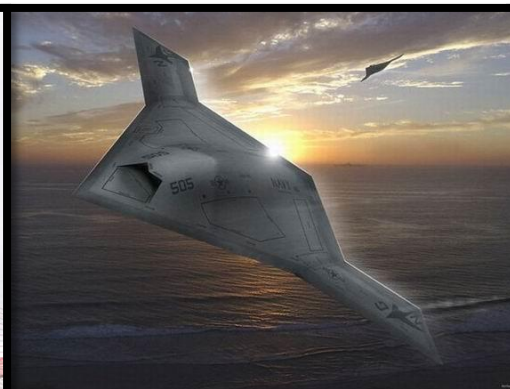


# 机器人智能控制

易建强 蒲志强 袁如意

中国科学院自动化研究所

2020年秋季



# 第二讲 智能优化方法之 遗传算法

蒲志强 高级工程师

zhiqiang.pu@ia.ac.cn



# 本讲的主要内容

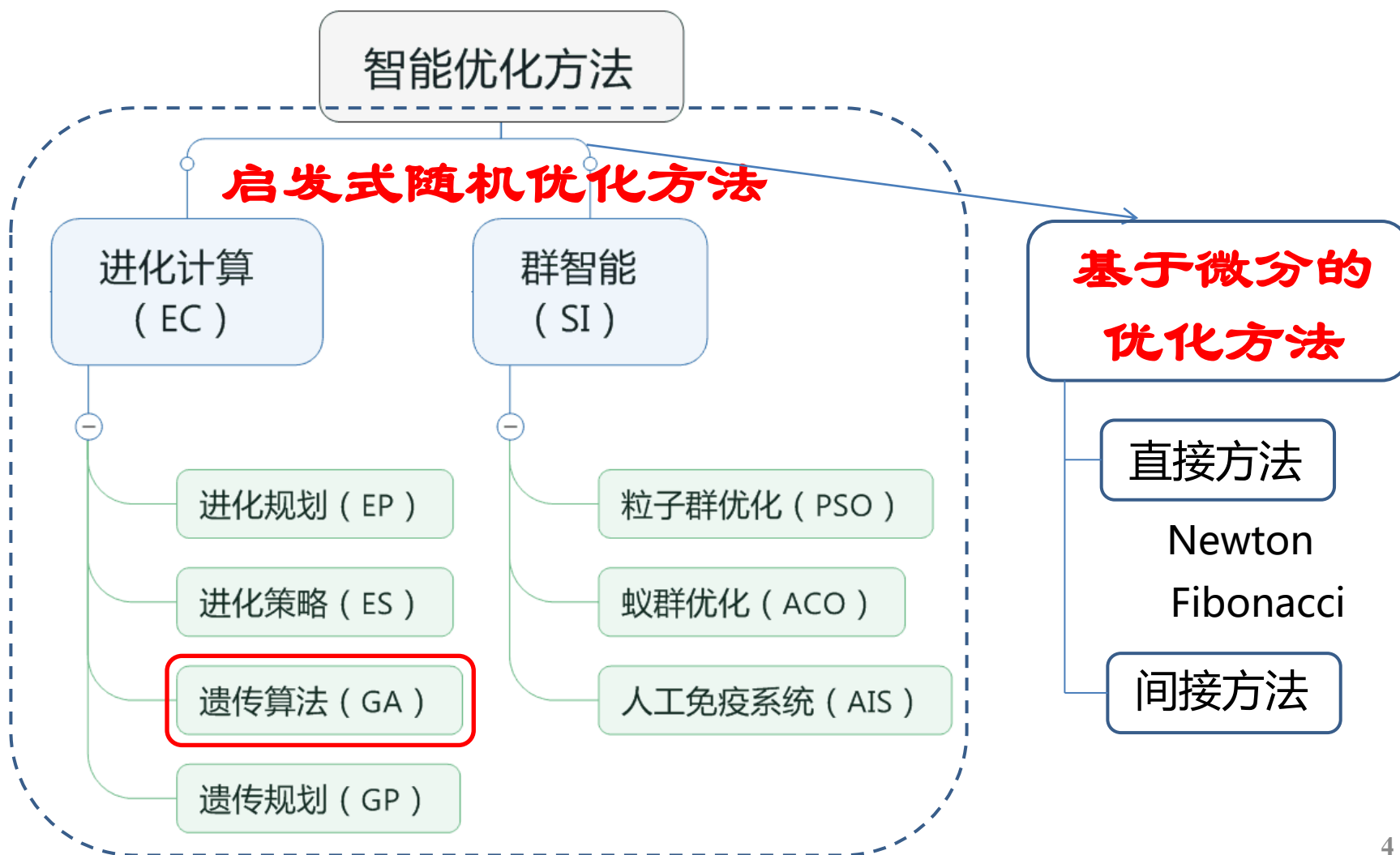
一、背景介绍

二、基本遗传算法

三、改进型遗传算法

四、应用实例

# 一、背景介绍



## ■ 什么是进化计算

进化计算是一种模拟自然界生物进化过程与机制进行问题求解的自组织、自适应的随机搜索技术。它以达尔文进化论的“**物竞天择、适者生存**”作为算法的进化规则，并结合孟德尔的**遗传变异**理论，将生物进化过程中的

- **繁殖** (Reproduction)
- **变异** (Mutation)
- **竞争** (Competition)
- **选择** (Selection)

引入到了算法中。

## ■ 进化计算的生物学基础

自然界生物进化过程是进化计算的生物学基础，它主要包括遗传 (Heredity)、变异 (Mutation) 和进化 (Evolution) 理论。

### □ 进化论

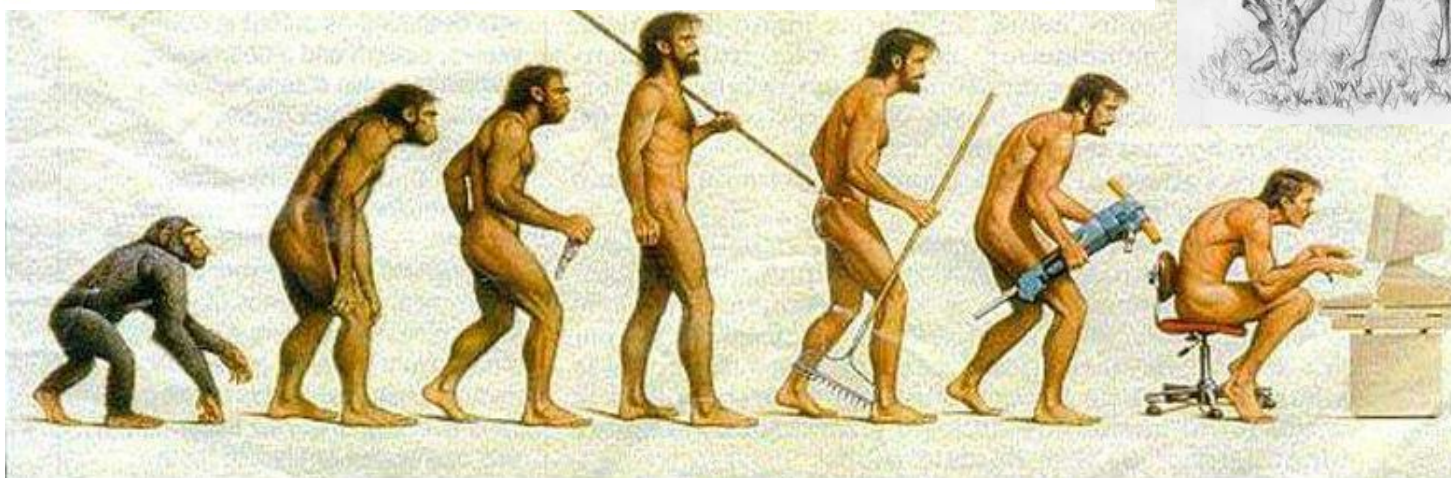
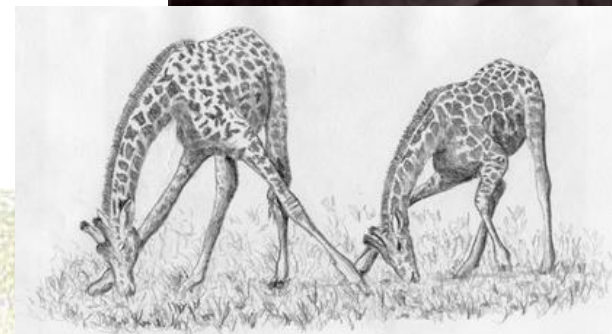
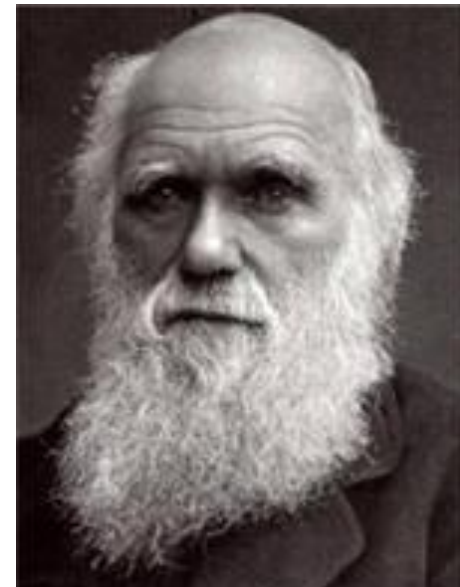
进化是指在生物延续生存过程中，逐渐适应其生存环境，使得其品质不断得到改良的这种生命现象。遗传和变异是生物进化的两种基本现象，优胜劣汰、适者生存是生物进化的基本规律。



长颈鹿的祖先当中，脖子有长有短。脖子短的无法和脖子长的竞争食物，渐渐被淘汰了。几代之后，所有的长颈鹿都是长颈的。



达尔文的自然选择学说认为：在生物进化中，一种基因有可能发生变异而产生出另一种新的生物基因。这种新基因将依据其与生存环境的适应性而决定其增殖能力。一般情况下，适应性强的基因会不断增多，而适应性差的基因则会逐渐减少。通过这种自然选择，物种将逐渐向适应于生存环境的方向进化，甚至会演变成为另一个新的物种，而那些不适应于环境的物种将会逐渐被淘汰。



## □ 遗传理论

遗传是指父代（或亲代）利用遗传基因将自身的基因信息传递给下一代（或子代），使子代能够继承其父代的特征或性状的这种生命现象。正是由于遗传的作用，自然界才能有稳定的物种。



孟德尔  
遗传学的奠基人

“种瓜得瓜，种豆得豆”  
“龙生龙，凤生凤，老鼠生儿打地洞”



## □ 变异理论

变异是指子代和父代之间，以及子代的各个不同个体之间产生差异的现象。变异是生物进化过程中发生的一种随机现象，是一种不可逆过程，在生物多样性方面具有不可替代的作用。引起变异的主要原因有以下两种：

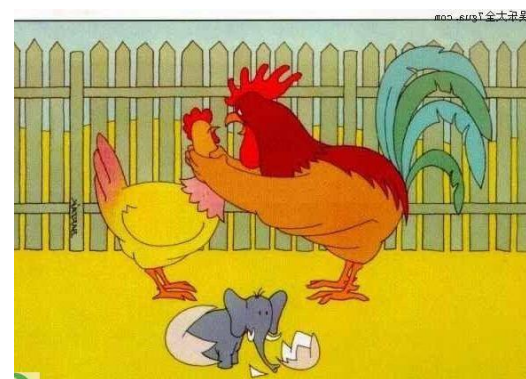
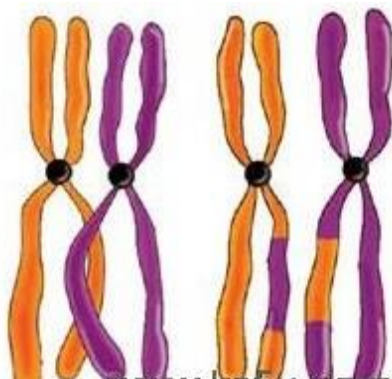
### ✓ 杂交

指有性生殖生物在繁殖下一代时两个同源染色体之间的交配重组，即两个染色体在某一相同处的DNA被切断后再进行交配重组，形成两个新的染色体。

### ✓ 复制差错

指在细胞复制过程中因DNA上某些基因结构的随机改变而产生出新的染色体。

龙生九子，九子九样



- 20世纪50年代和60年代

- 少数几个计算机科学家独立地进行所谓的“人工进化系统”研究，其出发点是进化的思想可以发展成为许多工程问题的优化工具。
- 遵循适者生存的仿自然法则。
- 有些系统基于种群，引入选择和变异操作。
- 有些系统对生物染色体抽象处理，引入二进制编码。

- 20世纪60年代初期

- 柏林工业大学的Rechenberg和Schwefel等在进行风洞试验时，由于描述物体形状的参数难以用传统方法进行优化，提出了利用生物变异的思想来随机改变参数值，效果很好。
- 对这种方法深入研究，形成了另一个分支  
进化策略ES (Evolutionary Strategy)。

