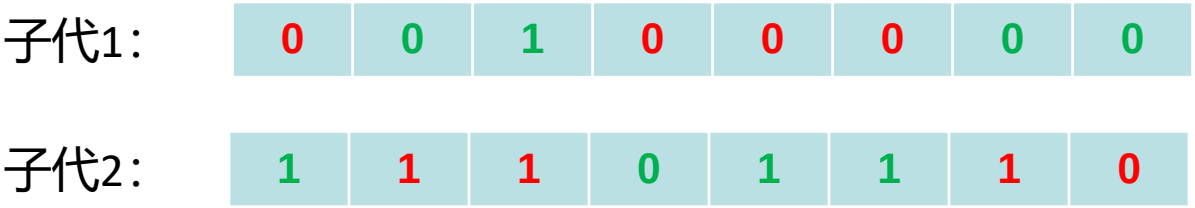
1. 在1和L-1之间随机选择两个不同的随机整数$k_1 \neq k_2$
2. 在两个父代染色体中第k_1个位置和第k_2个位置中间部分进行交换
3. 交换后得到的新的两个染色体就是子代解



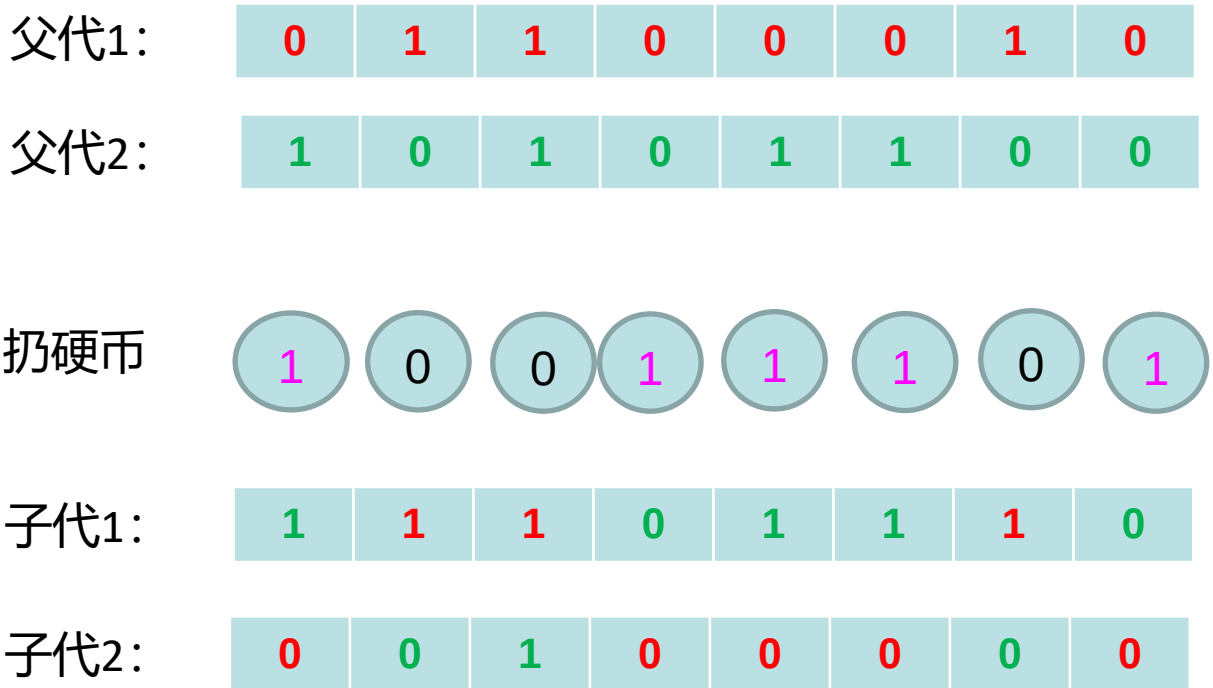
多点交叉

1. 在1和L-1之间随机选择多个不同的随机整数$k_1 \neq k_2 \neq \ldots \neq k_n$
2. 染色体中由多个交叉点分割的区间交替g
3. 交换后得到的新的两个染色体就是子代解





均匀交叉



规则：扔硬币为1交换，为0不交换

半均匀交叉

在半均匀交叉机制中，没有匹配的位（non-matching bits）的50%进行交换

1. 计算父代染色体的汉明距离（Hamming distance）(不同位的数量)
2. 将不同位的数量除以2
3. 得到的数字就是两个父代染色体不匹配的位进行交换的数量（以一定概率）
4. 选择染色体交换的位置（数量=不同位的数量除以2），进行交换

