最优化方法

第十讲 智能优化算法

智能优化算法简介

一、 传统优化算法的步骤及局限性

1 步骤：

（1）选择一个初始解，

（2）向改进方向移动判断停止准则是否满足，若满足停止，否则转下一步。

（3）向改进方向移动，得新的解，转回第2步。

2 局限性：

（1）单点运算方式限制了计算效率的提高

（2）向改进方向移动限制了跳出局部最优的能力

（3）停止条件仅是局部最优的条件

（4）对目标函数，约束条件的要求限制了算法的应用

智能优化算法简介

二、智能优化算法的产生与发展

1 最优化方法的新的需求

（1）对目标函数，约束函数的要求更为宽松

（2）计算效率比理论上的最优性更重要

（3）算法随时终止都能得到较好的解

（4）对优化模型中数据质量要求更加宽松。

2 智能算法及代表人物

1975年，Holland提出遗传算法(Genetic Algorithms)

1977年，Glover提出禁忌算法(Tabu Search)

1983年，Kirkpatrick提出模拟退火算法 （Simulated Annealing）

90年代初，Dorigo提出蚁群算法 (Ant Colony Optimization)

1995年，Kennedy,Eberhart提出的粒子群算法(Particle Swarm)

1999年，Linhares 提出的捕食搜索(Predatory Search)

智能优化算法简介

三、如何学习研究智能优化算法

1 应用智能优化方法解决各类问题是重点

2 智能算法的改进有很大的空间

3 多种算法结合是一种很好的途径

4 不提倡刻意追求理论成果

5 算法性能的测试是一项要下真功夫的工作

6 创造出新算法

常用的智能优化算法

- 遗传算法
- 粒子群算法
- 蚁群算法
- 模拟退火算法
- 禁忌搜索算法……
- AlphaGo

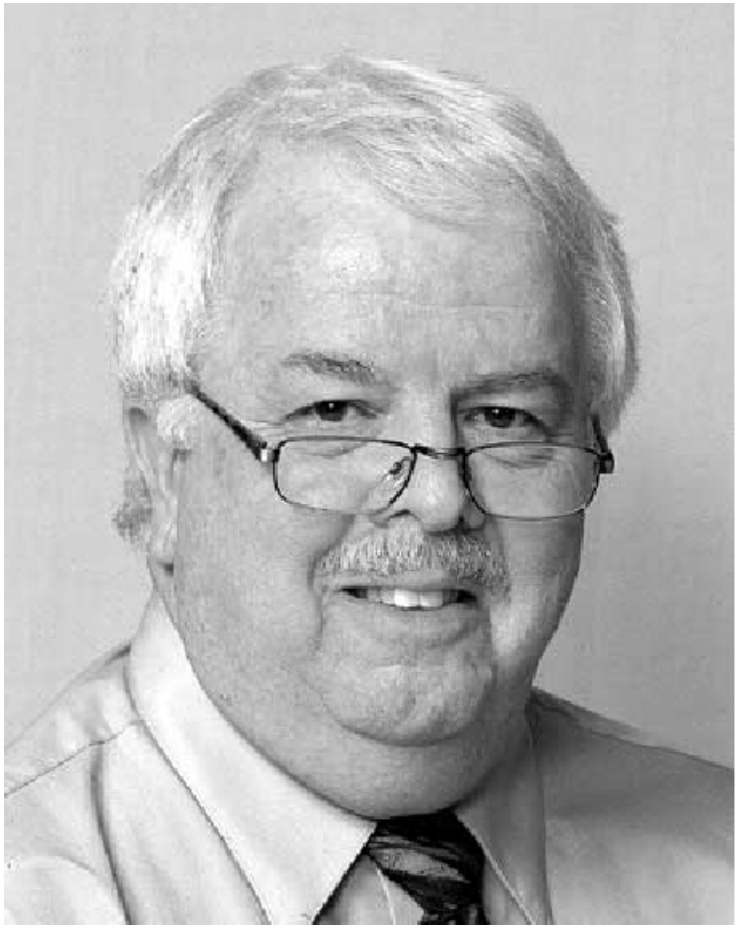
智能优化算法的特点

- 从任一解出发，按照某种机制，以一定的概率在整个求解空间中探索最优解。
- 求解的过程通常是一个迭代的过程，即不是一步就完成。
- 由于它们可以把搜索空间扩展到整个问题空间，因而具有全局优化性能。