- 20世纪60年代初期

- L. J. Fogel 等人在设计有限自动机时提出了  
进化规划EP (Evolutionary Programming) 。
- 将此方法应用到数据诊断、模式识别和分类及控制系统设计等问题中，取得了较好的效果。
- 后来用于数值优化和神经网络的训练上。

- 20世纪60年代中期

- 美国 Michigan 大学的 John Holland提出了位串编码技术，适用于变异操作，并用于自然和人工系统的自适应行为研究。



- 1975年

- Holland出版了开创性著作“Adaption in Natural and Artificial Systems”，提出了遗传算法Genetic Algorithm (GA)



- 20世纪90年代

- 1992年，美国斯坦福大学的Koza在其出版的著作中提出遗传规划（也叫遗传编程，Genetic Programming, GP）算法。
- GP采用层次化的树结构表达问题，类似于计算机程序分行或分段描述问题
- GP也采用遗传算子（复制、选择、变异等）对种群不断迭代，因此可以看成是GA的特殊情况

## • 40余年来

- 遗传算法本身逐渐成熟，用来解决实际问题或建模的应用范围不断扩展。
- 很多遗传算法与Holland的算法已有很大区别，但改进方法的灵感都来自大自然的生物进化。
- 进化计算与人工神经网络、模糊系统理论形成了一个新的研究方向：**计算智能**。
- 人工智能已从传统的基于符号处理的符号主义、向以神经网络为代表的连接主义和进化计算为代表的进化主义方向发展。

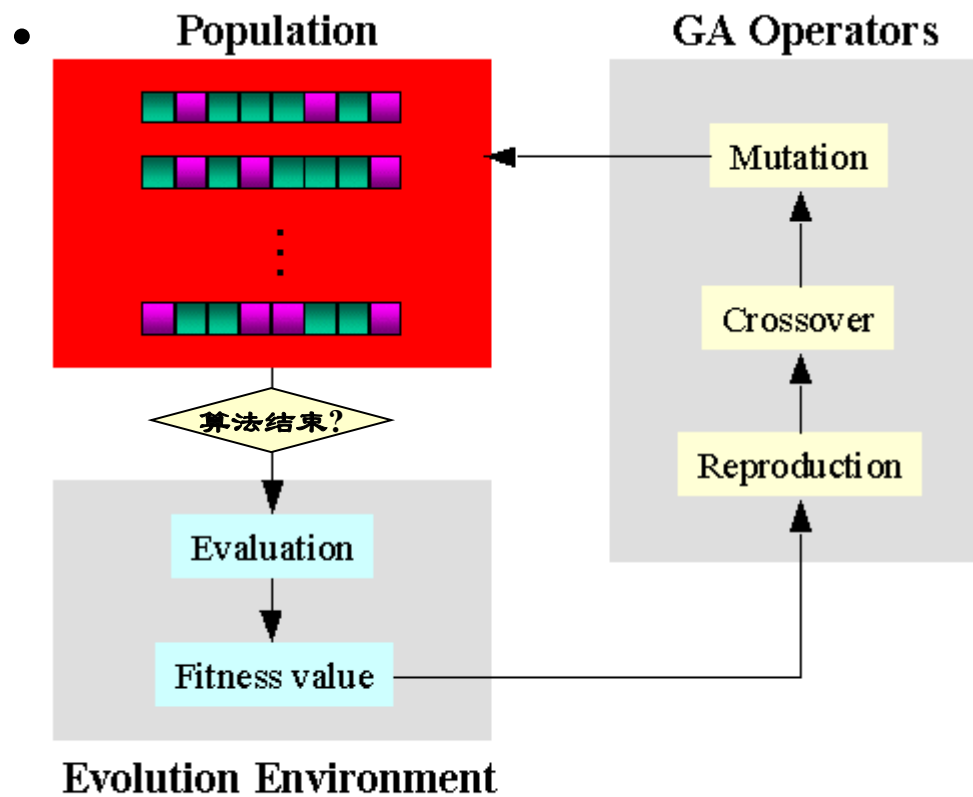
## 二、基本遗传算法

基本遗传算法 Simple Genetic Algorithm (SGA)

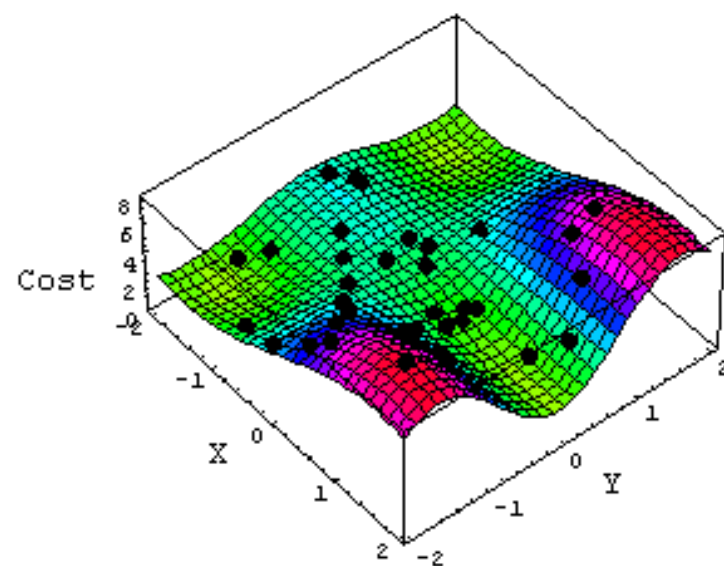
1. 基本概念
2. 个体：编码方法，个体数
3. 适应度：定义，尺度变换
4. 遗传操作：选择，交叉，变异
5. 终止条件
6. 例子向导
7. 例子讨论

遗传算法的基本思想是从初始种群出发，采用优胜劣汰、适者生存的自然法则选择个体，并通过杂交、变异来产生新一代种群，如此逐代进化，直到满足目标为止。遗传算法所涉及到的基本概念主要有以下几个：

- ✓ **种群 (Population)**：种群是指用遗传算法求解问题时，初始给定的多个解的集合。遗传算法的求解过程是从这个子集开始的。
- ✓ **个体 (Individual)**：个体是指种群中的单个元素，它通常由一个用于描述其基本遗传结构的数据结构来表示。例如，可以用0、1组成的长度为 $l$ 的串来表示个体。
- ✓ **染色体 (Chromos)**：染色体是指对个体进行编码后所得到的编码串。染色体中的每1位称为基因，染色体上由若干个基因构成的一个有效信息段称为基因组。
- ✓ **适应度 (Fitness) 函数**：适应度函数是一种用来对种群中个体的环境适应性进行度量的函数。其函数值是遗传算法实现优胜劣汰的主要依据
- ✓ **遗传操作 (Genetic Operator)**：遗传操作是指作用于种群而产生新的种群的操作。标准的遗传操作包括以下三种基本形式：
  - 选择 (Selection)
  - 交叉 (Crossover)
  - 变异 (Mutation)



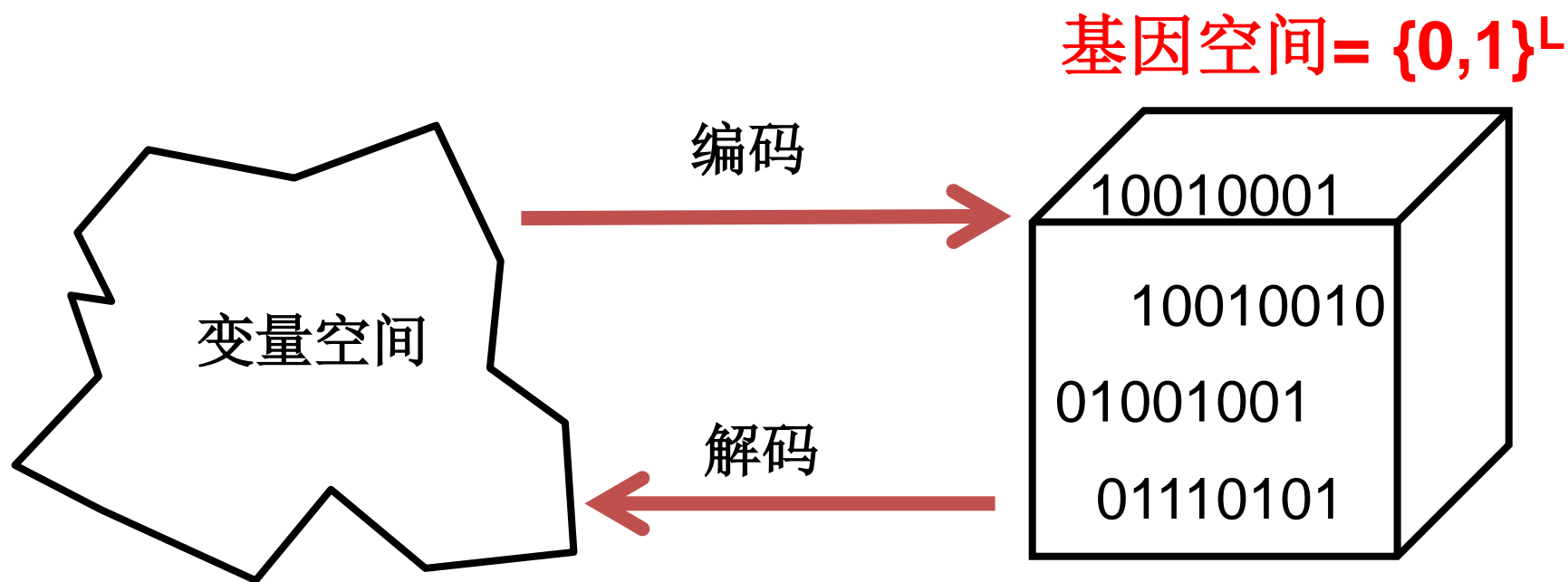
遗传算法的基本流程



逼近最优解的过程



- 待优化的参数：单变量，多变量
  - 编码：二进制，十进制，符号
  - 个体数 ( $M$ )：初始种群数目



## □ 个体数的选取

- $M$  越大，个体的多样性越高，但运行效率变低
- $M$  越小，可提高算法的运行速度，但个体的多样性变差
- 一般取20-100

## □ 个体编码-二进制编码

### • 编码方法

$$\begin{array}{ll}
 000000 \cdots 000000 = 0 & \rightarrow U_{\min} \\
 000000 \cdots 000001 = 1 & \rightarrow U_{\min} + \delta \\
 \vdots & \vdots \\
 111111 \cdots 111111 = 2^l - 1 & \rightarrow U_{\max}
 \end{array}$$

### • 编码精度和编码长度

$$\delta = \frac{U_{\max} - U_{\min}}{2^l - 1} \quad \longleftrightarrow \quad l \geq \log_2 \left( \frac{U_{\max} - U_{\min}}{\delta} + 1 \right)$$

## □ 个体编码-二进制编码

- 举例

$$(U_{\min}, U_{\max}) = (-12.3, 23.4)$$

$$\delta = 0.1$$

$$x = 6.7$$

试写出 $x$ 的二进制编码 $X$

$$l = 9 \geq 8.48$$

$$X = \text{Int}\left(\frac{x - U_{\min}}{U_{\max} - U_{\min}} \times (2^l - 1)\right) = 272 = \{100010000\}$$

## □ 个体编码-格雷编码

- 二进制码  
 $X_1$ : 0111  
 $X_2$ : 1000
- 格雷码  
 $X_1$ : 0100  
 $X_2$ : 1100
- 格雷码属于可靠性编码，是一种错误最小化的编码方式。编码串之间的一位差异，对应的参数值也只是微小的差别。相当于提高了遗传算法的局部搜索能力。

十进制	二进制	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



## ➤ 格雷码特性

- 格雷码是一种变权码，每一位码没有固定的大小，很难直接进行比较大小和算术运算，常用到与二进制码之间的转化。
- 格雷码的十进制数奇偶性与其码字中1的个数的奇偶性相同。

## ➤ 格雷码的递归生成规则

1. 1位格雷码有两个码字
2.  $(n+1)$ 位格雷码中的前 $2^n$ 个码字等于 $n$ 位格雷码的码字，按顺序书写，加前缀0
3.  $(n+1)$ 位格雷码中的后 $2^n$ 个码字等于 $n$ 位格雷码的码字，按逆序书写，加前缀1（**对偶**）
4.  $n+1$ 位格雷码的集合 =  $n$ 位格雷码集合(顺序)加前缀0 +  $n$ 位格雷码集合(逆序)加前缀1