半均匀交叉

父代1：

0	1	1	0	0	0	1	0

父代2：

1	0	1	0	1	1	0	0

扔硬币

子代1：

1	1	1	0	1	1	1	0

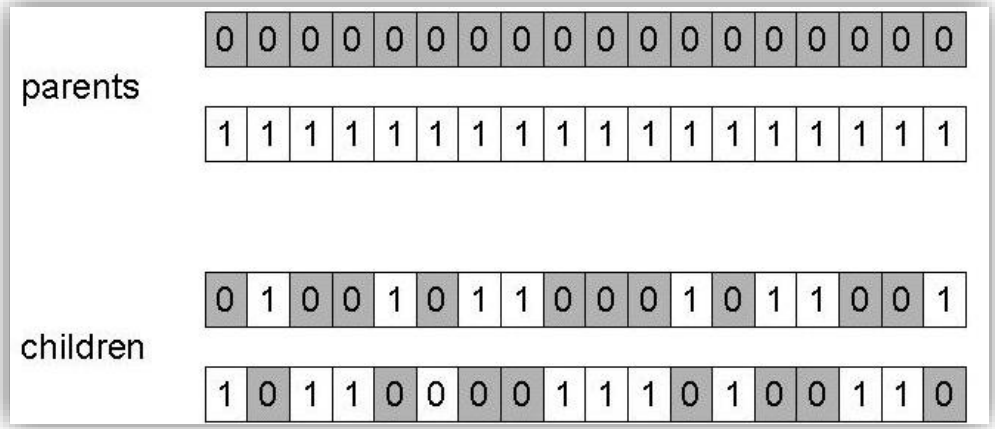
子代2：

0	0	1	0	0	0	0	0

规则：扔硬币为1交换，为0不交换

均匀交叉

1. 均匀交叉是多点交叉的更通用的版本
2. 针对父代染色体的每一位，通过扔硬币的方式（随机整数0或1，概率50%）决定这一位是否进行交换，如果随机数为1则进行交换，否则保持不变
3. 交换后得到的新的两个染色体就是子代解
4. 遗传与基因所在的位置无关



洗牌交叉

- 洗牌交叉将父代的解的特征随机重新排列
- 在为给定的子代提供解的特征的所有父代中，解的特征用相同的方式重排
- 然后采用上面的一种交叉方法（通常单点交叉）得到子代，再撤除对子代解的特征的重排
- 算法8.11是与单点交叉结合的洗牌交叉算法

洗牌交叉

算法 8.11 n 维父代与单点交叉结合的洗牌交叉算法. $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 是两个父代, $\boldsymbol{t}_1$ 和 $\boldsymbol{t}_2$ 是撤除洗牌之前的两个子代, $\boldsymbol{c}_1$ 和 $\boldsymbol{c}_2$ 是撤除洗牌后的子代.

$\{r_1, r_2, \cdots, r_n\} \leftarrow \{1, 2, \cdots, n\}$ 的一个随机排列

交叉点 $m \leftarrow U[1, n-1]$

For $k = 1$ to m

$t_1(k) \leftarrow p_1(r_k)$

$t_2(k) \leftarrow p_2(r_k)$

下一个 k

For $k = m+1$ to n

$t_1(k) \leftarrow p_2(r_k)$

$t_2(k) \leftarrow p_1(r_k)$

下一个 k

For $k = 1$ to n

$c_1(r_k) \leftarrow t_1(k)$

$c_2(r_k) \leftarrow t_2(k)$

下一个 k

假设m=3

父代1:	1	2	3	4	5	6	7	8
父代2:	A	B	C	D	E	F	G	H

随机序列:	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
	4	2	7	1	3	6	8	5

t_1:	4	2	7	A	C	F	H	E
t_2:	D	B	G	1	3	6	8	5

洗牌交叉

算法 8.11 n 维父代与单点交叉结合的洗牌交叉算法. $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 是两个父代, $\boldsymbol{t}_1$ 和 $\boldsymbol{t}_2$ 是撤除洗牌之前的两个子代, $\boldsymbol{c}_1$ 和 $\boldsymbol{c}_2$ 是撤除洗牌后的子代.

$\{r_1, r_2, \cdots, r_n\} \leftarrow \{1, 2, \cdots, n\}$ 的一个随机排列

交叉点 $m \leftarrow U[1, n-1]$

For $k = 1$ to m

$t_1(k) \leftarrow p_1(r_k)$

$t_2(k) \leftarrow p_2(r_k)$

下一个 k

For $k = m+1$ to n

$t_1(k) \leftarrow p_2(r_k)$

$t_2(k) \leftarrow p_1(r_k)$

下一个 k

For $k = 1$ to n

$c_1(r_k) \leftarrow t_1(k)$

$c_2(r_k) \leftarrow t_2(k)$

下一个 k

随机序列:

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
4	2	7	1	3	6	8	5

t_1:	4	2	7	A	C	F	H	E
t_2:	D	B	G	1	3	6	8	5

c_1:	A	2	C	4	E	F	7	H
c_2:	1	B	3	D	5	6	G	8

洗牌交叉

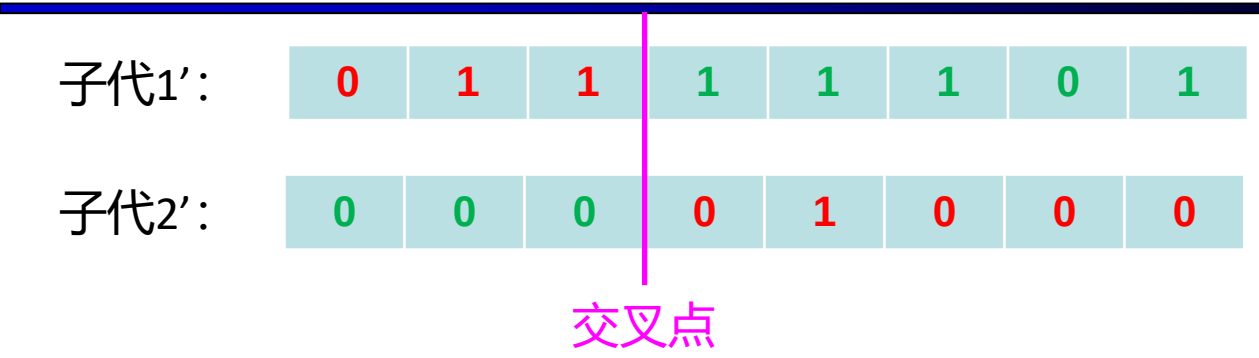
父代1:	0	1	1	0	0	0	1	0
父代2:	1	0	1	0	1	1	0	0

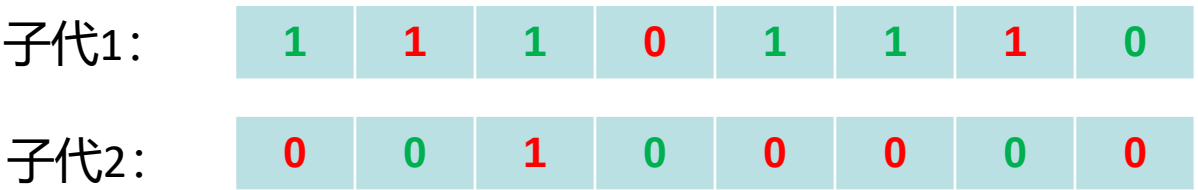
随机序列:	4	2	7	1	3	6	8	5

父代1':	0	1	1	0	1	0	0	0
父代2':	0	0	0	1	1	1	0	1

交叉点

洗牌交叉


子代1’:
0 1 1 1 1 1 0 1
子代2’:
0 0 0 0 1 0 0 0
交叉点
随机序列:
4 2 7 1 3 6 8 5


子代1:
1 1 1 0 1 1 1 0
子代2:
0 0 1 0 0 0 0 0

三亲交叉

- 在当前种群中随机选择三个父代染色体
- 第一个父代染色体的每一位（bit）与第二个父代染色体的每一位进行比较
- 如果相同，那么该位传递给子代
- 如果不相同，就把第三个父代染色体中相应的位传递给子代

三亲交叉

父代1:	0	1	1	0	0	0	1	0
父代2:	1	0	1	0	1	1	0	0
父代3:	1	0	0	1	0	1	0	0
子代:	1	0	1	0	0	1	0	0

二进制交叉技术

- 二进制编码是最简单的编码技术，它对应的二进制交叉技术与其他遗传算法编码交叉技术相比也是运行速度最快的技术

- 不存在某种遗传操作算子在面对任意待求解问题时都具有强于其他操作算子的性能

- 但是，当面对特定问题进行EA算法的设计时，特别是当所采用的待求解问题表示方式中明显存在能够可以有效利用的已知模式或依赖关系时，对不同交叉操作算子的偏差类型透彻理解是非常有价值的

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

变异操作

遗传算法操作包括如下：

1. 遗传算法中，变异操作用于保持基因从一代种群到下一代种群的基因多样性
2. 类比于生物学变异
3. 遗传算法中，利用人工模拟生物学变异的方式对当前解带来局部变化