粒子群算法简介

粒子群算法(particle swarm optimization，PSO)由Kennedy和Eberhart在1995年提出，该算法模拟鸟集群飞行觅食的行为，鸟之间通过集体的协作使群体达到最优目的，是一种基于Swarm Intelligence的优化方法。同遗传算法类似，也是一种基于群体叠代的，但并没有遗传算法用的交叉以及变异，而是粒子在解空间追随最优的粒子进行搜索。PSO的优势在于简单容易实现同时又有深刻的智能背景，既适合科学研究，又特别适合工程应用，并且没有许多参数需要调整。

James Kennedy received the Ph.D. degree from theUniversity of North Carolina, Chapel Hill, in 1992.He is with the U.S. Department of Labor, Washington,DC. He is a Social Psychologist who has been working with the particle swarm algorithm since 1994. He has published dozens of articles and chapters on particle swarms and related topics, in computer science and social science journals and proceedings. He is a coauthor of *Swarm Intelligence* (San Mateo, CA: Morgan Kaufmann, 2001), with R.C. Eberhart and Y. Shi, now in its third printing.



Russell C. Eberhart (M'88–SM'89–F'01) received the Ph.D. degree in electrical engineering from Kansas State University, Manhattan.He is the Chair and Professor of Electrical and Computer Engineering, Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI),Indianapolis, IN. He is coeditor of *Neural Network PC Tools*(1990),coauthor of *Computational Intelligence PC Tools (*1996), coauthor of *Swarm Intelligence*(2001), *Computational Intelligence: Concepts to Implementations*(2004). He has published over 120 technical papers.Dr. Eberhart was awarded the IEEE Third Millenium Medal. In 2002, he became a Fellow of the American Institute for Medical and Biological Engineering.

基本PSO算法

粒子群优化算法源于1987年Reynolds对鸟群社会系统boids的仿真研究，boids是一个CAS。在boids中，一群鸟在空中飞行，每个鸟遵守以下三条规则：

1）避免与相邻的鸟发生碰撞冲突；

2）尽量与自己周围的鸟在速度上保持协调和一致；

3）尽量试图向自己所认为的群体中靠近。

仅通过使用这三条规则，boids系统就出现非常逼真的群体聚集行为，鸟成群地在空中飞行，当遇到障碍时它们会分开绕行而过，随后又会重新形成群体。

基本PSO算法（续）

Reynolds仅仅将其作为CAS的一个实例作仿真研究，而并未将它用于优化计算中 。

Kennedy和Eberhart在中加入了一个特定点，定义为食物，鸟根据周围鸟的觅食行为来寻找食物。他们的初衷是希望通过这种模型来模拟鸟群寻找食源的现象，然而实验结果却揭示这个仿真模型中蕴涵着很强的优化能力，尤其是在多维空间寻优中。

基本PSO算法(续)

PSO中，每个优化问题的解都是搜索空间中的一只鸟。称之为“粒子(Particle)”。所有的粒子都有一个由被优化的函数决定的适应值，每个粒子还有一个速度决定他们飞翔的方向和距离。然后粒子们就追随当前的最优粒子在解空间中搜索.