## ➤ 格雷码与二进制码间的转换

**n位二进制码→n位格雷码（编码）：**

- 对n位二进制的码字，从右到左，以0到n-1编号
- 如果二进制码字的第i位和i+1位相同，则对应的格雷码的第i位为0，否则为1（当i+1=n时，二进制码字的第n位被认为是0，即第n-1位不变）。

$$G_i = B_i \oplus B_{i+1} (n-1 \geq i \geq 0)$$

**n位格雷码→n位二进制码（解码）：**

从左边第二位起，将每位与左边一位**解码后的值**异或，作为该位解码后的值（最左边一位依然不变）。依次异或转换后的值（二进制数）就是格雷码转换后二进制码的值。

$$B_i = G_i \oplus B_{i+1} (n-1 \geq i \geq 0)$$

## □ 个体编码-浮点数编码

- 在多参数、高精度的函数优化时，二进制编码可能使搜索空间变得很大，使遗传算法的运行性能变差。
- 使用变量的真实值，也称为真值编码。
- 函数优化问题，应用较多。

$$x = \{1.2, 2.3, 3.4\}$$



$$X = \{1.2, 2.3, 3.4\}$$

## □ 个体编码-符号编码

- 染色体编码串中的基因值取自一个无数值含义、而只有代码含义的符号集，如：

$$T = (1, 2, 3, 4, 5, 6, 7, 8, \dots)$$

$$T = (A, B, C, D, E, F, G, \dots)$$

- 特点：编码代表实际意义，而交叉和变异操作需要特别设计。

## □ 个体编码练习

已知

- 取值范围：  $(U_{\min}, U_{\max}) = (-10, 10)$
- 精度精度：  $\delta = 0.1$

写出以下个体的二进制编码和格雷编码

$$x = \{-5.6, 7.4, 3.2\}$$

$$l \geq \log_2 \left( \frac{U_{\max} - U_{\min}}{\delta} + 1 \right) = \log_2 201 = 7.65$$

$$\text{Int} \left( \frac{-5.6 - U_{\min}}{U_{\max} - U_{\min}} \times (2^l - 1) \right) = 56 = \{00111000\}$$

$$X = \{00111000, 11011101, 10101000\}$$

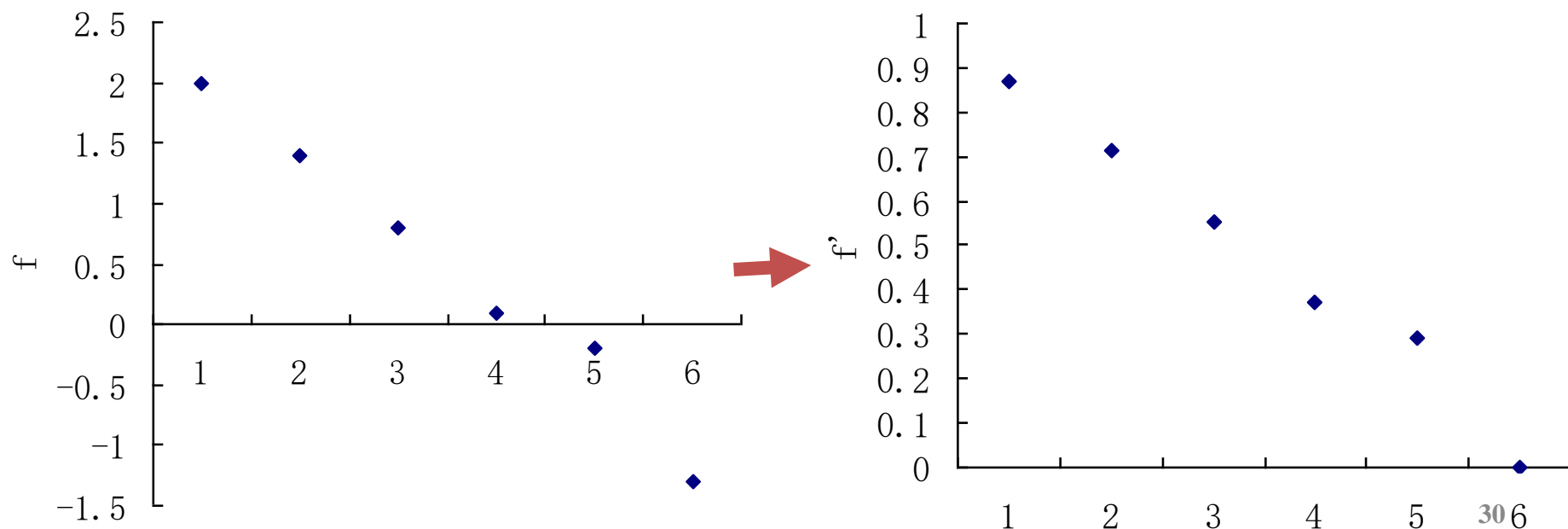


- 适应度：个体在进化过程中有可能达到或接近于最优解的优良程度。
- 适应度函数：度量个体适应度的函数，也称为待求解问题的目标函数值。
- 适应度尺度变换的根据：
  - 目标函数值的范围及分布
  - 目标函数的最大值问题
  - 目标函数的最小值问题

## □ 尺度变换1——线性变换

— 为保证适应度非负

$$f' = \alpha \times f + \beta \quad \alpha = \frac{f_{avg}}{f_{avg} - f_{min}} \quad \beta = \frac{-f_{min} f_{avg}}{f_{avg} - f_{min}}$$



## □ 尺度变换2—非线性变换

— 幂函数变换法

$$f' = f^k$$

— 指数函数变换法

$$f' = e^{-\alpha f}$$

— 对数函数变换法

$$f' = \log f$$

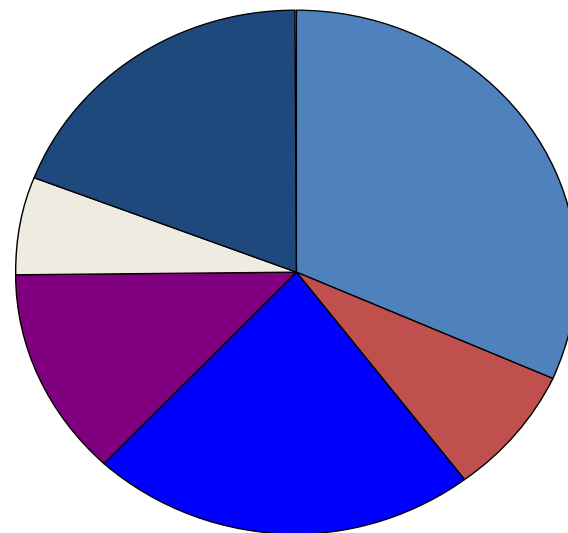
## □ 选择 (selection)

是根据个体的适应度及其分布，确定其在下一代中被选择的概率，也称复制。

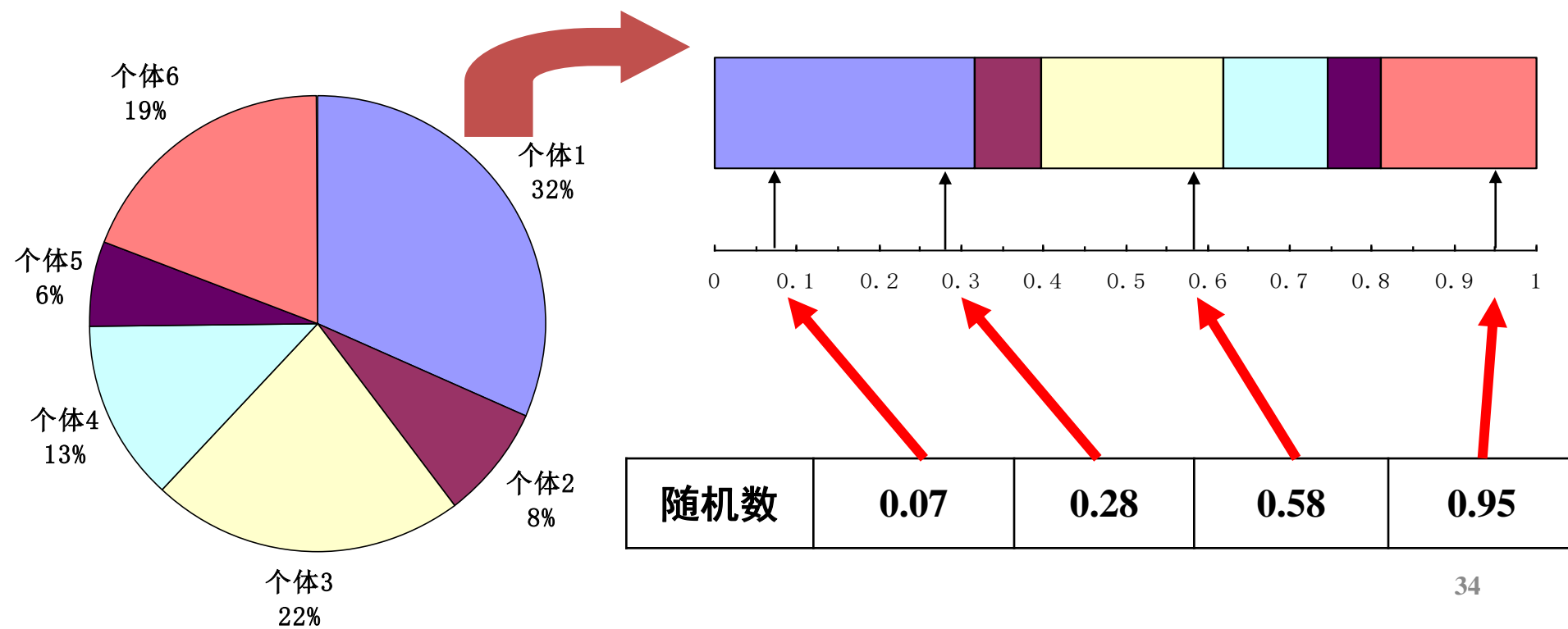
- 个体选择概率的常用分配方法：
  - 按比例的比例适应度分配方法
  - 基于排序的适应度分配
    - 适应度仅取决于个体在种群中的位置，而不是实际的目标值。

## □ 选择方法1—轮盘赌

- 轮盘赌选择法
  - 产生  $[0, 1]$  之间的随机数，作为选择指针来确定被选个体
  - 适应度为正



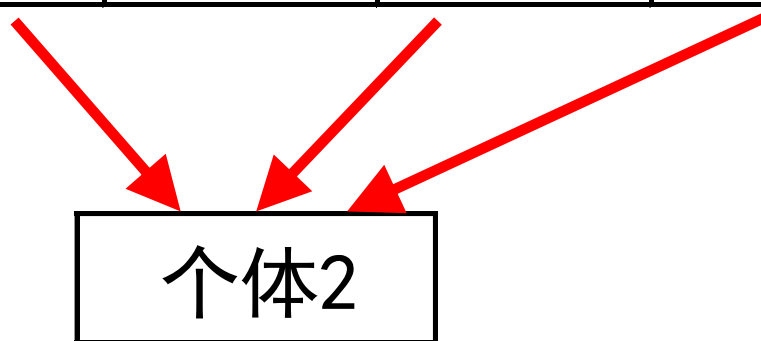
	个体1	个体2	个体3	个体4	个体5	个体6
适应度	2	0.5	1.4	0.8	0.4	1.2
选择概率	0.32	0.08	0.22	0.13	0.06	0.19
累积概率	0.32	0.40	0.62	0.75	0.81	1.00



## □ 选择方法2—竞标赛选择法

- 随机地从种群中挑选一定数目个体，然后将最好的个体选择出来。
- 竞赛规模的取值范围： $[2, \text{个体总数}]$
- 个体适应度可正可负。

	个体1	个体2	个体3	个体4	个体5	个体6
适应度	2	1.4	0.8	0.1	-0.2	-1.3



## □ 选择方法3—排序选择法

- 对群体中的所有个体按其适应度大小进行降序排序，基于这个排序来分配各个个体被选中的概率。
- 适应度可正可负。
- 概率值人为设定，与适应度的大小无关。

	个体1	个体2	个体3	个体4	个体5	个体6
适应度	2	1.4	0.8	0.1	-0.2	-1.3
选择概率	0.3	0.25	0.2	0.15	0.09	0.01



## □ 选择方法4—最佳个体保存法

- 也称为精英保留 (elitist preserve) 策略
- 群体中适应度最高的个体不进行交叉和变异操作，而直接复制到下一代。
- 可使最优解不被破坏。
- 也可能收敛到局部最优。
- 通常与前几种选择方法同时使用。

## □ 选择方法特点

- 种群中出现个别适应度相当高的个体时，会导致其在种群中快速繁殖，使算法快速收敛到局部最优解。
- 当个体的适应度彼此非常接近时，且交叉后的个体变化不大，容易使进化过程陷于停顿，难以找到最优解。
- 解决思路之一：适应度尺度变换。