二进制编码遗传算法中变异操作存在许多变形：

- 翻转（Flipping）
- 交换（Interchanging）
- 颠倒（Reversing）

二进制编码遗传算法的变异操作

1）在二进制编码遗传算法中，变异操作非常简单直观

2）一个或几个1转变为0，或者一个或几个0转变为1

3）常用的方法是对序列中的每一位生成一个随机数，然后与变异概率p_m进行比较

4）变异概率p_m决定了当前这一位是否进行变异

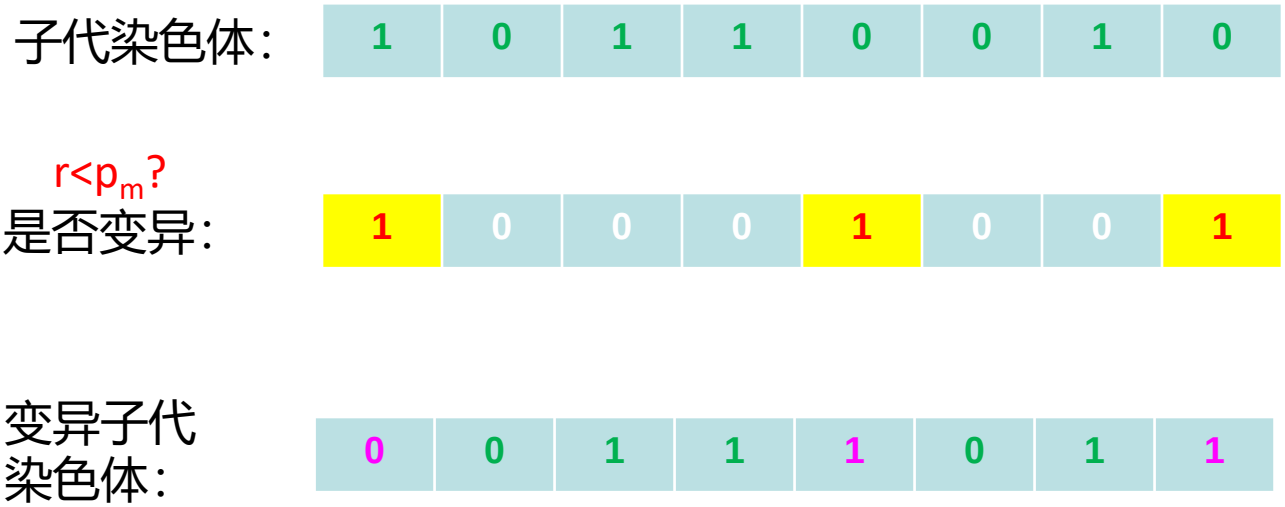
注意：

1）为了避免过大的转向或偏离，通常变异概率p_m的设置值比较小

2）变异概率p_m的范围通常位于0.1/L到1.0/L之间，其中L是编码的长度

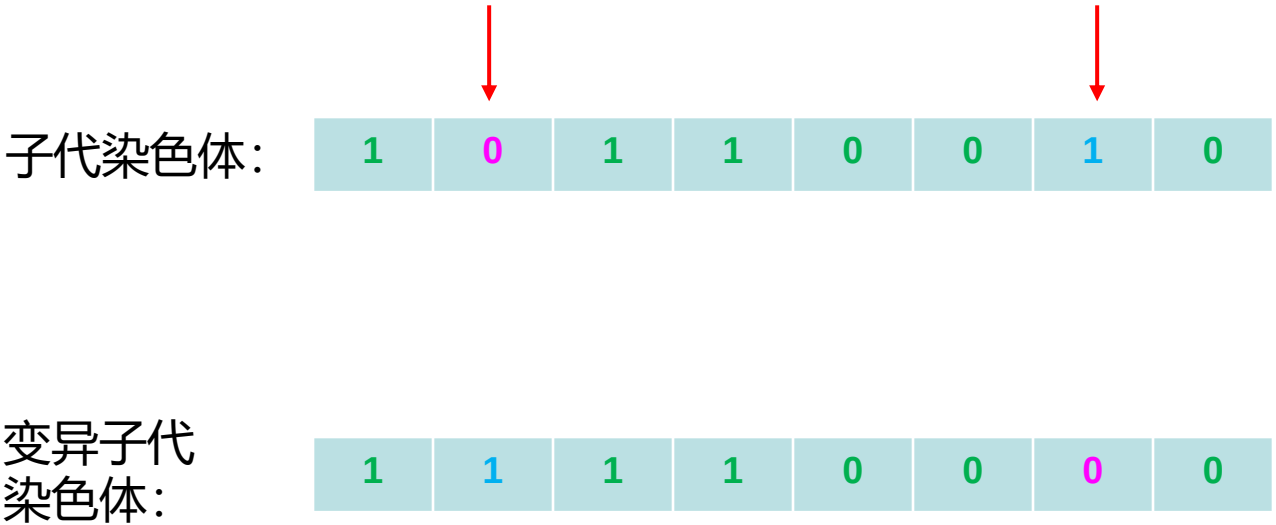
二进制编码遗传算法的变异操作：翻转（Flipping）

- 对应变异染色体的位，在当前染色体的位上进行翻转（1变为0， 或者0变为1）



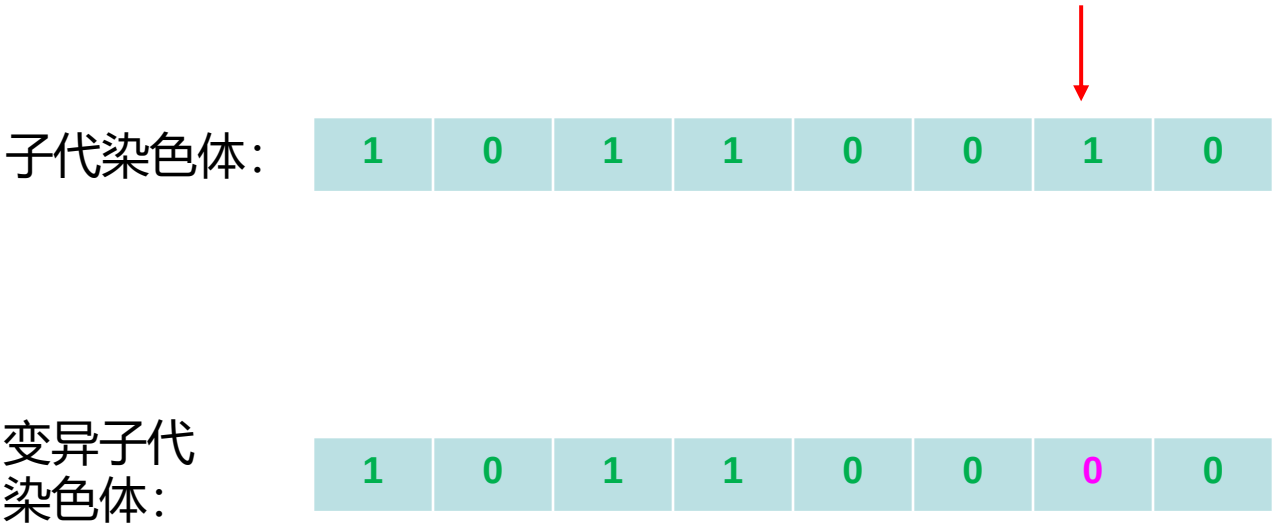
二进制编码遗传算法的变异操作：交换（Interchanging）

- 在子代染色体上随机选择两个位置，把这两个位置对应的值进行交换

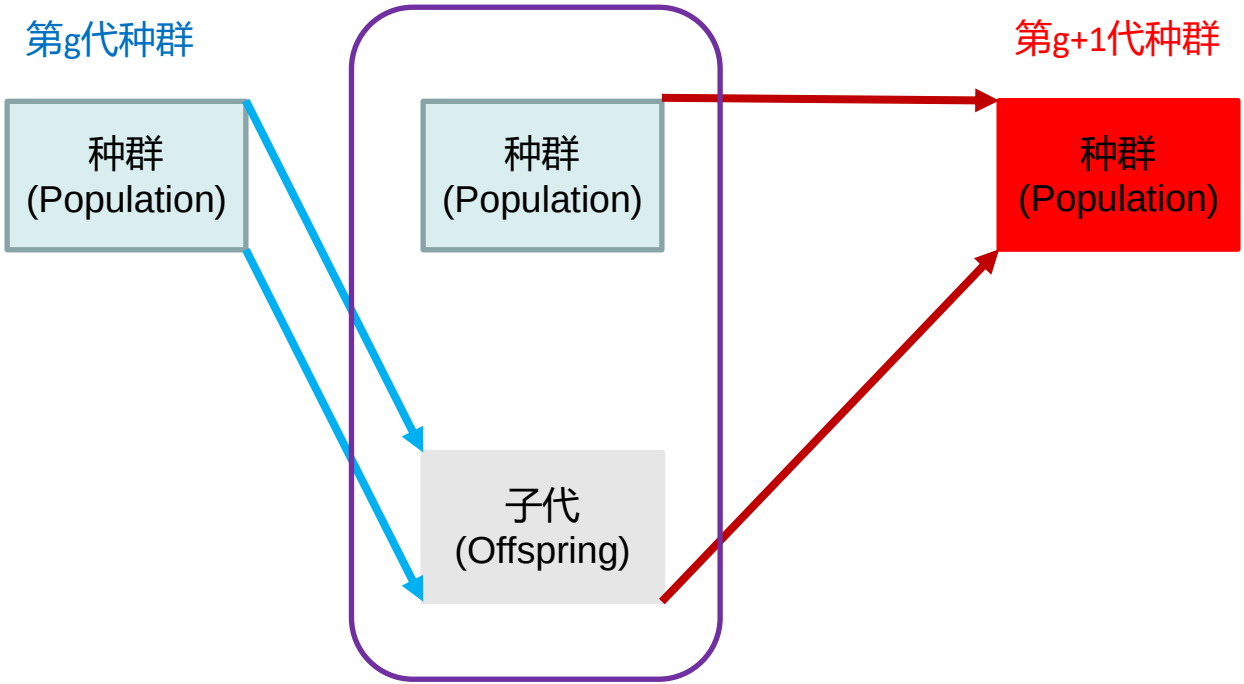