PSO 初始化为一群随机粒子。然后通过叠代找到最优解。在每一次叠代中，粒子通过跟踪两个"极值"来更新自己。第一个就是粒子本身所找到的最优解。这个解叫做个体极值*pBest*. 另一个极值是整个种群目前找到的最优解。这个极值是全局极值*gBest*。另外，也可以不用整个种群而只是用其中一部分的邻居。

基本PSO算法(续)

PSO算法数学表示如下：

设搜索空间为D维，总粒子数为n。第i个粒子位置表示为向量$X_i=(x_{i1}, x_{i2}, \ldots, x_{iD})$；第$i$个粒子“飞行”历史中的过去最优位置（即该位置对应解最优）为$P_i=(p_{i1}, p_{i2}, \ldots, p_{iD})$，其中第$g$个粒子的过去最优位置$P_g$为所有$P_i$ ($i=1, \ldots, n$)中的最优；第i个粒子的位置变化率（速度）为向量$V_i=(v_{i1}, v_{i2}, \ldots, v_{iD})$。每个粒子的位置按如下公式进行变化（“飞行”）：

基本PSO算法(续)

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times rand() \times [p_{id}(t) - x_{id}(t)] + c_2 \times rand() \times [p_{gd}(t) - x_{id}(t)] \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

$$1 \le i \le n \quad 1 \le d \le D$$

其中，C*1,C2*为正常数，称为加速因子；*rand*( )为[0，1]之间的随机数；*w*称惯性因子，*w*较大适于对解空间进行大范围探查(exploration)，*w*较小适于进行小范围开挖(exploitation)。第*d*（1≤*d*≤D）维的位置变化范围为[-XMAXd , XMAXd]，速度变化范围为[-VMAXd , VMAXd]，迭代中若位置和速度超过边界范围则取边界值。

基本PSO算法(续)

粒子群初始位置和速度随机产生，然后按公式(1)(2)进行迭代，直至找到满意的解。目前，常用的粒子群算法将全体粒子群(Global)分成若干个有部分粒子重叠的相邻子群，每个粒子根据子群(Local)内历史最优P_l调整位置，即公式(2)中P_{gd}换为P_{ld}。

基本粒子群算法

一、 基本粒子群算法的构成要素

1 粒子数

- 2 最大速度
- 3 学习因子
- 4 惯性权重

基本粒子群算法

二、基本粒子群算法步骤

1 随机初始化种群中各种微粒的位置和速度；

2 评价每个微粒的适应度,将当前各微粒的位置和适应值存储在各微粒的pbest中,将所有pbest中适应值最优个体的位置和适应值存于gbest中；

3 用下面的式子更新粒子的速度和位移；

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1[p_{ij} - x_{ij}(t)] + c_2 r_2[p_{gj} - x_{ij}(t)],$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \qquad j = 1,2,...,d$$

4 对每个微粒子;将其适应值与其经历过的最好位置作比较,如果较好,则将其作为当前的最好位置；

5 比较当前所有pbest和gbes的值，更新gbest；

6 若满足停止条件（通常为预设的运算精度或迭代步数），搜索停止，输出结果，否则返回3，继续搜索。

遗传算法概述

- 美国J. Holland教授于1975年在专著《自然界和人工系统的适应性》中首先提出。
- 借鉴生物界自然选择和自然遗传机制的随机化搜索算法。
- 模拟自然选择和自然遗传过程中发生的繁殖、交叉和基因突变现象。

遗传算法概述

- 在每次迭代中都保留一组候选解，并按某种指标从解群中选取较优的个体，利用遗传算子(选择、交叉和变异)对这些个体进行组合，产生新一代的候选解群，重复此过程，直到满足某种收敛指标为止。
- 基本遗传算法（Simple Genetic Algorithms，GA)又称简单遗传算法或标准遗传算法），是由Goldberg总结出的一种最基本的遗传算法，其遗传进化操作过程简单，容易理解，是其它一些遗传算法的雏形和基础。