## □ 选择方法练习

	个体1	个体2	个体3	个体4
适应度	0.2	1	3.1	0.3

### • 选择方法：轮盘赌

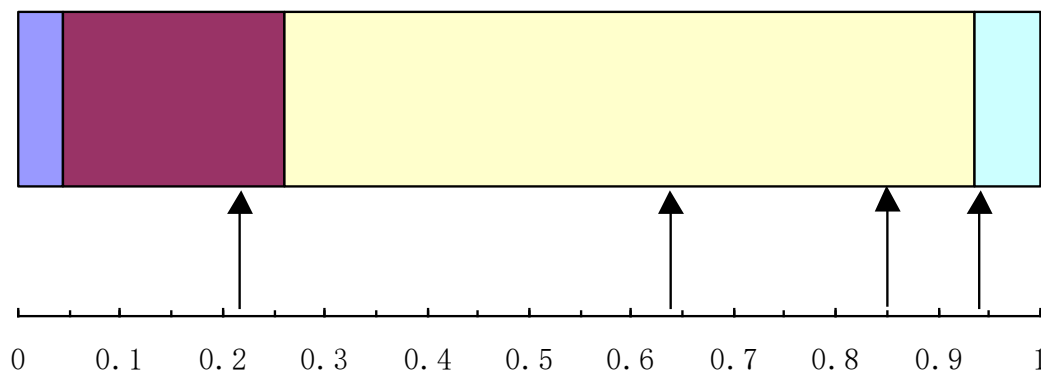
**第一步：**

选择概率	0.04	0.22	0.67	0.07
------	------	------	------	------

**第二步：**

累积概率	0.04	0.26	0.93	1.00
------	------	------	------	------

**第三步：**



随机数	0.22	0.85	0.94	0.64
-----	------	------	------	------

## □ 交叉 (crossover)

是指按照某种方式对选择的父代个体的染色体的部分基因进行交配重组，从而形成新的个体。交配重组是自然界中生物遗传进化的一个主要环节，也是遗传算法中产生新的个体的最主要方法。根据个体编码方法的不同，遗传算法中的交叉操作可分为**二进制交叉**和**实值交叉**两种类型。

## □ 二进制交叉1-单点交叉

- 
- 父个体 X1: 10000111
  - 父个体 X2: 11001010
  - 子个体 Y1: 11000111
  - 子个体 Y2: 10001010

随机选取一个交叉点。

交叉概率 $P_c$ 一般取值较大，建议范围0.4~0.99。

太大，会破坏优良个体；太小，新个体产生速度过慢。

## □ 二进制交叉2-多点交叉

– 父个体 X1: 1000011101101100

– 父个体 X2: 1100101011000101

– 子个体 Y1: 1100011101000101

– 子个体 Y2: 1000101011101100

## □ 二进制交叉3-随机交叉

– 对个体的每一位随机选择交叉与否。

– 父个体 X1: 0000000000000000

– 父个体 X2: 1111111111111111

– 子个体 Y1: 1101010010100010

– 子个体 Y2: 0010101101011101

## □ 实值交叉

$$\begin{cases} Y_1 = \alpha X_1 + (1 - \alpha) X_2 \\ Y_2 = \alpha X_2 + (1 - \alpha) X_1 \end{cases}$$

– 父个体  $x_1$ : 12.3, -5.6

– 父个体  $x_2$ : 23.4, 8.9

– 随机选取  $\alpha=0.4$

– 子个体  $y_1$ : 18.96, 3.1

– 子个体  $y_2$ : 16.74, 0.2

$$Y_1 = 0.4 \times 12.3 + 0.6 \times 23.4$$

$$Y_2 = 0.4 \times 23.4 + 0.6 \times 12.3$$

$$Y_1 = -0.4 \times 5.6 + 0.6 \times 8.9$$

$$Y_2 = 0.4 \times 8.9 - 0.6 \times 5.6$$



## □ 变异 (mutation)

是指对选中个体的染色体中的某些基因进行变动，以形成新的个体。变异也是生物遗传和自然进化中的一种基本现象，它可增强种群的多样性。遗传算法中的变异操作增加了算法的局部随机搜索能力，从而可以维持种群的多样性。根据个体编码方式的不同，变异操作可分为二进制变异和实值变异两种类型。

## □ 二进制变异

– 变异前: 10000111

– 变异后: 10**1**00111

**变异概率 $P_m$ 一般取值较小, 建议范围0.0001~0.1。**

**太大, 会破坏优良个体; 太小, 新个体产生速度过慢。**

## □ 实值变异

– 均匀变异：随机选取  $\alpha$

$$Y = U_{\min} + \alpha \times (U_{\max} - U_{\min})$$

– 高斯变异：随机数符合高斯分布

- 终止代数
  - 进化计算运行到指定代数就终止
  - 一般取值为50-1000
- 判定准则
  - 连续几代的平均适应度的差异小于一个指定阈值

- 最大化  $f(x)=x^2$
- $x$  范围:  $[0, 31]$
- 个体数: 4
- 二进制编码: 5位
- 精度: 1
- 适应度函数:  $f(x)$

$$\delta = \frac{31-0}{2^5-1} = 1$$

## □ 选择

String no.	Initial population	$x$ Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

## □ 交叉

String no.	Mating pool	Crossover point	Offspring after xover	$x$ Value	Fitness $f(x) = x^2$
1	0 1 1 0   1	4	0 1 1 0 0	12	144
2	1 1 0 0   0	4	1 1 0 0 1	25	625
2	1 1   0 0 0	2	1 1 0 1 1	27	729
4	1 0   0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

## □ 变异

String no.	Offspring after xover	Offspring after mutation	$x$ Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	<span style="border: 1px solid red;">1</span> 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 <span style="border: 1px solid red;">1</span> 0 0	18	324
Sum				2354
Average				588.5
Max				729



## □ 简单计算流程

0. 种群初始化

1. 计算适应度，进行选择操作

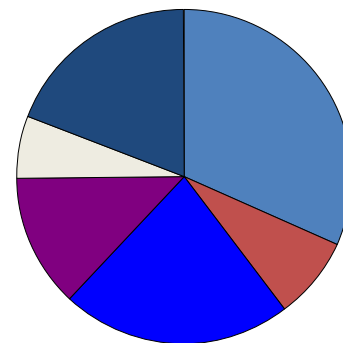
2. 交叉

00111000	→	00111110
11011110		11011000

3. 变异

4. 形成新一代，更新种群

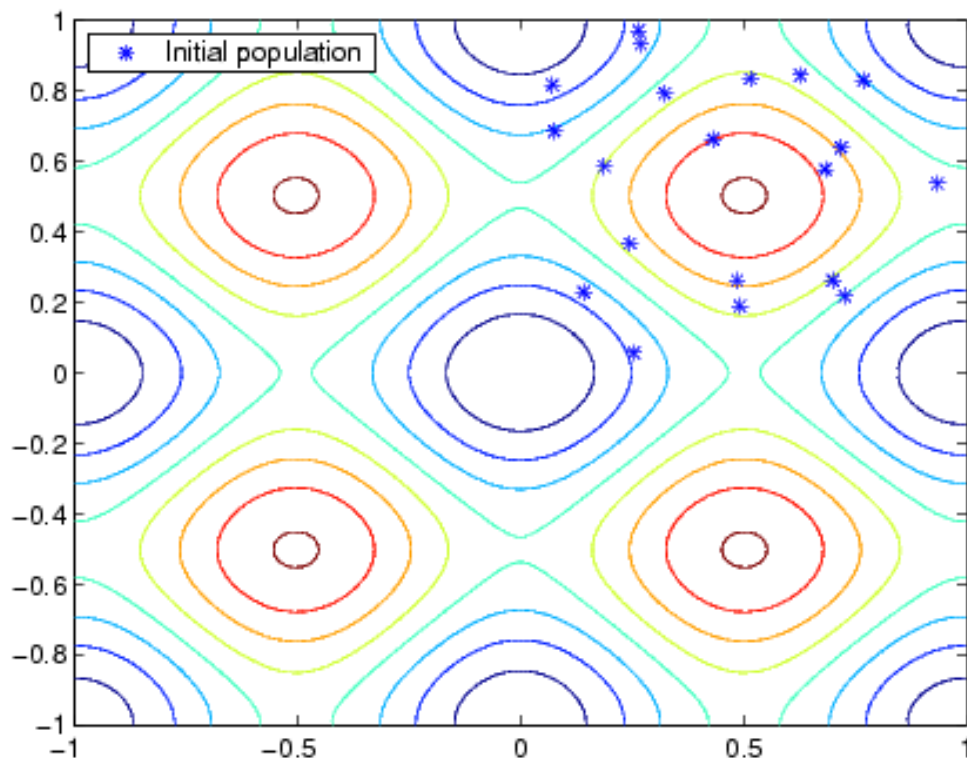
5. 若满足终止条件则结束，否则返回1



00111000  
↓  
00110000

## □ 种群初始化

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$



- 个体数：20
- 数值范围：[0, 1]
- 二进制编码：8位
- 精度：

$$\delta = \frac{U_{\max} - U_{\min}}{2^l - 1} = 0.0039$$

## □ 计算适应度

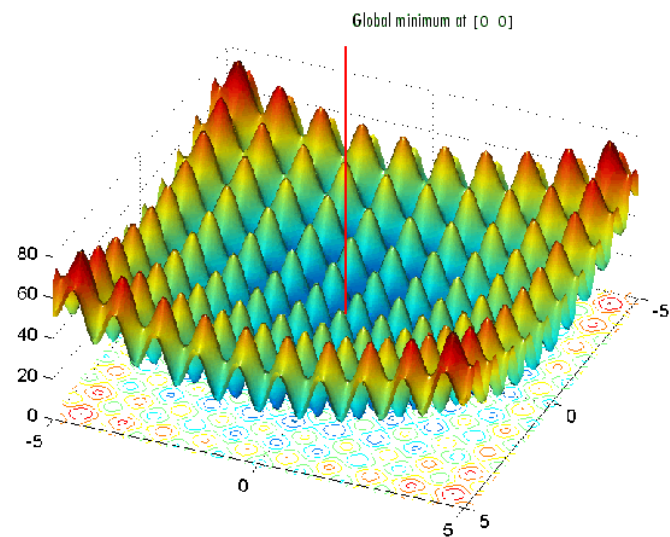
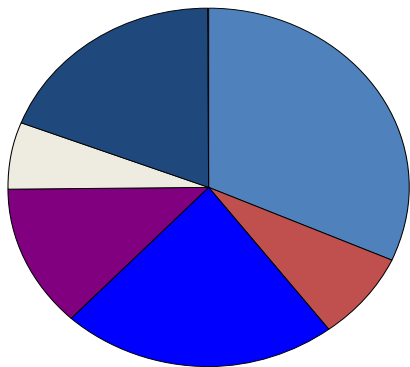
- 计算适应度  $Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$

- 适应度尺度变换

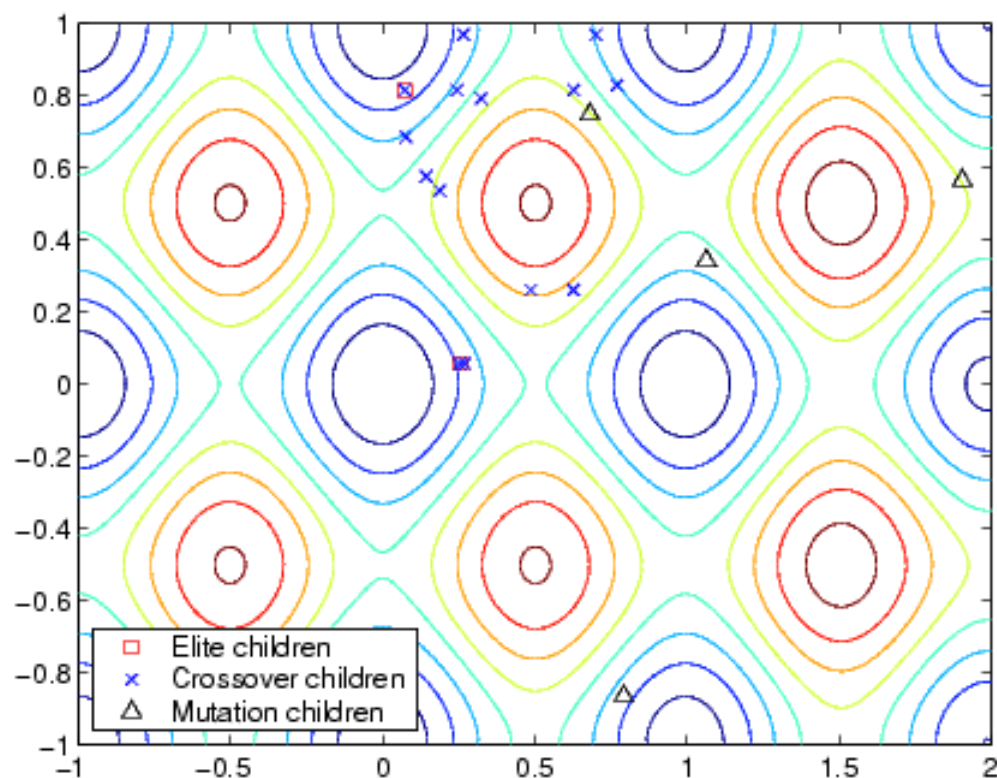
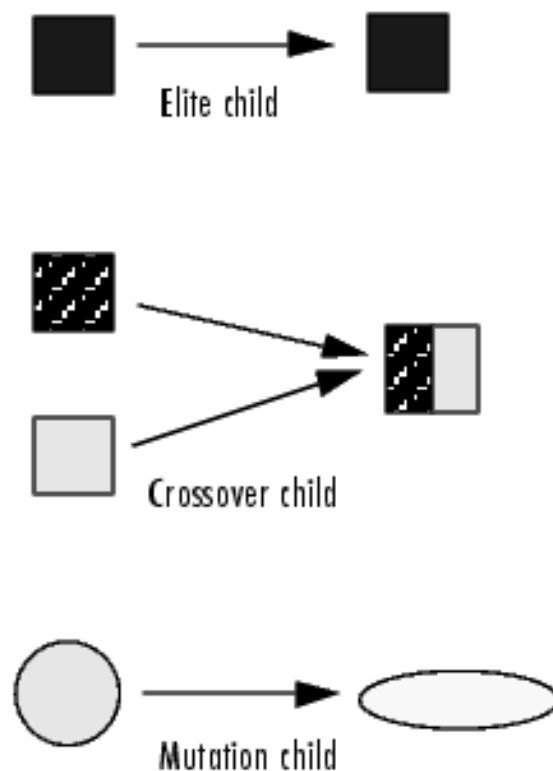
$$f' = e^{-Ras(x)}$$