二进制编码遗传算法的变异操作：颠倒（Reversing）

- 在子代染色体上随机选择一个位置，把这个位置进行翻转



原有种群和子代中选择适应度最好的个体



挑选优质解(原有种群，子代)

交叉操作和变异操作

长久时间的辩论:交叉操作和变异操作哪个更好?/哪个是必要的?

答案:

- 取决于优化问题
- 通常，遗传算法最好包括交叉操作和变异操作
- 交叉操作和变异操作承担着不同的角色/任务
- 仅包括变异操作的遗传算法是可能的，仅包括交叉操作的遗传算法通常是不可行的

交叉操作和变异操作

探索（Exploration）：在搜索空间中发现具有优质解的区域，即获得优化问题的信息

开发（Exploitation）：在有希望的区域进行优化，即利用信息

在全局探索和局部开发之间存在着合作与竞争

- 交叉操作具有探索性，它实现了在两个父代区域之间某个区域的大飞跃
- 变异操作具有开发性，它生成了一些随机的小的偏离，因此在停留在父代解区域附近

交叉操作和变异操作

- 只有交叉操作能够把两个父代解的信息进行整合
- 只有变异操作能够引入新的信息（基因）
- 交叉操作不会改变基因在整个种群中的占比（e.g. 假设初始种群中基因“0”在种群中占比50%，在n次交叉操作后基因“0”在种群中占比依然是50%）
- 达到最优解通常需要一个“幸运”的变异