基本遗传算法的组成

（1）编码（产生初始种群）

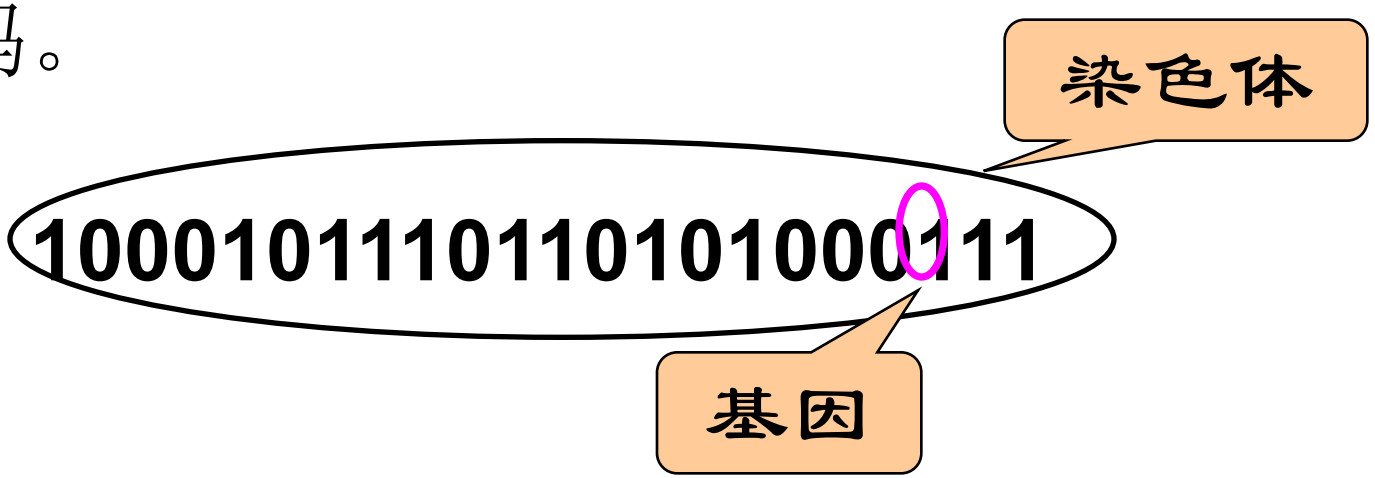
（2）适应度函数

（3）遗传算子（选择、交叉、变异）

（4）运行参数

编码

- 遗传算法（GA）通过某种编码机制把对象抽象为由特定符号按一定顺序排成的串。
- 正如研究生物遗传是从染色体着手，而染色体则是由基因排成的串。
- 基本遗传算法（SGA）使用二进制串进行编码。