- 遗传操作—选择:

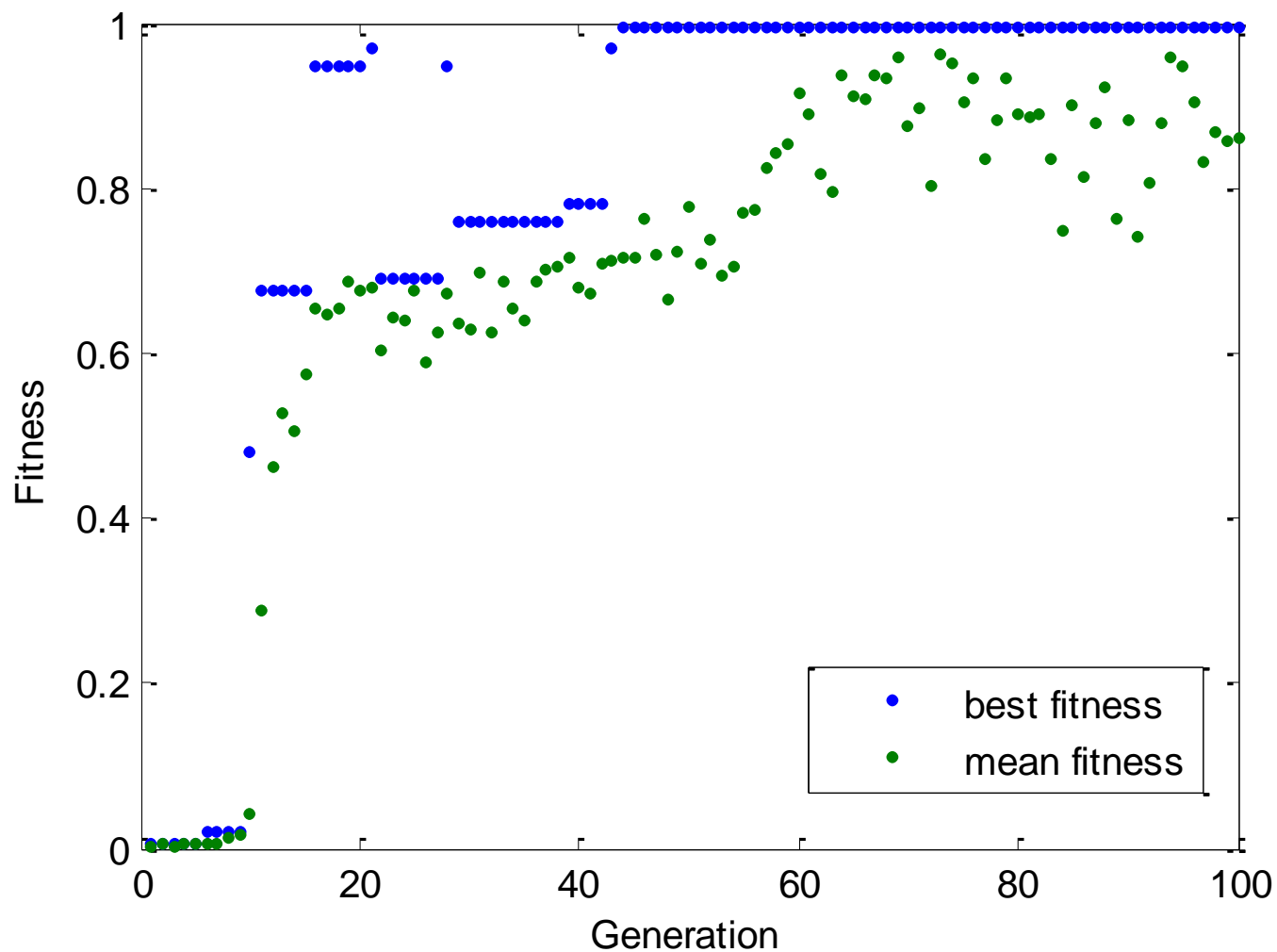
- 适应度大于零，轮盘赌方法



## □ 遗传操作



## □ 适应度变化



## □ 结果

- $(x_1, x_2) = (0.0039, -0.0039)$
- $Ras(x) = 0.0061$
- 特点：
  - 不能保障收敛到全局极值，局部极值
  - 收敛速度慢

## □ 参数讨论

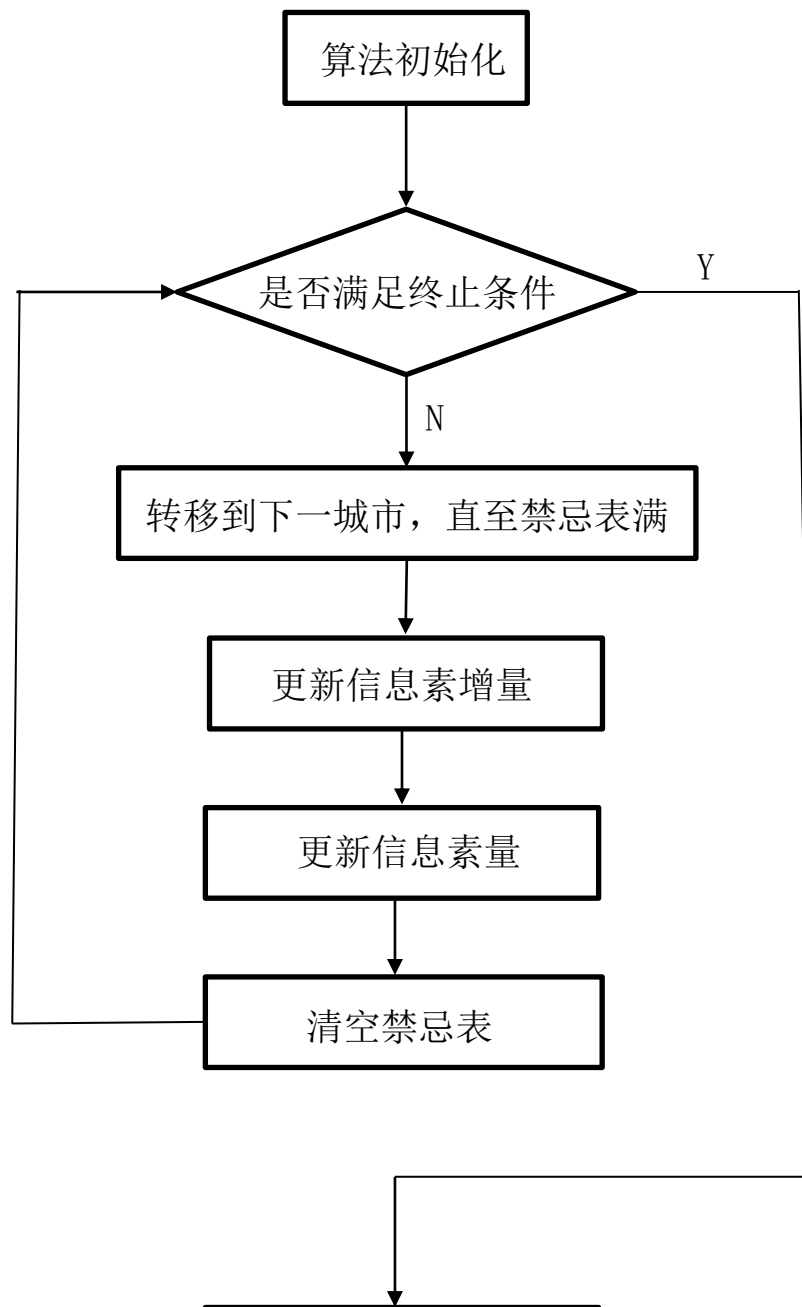
	早熟：提前收敛到局部最优值	收敛慢	建议值
个体数	过小	过大	20~100
适应度函数	差别大	差别小	尺度变换
交叉概率	过大	过小	0.4~0.99
变异概率	过大	过小	0.0001~0.1

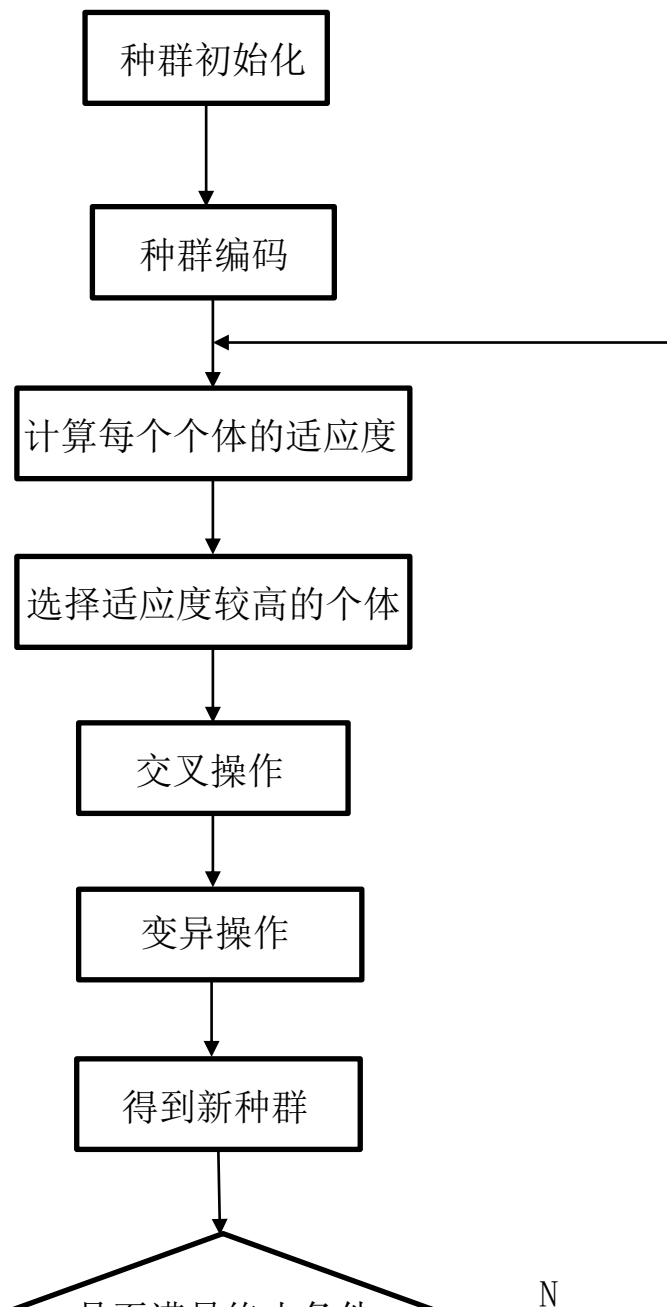
# 三、改进型遗传算法

与基本遗传算法(SGA)相对应

1. 分层遗传算法
2. 混合遗传算法
3. 自适应遗传算法
4. 原对偶遗传算法







## □ 基本思想

模拟生物进化过程中的基因隔离和基因迁移，各子群之间既有相对的封闭性，又有必要的交流和沟通。

- 随机生成 $N \times n$ 个样本，将它们分成 $N$ 个子种群，每个子种群包括 $n$ 个样本。
- 每个子种群独自运行各自的遗传算法。
- 记录每个种群的平均适应度。
- 根据适应度值进行种群间个体的遗传操作。
- 重复第二步骤，若满足收敛条件结束。

## □ 思考

- 同基本遗传算法 (SGA) 的区别
  - 本质：同SGA类似
  - 特点：
    - 子种群内：SGA
    - 子种群间：交叉、变异
    - 分布式并行