遗传算法的五个关键问题

- 用遗传算法求解问题需要解决以下五个问题：

1. 对问题的潜在解进行基因的表示，即编码问题
2. 构建一组潜在的解决方案，即种群初始化问题
3. 根据潜在解的适应性来评价解的好坏，即个体评价问题
4. 改变后代基因组成的遗传算子（选择、交叉、变异等），即遗传算子问题
5. 设置遗传算法使用的参数值（种群大小、应用遗传算子的概率等），即参数选择问题

控制参数的选取

- 种群规模大小: 直接影响遗传算法的收敛性和计算效率

1. 规模太小，算法容易收敛到局部最优解；
2. 规模过大，又会使算法计算速度降低。
3. 一般情况下取N=20~100

控制参数的选取

- 交叉概率: 控制交叉算子的应用概率

1. 较大的交叉概率可使各代充分交叉，产生更多的新解，以使算法的搜索能力增强，但同时种群中的优良模式遭到破坏的可能性也会增大，进而产生较大的代沟，使搜索走向随机化;
2. 若交叉概率太低，则会使更多的个体直接复制到下一代，遗传搜索就可能陷入停滞状态;
3. 一般取P_c=0.60~0.90

控制参数的选取

- 变异概率

1. 若变异概率取值较大，虽然能够产生较多的个体，增加种群的多样性，但也有可能破坏掉很多好的模式，使遗传算法的性能近似于随机搜索的性能；
2. 若变异概率取值较小，虽然种群的稳定性较好，但算法一旦陷入局部极值就很难再跳出来，容易发生早熟收敛情况；
3. 一般取P_m=0.001~0.01

收敛判据的确定

- 遗传算法使一种反复迭代的搜索方法，它通过多次进化逐渐逼近最优解而不是恰好获得最优解，因此其需要收敛判据。

- 遗传算法的优化准则根据问题的不同，有其相应的确定方式，通常会将以下4种准则作为判断收敛条件
  1. 当最优个体的适应度值达到给定的阈值时，迭代结束
  2. 当最优个体的适应度值和种群的适应度值不再上升时，迭代结束
  3. 当进化迭代次数达到预设的最大进化迭代次数时，迭代结束
  4. 在一些工程优化问题中，最优解的适应度值未知时，依据人为的经验或对问题的期望提出一个理想适应度值，一旦某代最优个体的适应度值超过了这个理想值，则迭代结束

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

示例1：Sphere function

$\min \quad f(x) = \sum_{i=1}^{2} x_i^2 \qquad 0 \le x_i \le 30 \qquad i = 1, \ 2,$

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

第一步：确定种群规模,迭代次数，二进制编码的位数，交叉概率，变异概率

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

示例1：Sphere function

$$\min \quad f(x) = \sum_{i=1}^{2} x_i^2 \qquad 0 \le x_i \le 30 \qquad i = 1, \quad 2,$$

决策变量：x_1, x_2 $\quad D = 2 \quad$ $f(x) = x_1^2 + x_2^2$

第一步：确定种群规模，迭代次数，二进制编码的位数，交叉概率，变异概率

$N_p = 4 \quad T = 10 \quad n = 4 \quad p_c = 0.8 \quad p_m = 0.3$

$$x = x_{\min} + \left(\frac{x_{\max} - x_{\min}}{2^n - 1}\right)(DV)$$

$$x = 0 + \left(\frac{30 - 0}{2^4 - 1}\right)(DV) = 2(DV)$$

第二步：在决策变量的取值范围内生成随机解

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad P_D = \begin{bmatrix} 9 & 12 \\ 3 & 7 \\ 4 & 5 \\ 6 & 10 \end{bmatrix} \quad P_A = \begin{bmatrix} 18 & 24 \\ 6 & 14 \\ 8 & 10 \\ 12 & 20 \end{bmatrix} \quad f = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \end{bmatrix}$$

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

$P_2 = [6 \quad 14]$ $\quad f_2 = 232$

$P_4 = [12 \quad 20]$ $\quad f_4 = 544$

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

$$P_A = \begin{bmatrix} 18 & 24 \\ 6 & 14 \\ 8 & 10 \\ 12 & 20 \end{bmatrix} \qquad f = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \end{bmatrix}$$

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

$P_2 = [6 \quad 14]$ $\quad f_2 = 232$

$P_4 = [12 \quad 20]$ $\quad f_4 = 544$

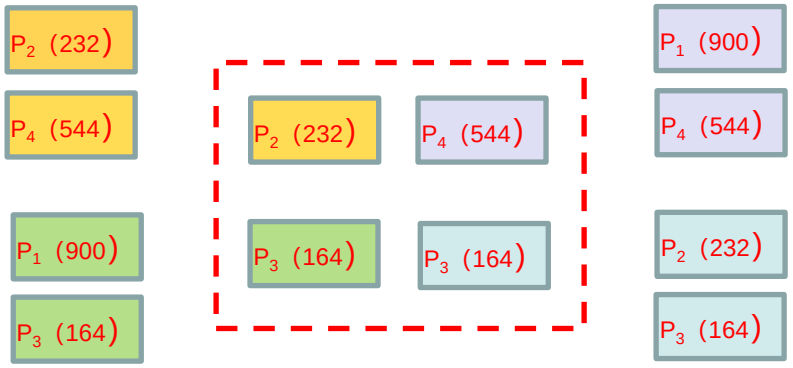
$$P = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