## □ 基本思想：与模拟退火的结合

- 模拟退火：将固体加热至融化，然后徐徐冷却使之凝固成规整晶体的热力学过程。
- 即随着温度的降低，物质能量逐渐趋于一个较低的状态，并最终达到某种平衡。

– 接受概率

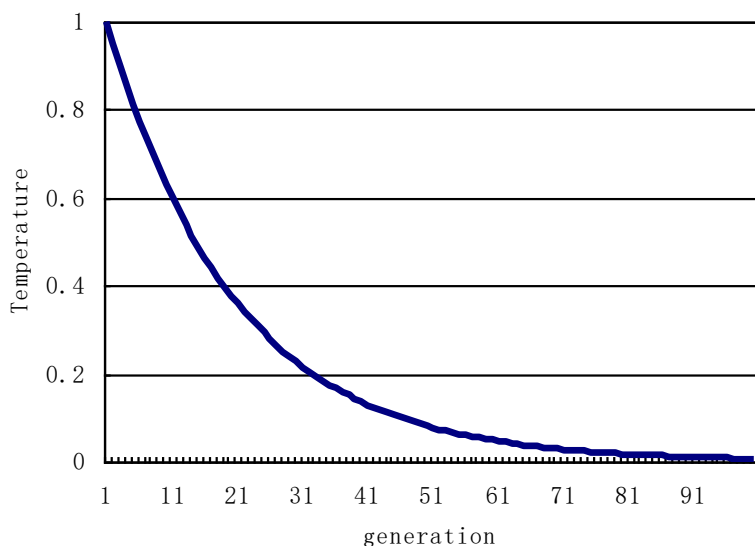
$$p(x) = \frac{1}{z} e^{-\frac{E(x)}{T}}$$

- 遗传过程的特点
  - 前期：个体间适应度差异大，容易早熟
  - 后期：个体间适应度差异小，容易停滞
- 模拟退火的作用：适应度缩放
  - 前期(高温)：减少个体间的适应度差异
  - 后期(低温)：放大个体间的适应度差异，突出优秀个体

## □ 适应度变换公式

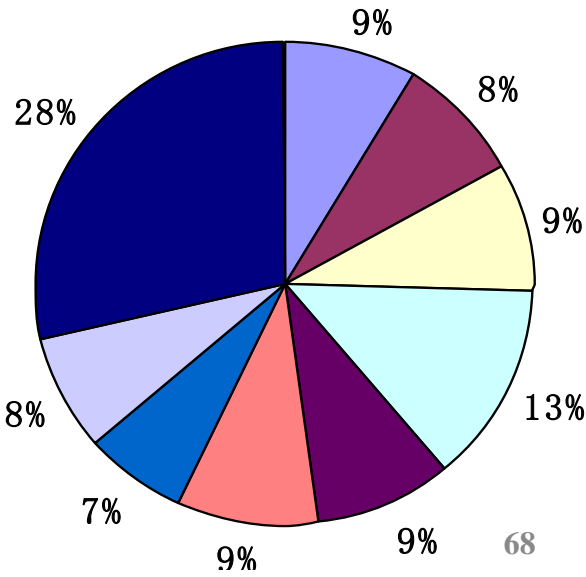
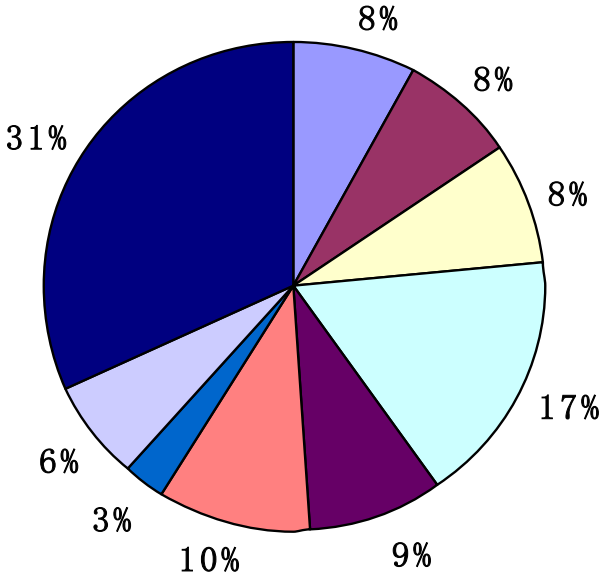
$$f'_i = \frac{e^{\frac{f_i}{T}}}{\sum_{i=1}^M e^{\frac{f_i}{T}}} \quad T = T_0(A^{g-1})$$

- $M$  为种群大小
- $f_i$  为第 $i$ 个个体的适应度
- $g$  为遗传代数
- $T$  为温度
- $T_0$  为初始温度
- $A$  为退火速度( $=0.99$ )



模拟退火前期

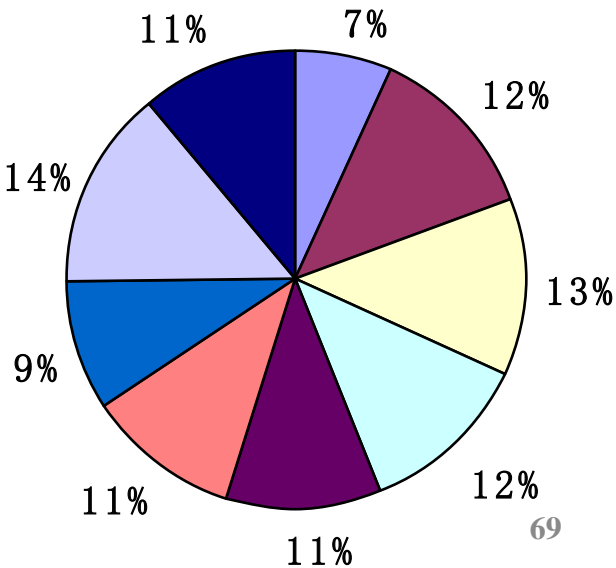
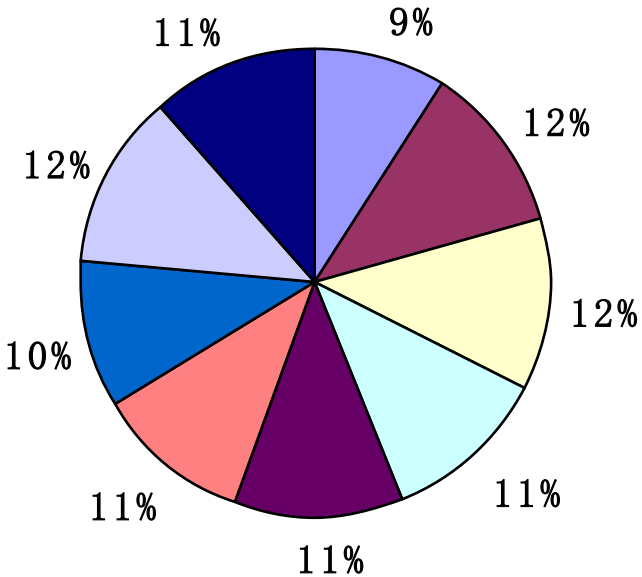
前期适应度	百分比	$e^{f_i/T}$	百分比
0.40	8%	1.49	9%
0.38	8%	1.47	8%
0.40	8%	1.49	9%
0.84	17%	2.31	13%
0.43	9%	1.54	9%
0.48	10%	1.61	9%
0.15	3%	1.16	7%
0.32	6%	1.38	8%
1.60	32%	4.95	28%





□ 模拟退火后期

后期适应度	百分比	$e^{fi/T}$	百分比
0.72	9%	6.93	7%
0.93	12%	12.32	12%
0.93	12%	12.47	13%
0.91	11%	11.71	12%
0.88	11%	10.93	11%
0.88	11%	10.68	11%
0.81	10%	8.95	9%
0.98	12%	14.06	14%
0.89	11%	11.15	11%



## □ 思考

- 同基本遗传算法(SGA)的区别
  - 本质：对适应度进行变换调整
  - 模拟退火的退火速度  $A$ 
    - 快：易振荡
    - 慢：易停滞

## □ 基本思想

- 交叉和变异概率
  - 越大：新个体产生速度越快，而优良个体被破坏的可能性也大
  - 过小：不易产生新个体，使搜索过程停滞
- 方法
  - 交叉和变异的概率随适应度自动改变

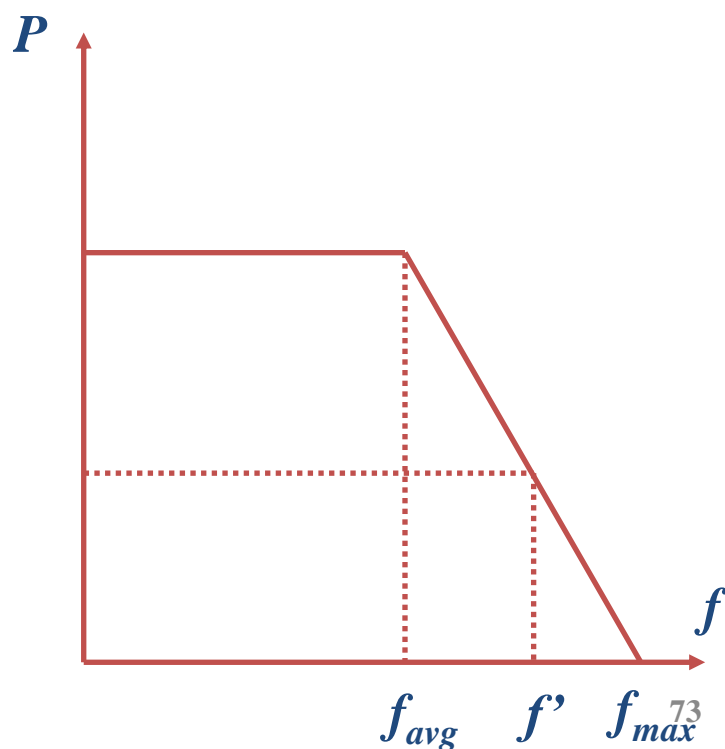
## □ 前期

- 特点：容易收敛到局部最优解
- 方法：
  - 群体适应度趋于一致，增加交叉和变异概率
  - 群体适应度比较分散，减少交叉和变异概率

## □ 后期

- 特点：可能产生最优解
- 方法：
  - 适应度高于平均适应度的个体，减少交叉和变异概率，予以保护
  - 适应度低于平均适应度的个体，增加交叉和变异概率，使其淘汰

$$p = \begin{cases} \frac{k_1(f_{\max} - f')}{f_{\max} - f_{avg}}, & \text{if } f \geq f_{avg} \\ k_2, & \text{if } f < f_{avg} \end{cases}$$



## □ 思考

- 前期：
  - 群体适应度的一致性计算，指导交叉和变异概率的变化
- 后期：
  - 防止收敛到局部最优解，对优良个体给以小的交叉和变异概率

## □ 基本思想

- Primal-dual GA(PDGA), 解决动态优化问题
  - 静态优化问题：适应值或目标函数保持不变
  - 动态优化问题：目标函数随环境等改变而变化
- 定义原染色体和对偶染色体
  - 原-对偶染色体定义为某个距离空间内距离最大的一对染色体
  - 在某种条件下将原染色体用其对偶染色体替代

## □基本方法

- 考虑0-1编码的染色体

原染色体  $x$                       对偶染色体  $x' = \mathbf{dual}(x)$

$\mathbf{dual}(\cdot)$ 为原对偶映射（PDM）函数

- 采用Hamming距离定义PDM函数

-Hamming距离：两个染色体对应基因位点的值不同的基因个数

原染色体  $x = (x_1, x_2, \dots, x_L) \in I = \{0, 1\}^L$

对偶染色体  $x' = \mathbf{dual}(x) = (x'_1, x'_2, \dots, x'_L) \in I, x'_i = 1 - x_i$

- 在一对原对偶染色体中，如果对偶染色体优于原染色体，则这样的映射为有效映射，否则为无效映射



## □基本方法

- PDGA算法流程

- 采用常规操作产生中间种群 $P(t)$
- 从 $P(t)$ 中选择一定数量个体 $D(t)$ ，并计算他们的对偶染色体
- 对于任何 $D(t)$ 中的个体 $x$ ，如果其对偶染色体适应度更大，则 $x$ 被其对偶染色体取代，否则 $x$ 被保留下来

- PDGA优势

- 只有有效的PDM运算才能够使好的对偶染色体有机会传递到下一代，这既有助于增强种群多样性，保持探索能力，又不会影响当前种群的迭代过程，保持种群的利用能力

## □改进方法

- PDGA算法潜在问题

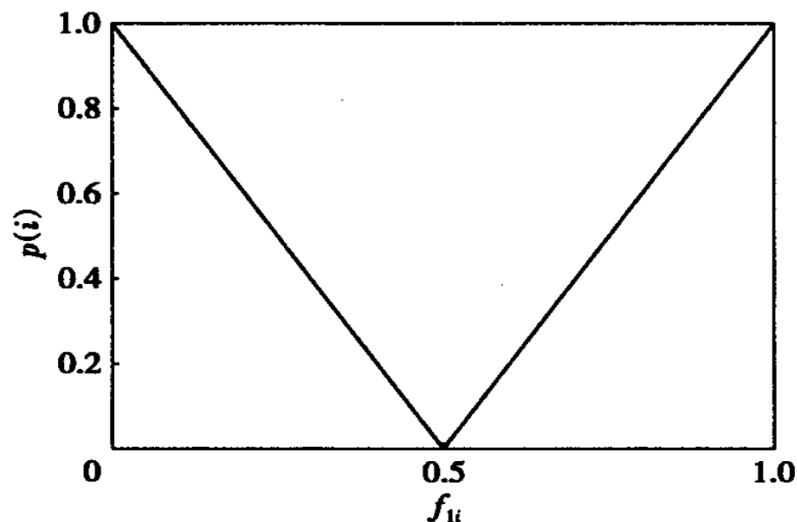
- 在基本算法中，一对原对偶染色体基因型截然相反，在种群进化后期，大多数染色体都分布在适应值较大区域，其对偶染色体适应值较低，因此大多数PDM运算都会失效，因此难以产生多样性和最好的解，但却会浪费计算资源
- 如果环境的变化较小，这样过于“强烈”的PDM运算不会明显改善算法性能

- 改进思路

- 并不让染色体中每一位基因都参与PDM运算，而是按概率决定是否参与运算

## 改进方法

- 设 $p(i)$ 为基因位点 $i$ 参与 PDM运算的概率,  $f_{si}$ 为 在种群中符号 $s$ 在基因位点 $i$ 上的出现频率,  $s$ 为基因位点的编码取值
- 对于0-1编码空间, 存在 $f_{0i} + f_{1i} = 1$
- $f_{1i}$ 即是基因位点 $i$ 取值趋向于“1”的程度, 如果所有 $f_{1i}$ 都趋近于1或者0, 则整个种群就趋向于收敛
- 设计 $p(i)$ 为 $f_{1i}$ 的函数



# 四、应用实例

## 旅行商问题求解

## □ 问题描述

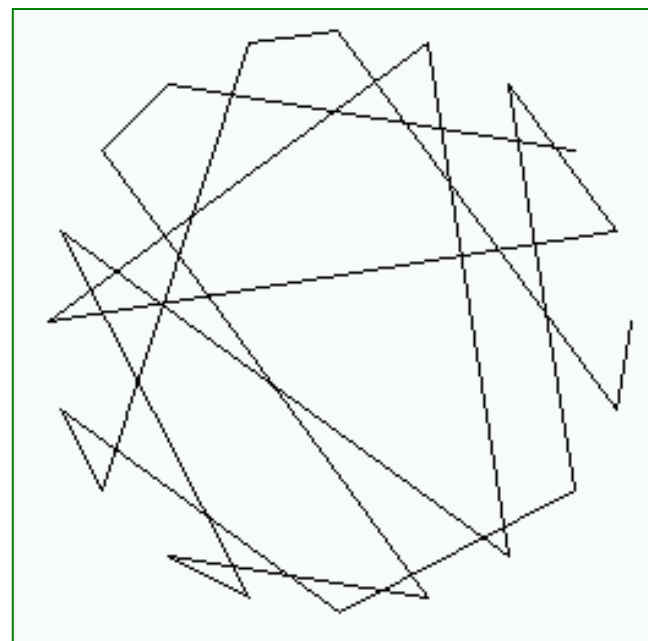
- 已知：
  - $N$ 个城市间的相互距离，现有一推销员必须遍访 $N$ 个城市，并且每个城市只能访问一次，最后又必须返回出发城市
- 问题：
  - 如何安排对这些城市的访问顺序，可使其旅行路线的总长度最短？

## □ 数学描述

- 对于城市  $V = \{v_1, v_2, \dots, v_n\}$  的一个访问顺序为  $T = \{t_1, t_2, \dots, t_n\}$ , 且  $t_{n+1} = t_1$ , 则旅行商问题的数学模型为

$$\min L = \sum_{i=1}^N d_{t_i, t_{i+1}}$$

- 典型的非线性规划(NP)问题



## □ 编码方法

### ➤ 符号编码

$$W = (v_1, v_2, v_3, \dots, v_n)$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$T = (1, 2, 3, \dots, n)$$

但交叉和变异操作困难!

### ➤ Grefenstette的编码方法

每访问过一个城市，就从城市列表中将该城市去掉，利用第*i*个城市在所有未访问的城市列表中的对应位置序号，记为*G*。

## ➤ Grefenstette的编码方法

- 设城市列表:  $W = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10})$
- 设巡回路线:  $T = (t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10})$
- 规定每访问完一个城市, 则从城市列表W中将该城市去掉。
- 用第 $i$  ( $i=1, 2, 3, \dots, 10$ ) 个所访问的城市在所有未访问的城市列表 $W - \{t_1, t_2, t_3, \dots, t_{i-1}\}$ 中的对应位置序号 $g_i$  ( $1 \leq g_i \leq n-i+1$ ), 可以表示具体访问哪个城市。
- 将全部按顺序排列:
$$G = (g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10})$$
- G完整表示了该巡回路线, 可作为遗传算法中的个体。



## □ Grefenstette编码方法举例

- 十个城市：

- 巡回路线为  $T = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$

- 编码为  $G = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$

- 巡回路线为  $T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$

- $T_y = (2, 3, 1, 4, 5, 10, 8, 9, 6, 7)$

- 编码为

- $G_x = (1, 3, 1, 5, 3, 4, 4, 3, 2, 1)$

- $G_y = (2, 2, 1, 1, 1, 5, 3, 3, 1, 1)$

## □ 交叉算子

### • 单点交叉

– Grefenstette 1985

### • 巡回路线

**交叉点左侧部分的旅程没有变化**

$$G_x = (1, 3, 1, 5, 3, 4, 4, 3, 2, 1)$$

$$G_y = (2, 2, 1, 1, 1, 5, 3, 3, 1, 1)$$

$$G'_x = (1, 3, 1, 5, 3, 4, 3, 3, 1, 1)$$

$$G'_y = (2, 2, 1, 1, 1, 5, 4, 3, 2, 1)$$

$$T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$

$$T_y = (2, 3, 1, 4, 5, 10, 8, 9, 6, 7)$$

$$T'_x = (1, 4, 2, 8, 6, 9, 7, 10, 3, 5)$$

$$T'_y = (2, 3, 1, 4, 5, 10, 9, 8, 7, 6)$$

## • 部分匹配交叉PMX

- Goldberg 1985
- 两个交叉点

思路：

- 先随机产生两个交叉点，定义这两点间的区域为匹配区域，交换两个父代的匹配区域
- 对于匹配区域外出现的重复数码，要依据匹配区域内的映射关系逐一替换

匹配区域的映射关系：

$4 \leftrightarrow 8 \leftrightarrow 10 \leftrightarrow 9 \quad 6 \leftrightarrow 5$

交叉

$$T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$

$$T_y = (2, 3, 1, 4, 5, 10, 8, 9, 6, 7)$$

$$T'_x = (*, *, *, 4, 5, 10, 8, *, *, *)$$

$$T'_y = (*, *, *, 8, 6, 9, 10, *, *, *)$$

保留

$$T'_x = (1, *, 2, 4, 5, 10, 8, 7, *, 3)$$

$$T'_y = (2, 3, 1, 8, 6, 9, 10, *, *, 7)$$

映射

$$T'_x = (1, *, 2, 4, 5, 10, 8, 7, 6, 3)$$

$$T'_y = (2, 3, 1, 8, 6, 9, 10, *, 5, 7)$$

填空

$$T'_x = (1, 9, 2, 4, 5, 10, 8, 7, 6, 3)$$

$$T'_y = (2, 3, 1, 8, 6, 9, 10, 4, 5, 7)$$

## • 顺序交叉OX

– Davis 1985

思路：

- 从父代 $T_x$ 随机选取一个编码子串，放到子代 $T'_x$ 的对应位置，子代 $T'_x$ 空余的位置从父代 $T_y$ 中按 $T_y$ 的顺序选取（与已有编码不重复）
- 同理可得 $T_y$ 的子代 $T'_y$

交叉后

$$\begin{array}{l}
 T_x = (1, 4, 2, \mid 8, 6, 9, 10, \mid 7, 5, 3) \\
 T_y = (2, 3, 1, \mid 4, 5, 10, 8, \mid 9, 6, 7) \\
 \hline
 T'_x = (2, 3, 1, \mid 8, 6, 9, 10, \mid 4, 5, 7) \\
 T'_y = (1, 2, 6, \mid 4, 5, 10, 8, \mid 9, 7, 3)
 \end{array}$$

## • 循环交叉CX

– Oliver 1987

思路：

CX与OX的不同之处在于：OX中来自第一个亲代的编码子串是随机产生的，而CX却不是，它是根据两个双亲相应位置的编码而确定的

- 从父代 $T_x$ 出发获取循环基因，用循环基因构成子代 $T'_x$ ，顺序与父代一样，用父代 $T_y$ 剩余的基因填满子代 $T'_x$
- 从父代 $T_y$ 出发获取循环基因，用类似的操作构成子代 $T'_y$

交叉后

$$T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$

$$T_y = (2, 3, 6, 5, 1, 10, 8, 9, 4, 7)$$

$$T'_x = (1, 3, 2, 5, 6, 10, 8, 9, 4, 7)$$

$$T'_y = (2, 4, 6, 8, 1, 9, 10, 7, 5, 3)$$

循环基因：

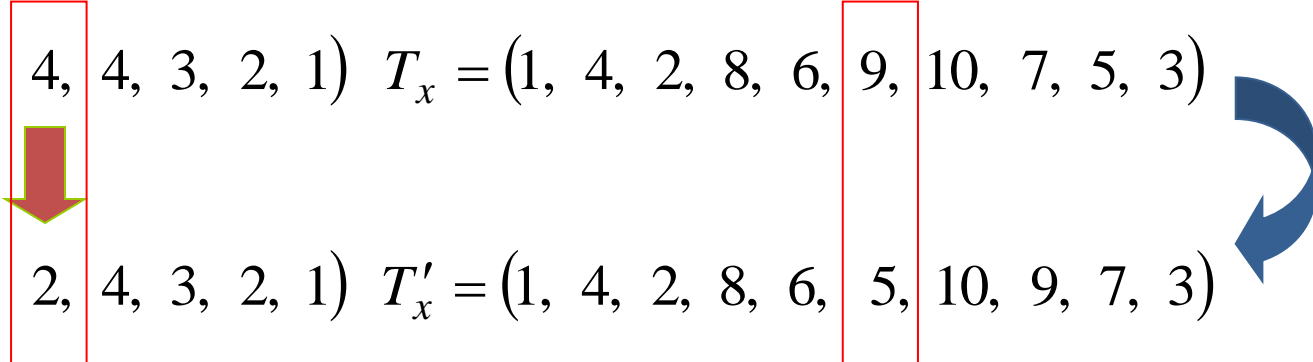
$1 \rightarrow 2 \rightarrow 6 \rightarrow 1$

$2 \rightarrow 1 \rightarrow 6 \rightarrow 2$

## □ 变异算子

- 常规变异

- 变异后的基因值只能从 $(1, 2, 3, \dots, n-i+1)$ 中取

$$G_x = (1, 3, 1, 5, 3, 4, 4, 3, 2, 1) \quad T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$

$$G'_x = (1, 3, 1, 5, 3, 2, 4, 3, 2, 1) \quad T'_x = (1, 4, 2, 8, 6, 5, 10, 9, 7, 3)$$

**1,4,2,8,6,10,9,7,5,3**

- 逆转变异

- 方式1：随机选择两点，将两点间的子串按反序插入到原位置中。

$$T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$



$$T'_x = (1, 4, 2, 10, 9, 6, 8, 7, 5, 3)$$

- 方式2：随机选择两个逆转点，将两点的基因交换

$$T_x = (1, 4, 2, 8, 6, 9, 10, 7, 5, 3)$$



$$T'_x = (1, 4, 7, 10, 9, 6, 8, 2, 5, 3)$$

## TSP —应用思考？

- 图论中的极值问题
  - 货物配送路线
  - 电路板钻孔方案
  - 机器人路径规划



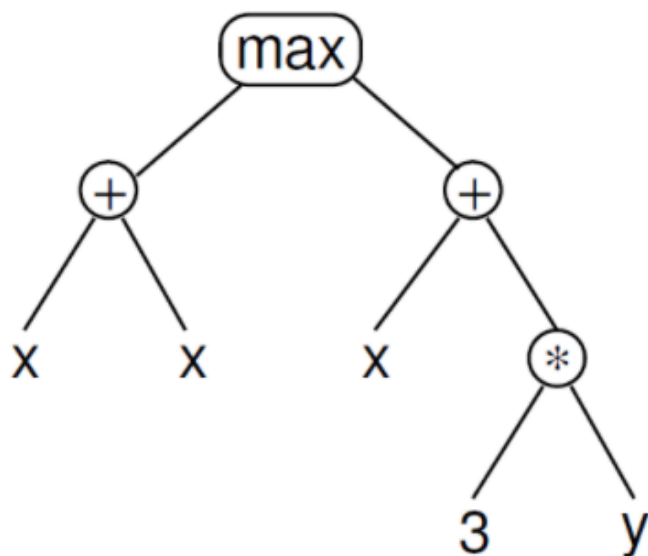
拓展：遗传规划/遗传编程

**Genetic Programming**

- GP的基本思想
  - GP继承了GA的基本思想，即从父代中择优产生子代，基本操作也包括复制、选择、交叉、变异等
  - 不同于GA的传统编码（固定长度基因）模式，GA的个体通常为计算机程序，具备多样的表现形式，如层次化的树、图等
- GP的颠覆性
  - 传统遗传优化：进化的基本单位是模型可变参数
  - GP：进化的基本单位是新算法以及新算法的参数
- 参考文献
  - Poli, Riccardo, et al. *A Field Guide to Genetic Programming*. Lulu Press, 2008.
  - J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