第四步：比较适应度函数值选择胜利者

进入配对池（*Mating pool*）（父代群）

$$P_A = \begin{bmatrix} 18 & 24 \\ 6 & 14 \\ 8 & 10 \\ 12 & 20 \end{bmatrix} \qquad f = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \end{bmatrix}$$

第五步：完成四次（N_p）锦标赛，生成父代群



最好的解被选中两次

最差的解没有被选中

交叉操作：单点交叉

- 生成子代
- 在种群中随机选择两个二进制编码作为父代执行单点交叉
- 父代的一部分编码进行互换生成两个子代
- 交叉点的位置从1到nD-1之间随机选择
- 两个父代进行单点交叉的概率是p_c

假设$n = 4$，$D = 2$，随机选取交叉点的位置 $= 3$

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

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

交叉：单点交叉

第六步：随机选择两个父代进行交叉

假设选择的两个父代解是

$$Parent_1 = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1]$$

$$Parent_2 = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$$

$$Parent = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{matrix} P_2 \\ P_3 \\ P_4 \\ P_3 \end{matrix}$$

交叉：单点交叉

第六步：随机选择两个父代进行交叉

假设选择的两个父代解是

$Parent_1 = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1]$

$Parent_2 = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

$p_c = 0.8$

$$Parent = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{matrix} P_2 \\ P_3 \\ P_4 \\ P_3 \end{matrix}$$

第七步：随机生成一个随机数决定是否进行交叉

假设$r = 0.2$

$r < p_c \rightarrow$ 进行交叉操作

第八步：随机选择一个交叉点

假设$r = 3$

$Parent_1 = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1]$

$Parent_2 = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

$offspring_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

$offspring_2 = [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1]$

交叉：单点交叉

第九步：随机选择两个父代进行交叉

假设选择的两个父代解是

$Parent_3 = [0\ 1\ 1\ 0\ 1\ 0\ 1\ 0]$

$Parent_4 = [0\ 1\ 0\ 0\ 0\ 1\ 0\ 1]$

$p_c = 0.8$

$$Parent = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{matrix} P_2 \\ P_3 \\ P_4 \\ P_3 \end{matrix}$$

第十步：随机生成一个随机数决定是否进行交叉

假设$r = 0.6$

$r < p_c \rightarrow$ 进行交叉操作

第十一步：随机选择一个交叉点

假设$r = 5$

$Parent_3 = [0\ 1\ 1\ 0\ 1 \mid 0\ 1\ 0]$

$Parent_4 = [0\ 1\ 0\ 0\ 0 \mid 1\ 0\ 1]$

$offspring_3 = [0\ 1\ 1\ 0\ 1 \mid 1\ 0\ 1]$

$offspring_4 = [0\ 1\ 0\ 0\ 0 \mid 0\ 1\ 0]$

如果$r>p_c$，不做交叉操作，子代=父代

变异操作：Bit-wise 变异

- 变异概率（p_m）把编码1变成0，或者0变成1
- 在变异过程中只涉及到一个解
- 种群中的每一个成员都可能经历变异
- 对于编码中的每一位，检查变异的概率

假设$p_m = 0.3$，父代二进制编码为

parent = [1 0 1 0 1 0 0 0]

假设每一位的随机数为r = [0.2 0.5 0.6 0.8 0.7 0.1 0.4 0.9]

变异发生在第一个（$r_1 = 0.2 < p_m = 0.3$）和第六个（$r_6 = 0.1 < p_m = 0.3$）变量，子代为

offspring = [0 0 1 0 1 1 0 0]

变异：bit-wise 变异

$p_m = 0.3$

第十二步：选择第一个子代进行变异操作

$Offspring_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

$$Offspring = O = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

第十三步：生成随机数决定哪一位进行变异操作

假设$r = [0.6 \quad 0.1 \quad 0.5 \quad 0.4 \quad 0.9 \quad 0.7 \quad 0.3 \quad 0.8]$

$r < p_m \rightarrow$ 进行变异操作

$Offspring_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$ $\Rightarrow Offspring_1 = [0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]$

变异： bit-wise 变异

第十四步：对所有子代进行变异操作 $p_m = 0.3$

	Offspring	Random number for mutation	New offspring
O_2	[0 1 0 1 0 1 1 1]	[0.5 0.1 0.6 0.3 0.5 0.4 0.6 0.3]	[0 0 0 1 0 1 1 1]
O_3	[0 1 1 0 1 1 0 1]	[0.4 0.5 0.1 0.8 0.5 0.7 0.4 0.2]	[0 1 0 0 1 1 0 0]
O_4	[0 1 0 0 0 0 1 0]	[0.8 0.6 0.3 0.9 0.7 0.5 0.4 0.6]	[0 1 0 0 0 0 1 0]