## • 树状结构表示

- 树叶节点为终止符(Terminals), 是程序中得到变量、常量、无参函数
- 树枝节点为应用在其子节点上的某种操作或函数 (Functions)
- 例: 用树状结构表示程序 $\max(x+x, x+3*y)$

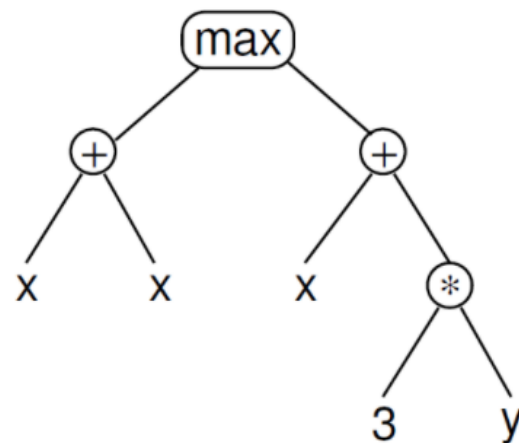


- Terminal set= $\{x, y, 3\}$
- Function set= $\{+, *, \max\}$
- Primitive set为上述二者的合集

## • 初始化

- 给定函数集、终止集和参数（种群数量、决策树深度等）
- 将随机生成个体的事件视作从函数集和终止集随机取得元素，创建节点(根节点root或子节点)并为其指定连接的随机子节点的随机过程
- 决策树深度

- **The depth of a node** is the number of edges that need to be traversed to reach the node starting from the tree's root node (which is assumed to be at depth 0).
- **The depth of a tree** is the depth of its deepest leaf.

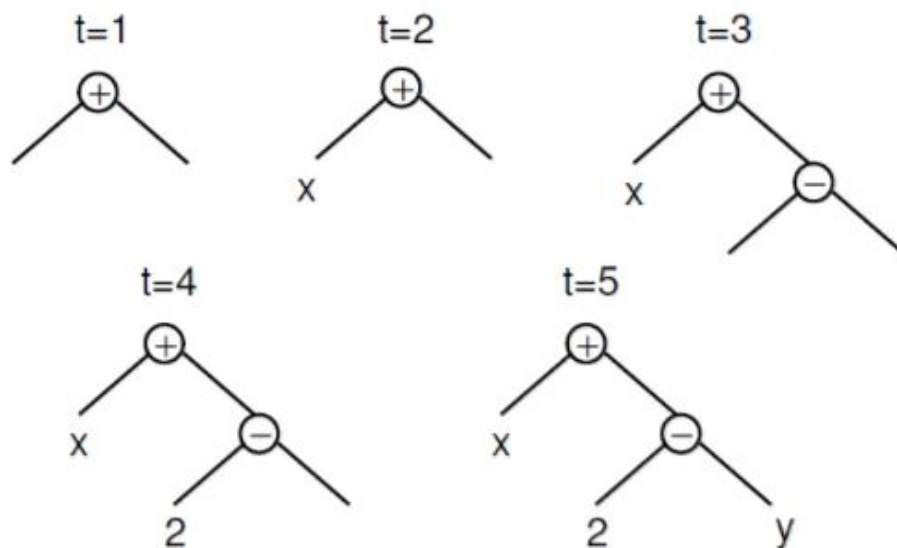


- 根节点max的深度为0
- 子节点\*的深度2
- 叶节点3的深度3

## • 节点随机生成方法

**-Grow方法：**从函数集中随机选择函数作为根节点，从整个 primitive set(函数集+终止集)中随机选择作为子节点，直到达到限制的深度（或因可连接节点全部随机选择到terminals或无参函数而自动停止），达到限制的深度时只能从终止集 (terminal set)中随机取用终止符作为leaves。

- 设函数集 $F=\{+, -, *, /\}$
- 终止集 $T=\{x, y, 0, 1, 2, 3\}$
- 给定深度 $depth=2$

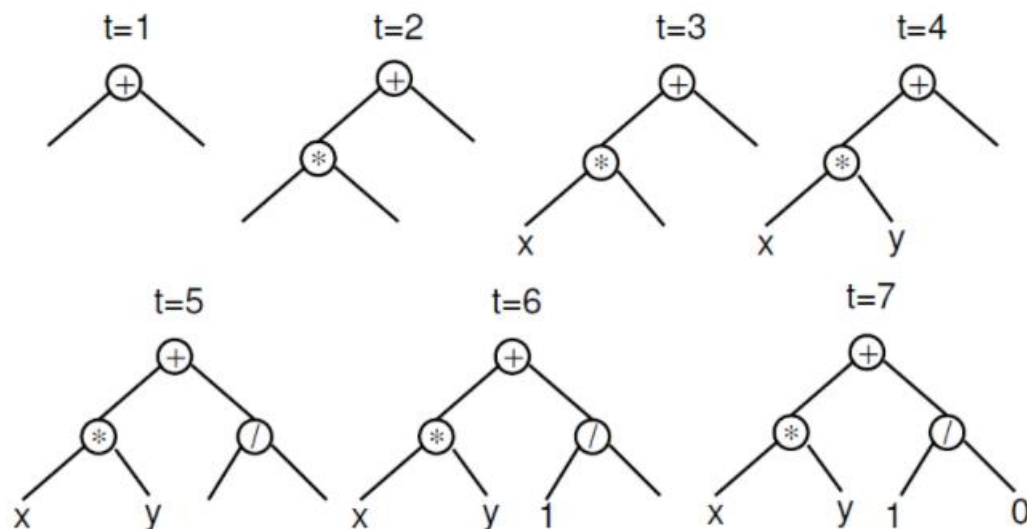


Grow生成的个体有着更为多样化的大小和形状

## • 节点随机生成方法

–**Full方法**：从函数集中随机取出函数作为根节点，从函数集中随机取出函数作为子节点，直到达到深度限制，达到指定深度时只能从终止集中随机取用终止符作为leaves。

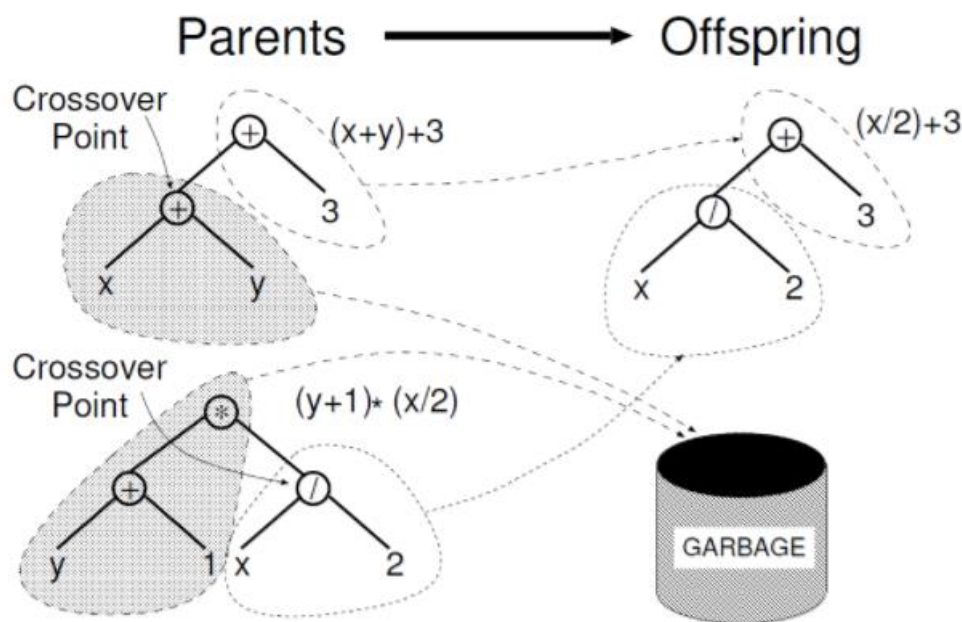
- 设函数集  $F = \{+, -, *, /\}$
- 终止集  $T = \{x, y, 0, 1, 2, 3\}$
- 给定深度  $\text{depth} = 2$



Full生成的个体具有相同的深度（由于节点的度不同，子节点数量有所不同）

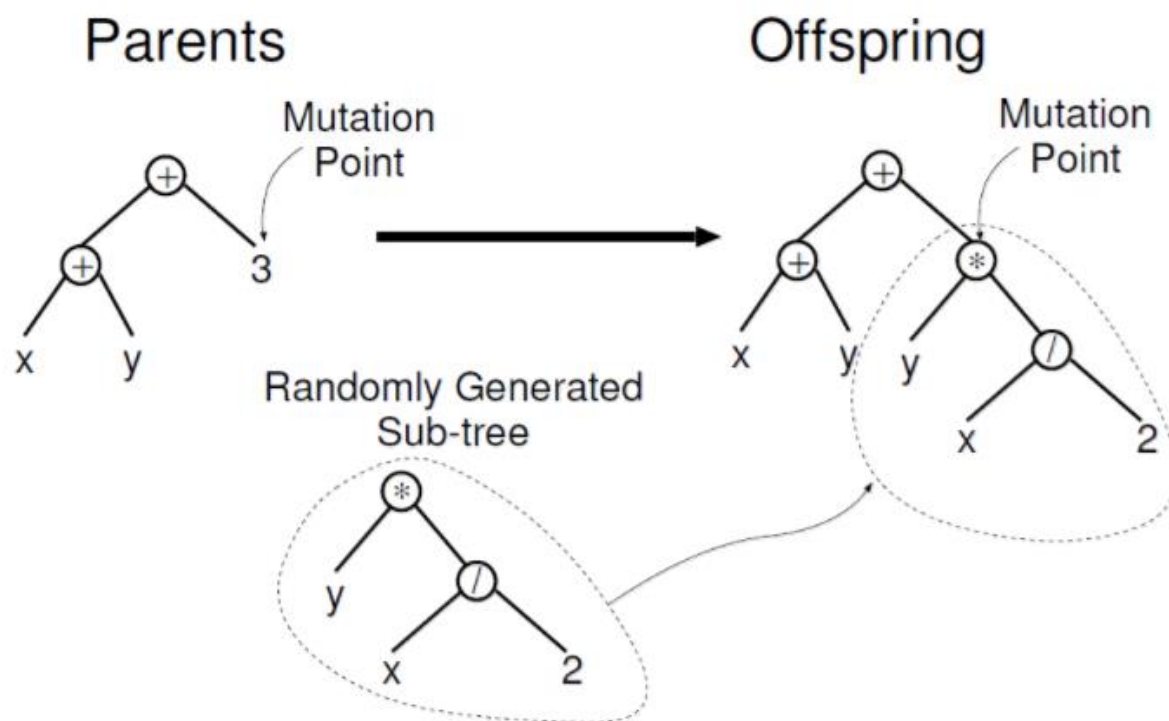
## • 交叉操作

–常用的一种subtree crossover过程：在每一个父体中随机选择一个杂交点(crossover point)，复制其中一个父体的树为 Acopy，复制第二个父体中以杂交点为根节点的子树(subtree)为 Bsubtree\_copy，将 Acopy 杂交点下的子树替换为 Bsubtree\_copy，生成子体。



## • 变异操作

–常用的一种subtree mutation：在一个父体中随机选择一个突变点(mutation point)，随机生成一个子树，将父体中以突变点为根节点的子树替换为这个随机生成的子树。





## • 算法流程

- 初始化：在给定初始条件（包括terminal sets, function sets和参数）后生成随机种群
- 通过(多种方法)比较，评估适应性(fitness evaluation)
- 依据fitness进行概率性选择（probabilistically selected based on fitness）
- 被选择的程序作为父系，通过交叉(Crossover)、变异(Mutation)、复制(Reproduction)等遗传算子(genetic operators)生成下一代(next generation)
- 判断是否符合终止标准(Termination Criterion)，不符合则继续迭代

应用：生成能够拟合特定数据集的函数（与现有方法的对比分析？）