变异： bit-wise 变异

第十四步：对所有子代进行变异操作 $p_m = 0.3$

	Offspring	Random number for mutation	New offspring
O_2	[0 1 0 1 0 1 1 1]	[0.5 0.1 0.6 0.3 0.5 0.4 0.6 0.3]	[0 0 0 1 0 1 1 1]
O_3	[0 1 1 0 1 1 0 1]	[0.4 0.5 0.1 0.8 0.5 0.7 0.4 0.2]	[0 1 0 0 1 1 0 0]
O_4	[0 1 0 0 0 0 1 0]	[0.8 0.6 0.3 0.9 0.7 0.5 0.4 0.6]	[0 1 0 0 0 0 1 0]

$$X = X_{\min} + \left(\frac{X_{\max} - X_{\min}}{2^n - 1}\right)(DV)$$

$$X = 0 + \left(\frac{30 - 0}{2^4 - 1}\right)(DV) = 2(DV)$$

$$f(x) = x_1^2 + x_2^2$$

$$O = \begin{bmatrix} \overbrace{0\ 1\ 1\ 0}^{X_1} & \overbrace{0\ 1\ 0\ 1}^{X_2} \\ 0\ 0\ 0\ 1 & 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 0 & 1\ 1\ 0\ 0 \\ 0\ 1\ 0\ 0 & 0\ 0\ 1\ 0 \end{bmatrix} \quad O_A = \begin{bmatrix} 12 & 10 \\ 2 & 14 \\ 8 & 24 \\ 8 & 4 \end{bmatrix} \quad f = \begin{bmatrix} 244 \\ 200 \\ 640 \\ 80 \end{bmatrix}$$

幸存策略

第十五步：合并所有的解，选择最佳的N_p（$N_p = 4$）个解

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \end{bmatrix}$$

$$O = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 244 \\ 200 \\ 640 \\ 80 \end{bmatrix}$$

$$P_{combined} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad f_{combined} = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \\ 244 \\ 200 \\ 640 \\ 80 \end{bmatrix}$$

幸存策略

第十五步：合并所有的解，选择最佳的N_p（$N_p = 4$）个解

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \end{bmatrix}$$

$$O = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad f = \begin{bmatrix} 244 \\ 200 \\ 640 \\ 80 \end{bmatrix}$$

$$P_{combined} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad f_{combined} = \begin{bmatrix} 900 \\ 232 \\ 164 \\ 544 \\ 244 \\ 200 \\ 640 \\ 80 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad f = \begin{bmatrix} 80 \\ 164 \\ 200 \\ 232 \end{bmatrix}$$

下一代种群

伪代码

Input : *Fitness function, lb, ub,* N_p, T, n, p_c, p_m, k

1. *Initialize a random population* (P) *of binary string* $(size : N_p \times nD)$
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 $r < p_c$

Select the crossover site

Generate two offspring $using$ $single - point$ *crossover*

else

Copy the selected parents as offspring

end

end

for $i = 1$ *to* N_p

Generate nD random numbers between 0 *and* 1

Perform bit – wise mutation of i^{th} *offspring*

Evaluate the fitness of offspring

end

Combine population and offspring to perform$(\mu + \lambda)$

end

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$j = 1, 2, \ldots, n$

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

0－1背包问题的数学公式如下：

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

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

$x_j = 0$或1, $j = 1, 2, \ldots, n$

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

示例2：0-1背包问题 （Knapsack problem）

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

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）

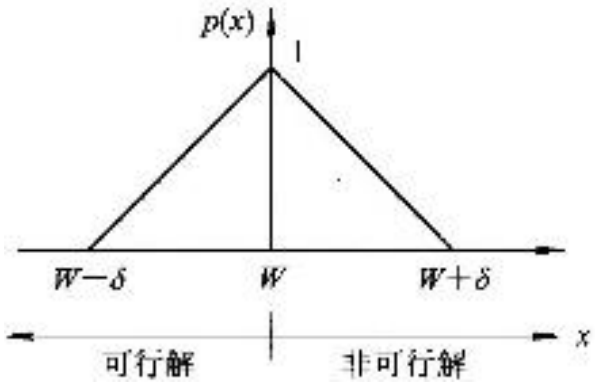
不可行解的处理方法：罚函数法

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

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

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



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) 使用第一拟合启发式算法选择物品，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

(2) 使用第一拟合启发式算法选择物品，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$g(x) = 80 \le W, f(x) = 80$

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

背包承重W=100

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

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

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

对于最大值问题，罚函数可设计为

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

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

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

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$f(x)p(x)$

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

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

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

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

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

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$r = 4$

$$Parent_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \quad offspring_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Parent_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad offspring_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

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

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

假设运行两次轮盘赌得到的第二对父代解为

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

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

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

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad eval = \begin{bmatrix} 79.1 \\ 3 \\ 70 \\ 58.4 \end{bmatrix}$$

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

$$Parent_3 = [0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0]$$

$$Parent_4 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0]$$

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$r = 2$

$$Parent_3 = [0 \quad 1 \mid 1 \quad 1 \quad 0 \quad 1 \quad 0] \qquad offspring_3 = [0 \quad 1 \mid 1 \quad 0 \quad 1 \quad 1 \quad 0]$$

$$Parent_4 = [1 \quad 0 \mid 1 \quad 0 \quad 1 \quad 1 \quad 0] \qquad offspring_4 = [1 \quad 0 \mid 1 \quad 1 \quad 0 \quad 1 \quad 0]$$

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

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

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

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

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
- 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）
- 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