编码示例

- 求下列一元函数的最大值：

$$f(x) = x \cdot \sin(10\pi \cdot x) + 1.0$$

其中$x \in [-1, 2]$，求解结果精确到6位小数。

一个问题

- 对于$x \in [-1, 2]$，结果精确到6位小数，

  （1）二进制编码要求取多少位？

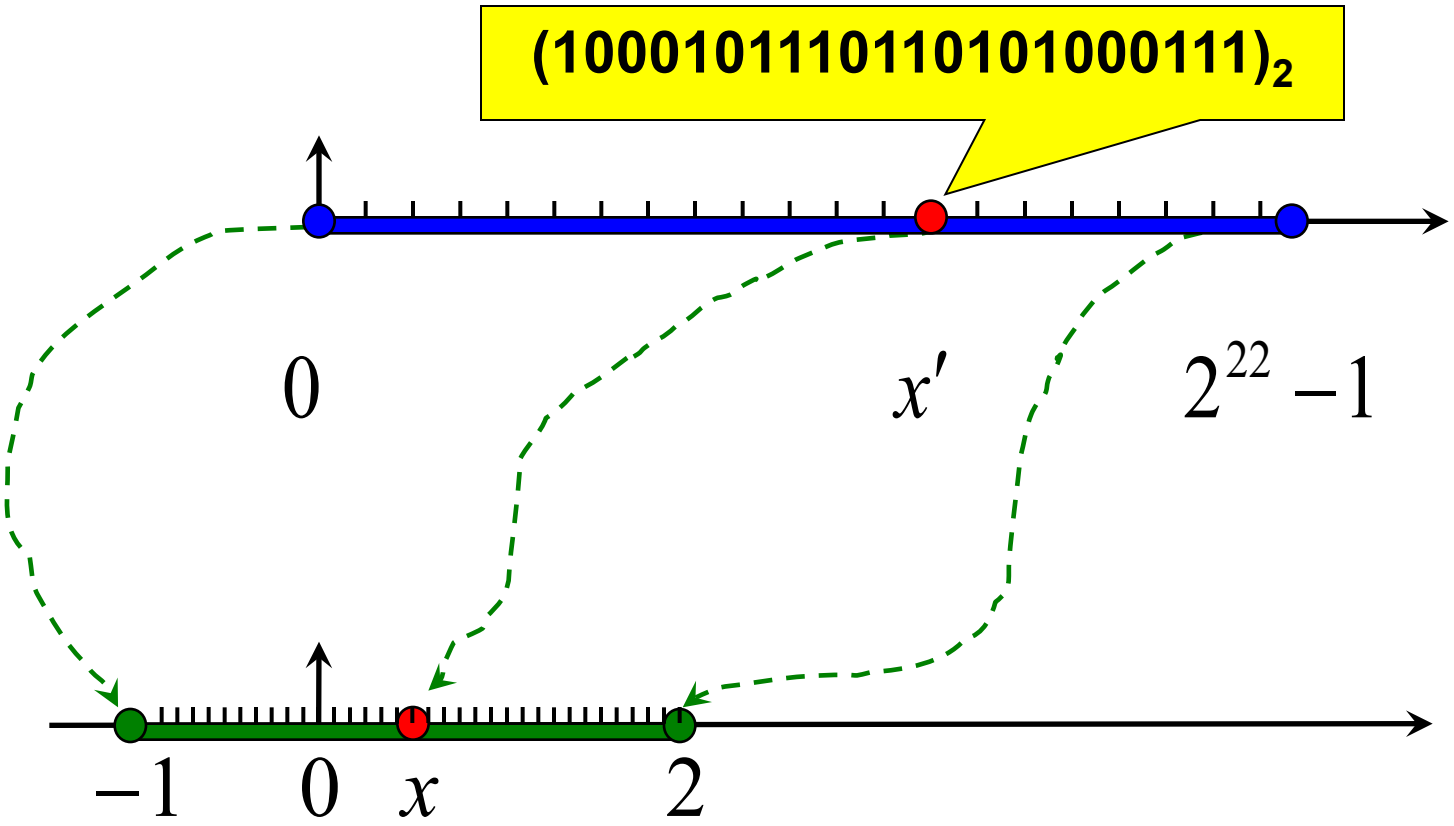
  （2）十进制实数与二进制编码之间应满足怎样的数学关系？

编码示例的分析

- 区间长度为3，求解结果精确到6位小数，因此可将自变量定义区间划分为3×10^6等份。
- 又因为$2^{21} < 3\times10^6 < 2^{22}$，所以二进制编码长度至少需要22位。
- 编码过程实质是将区间[−1，2]内对应的实数值转化为一个二进制串（$b_{21}b_{20}b_{19}b_{18}\ldots b_1b_0$）。

101010111111……101011000

编码(二进制)与实数(十进制)的转换



(1)将二进制串$(b_{21}b_{20}b_{19}\cdots b_0)$转换为十进制为

$$x' = (b_{21}b_{20}b_{19}\cdots b_0)_2 = (\sum_{i=0}^{21} b_i \cdot 2^i)_{10}$$

(2)二进制串对应的实数 x为

$$x = -1.0 + \frac{x'}{2^{22}-1} \times 3$$

例如(10001011101101010000111)表示0.637197,因为

$$x' = (\sum_{i=0}^{21} b_i \cdot 2^i)_{10} = 2288967$$

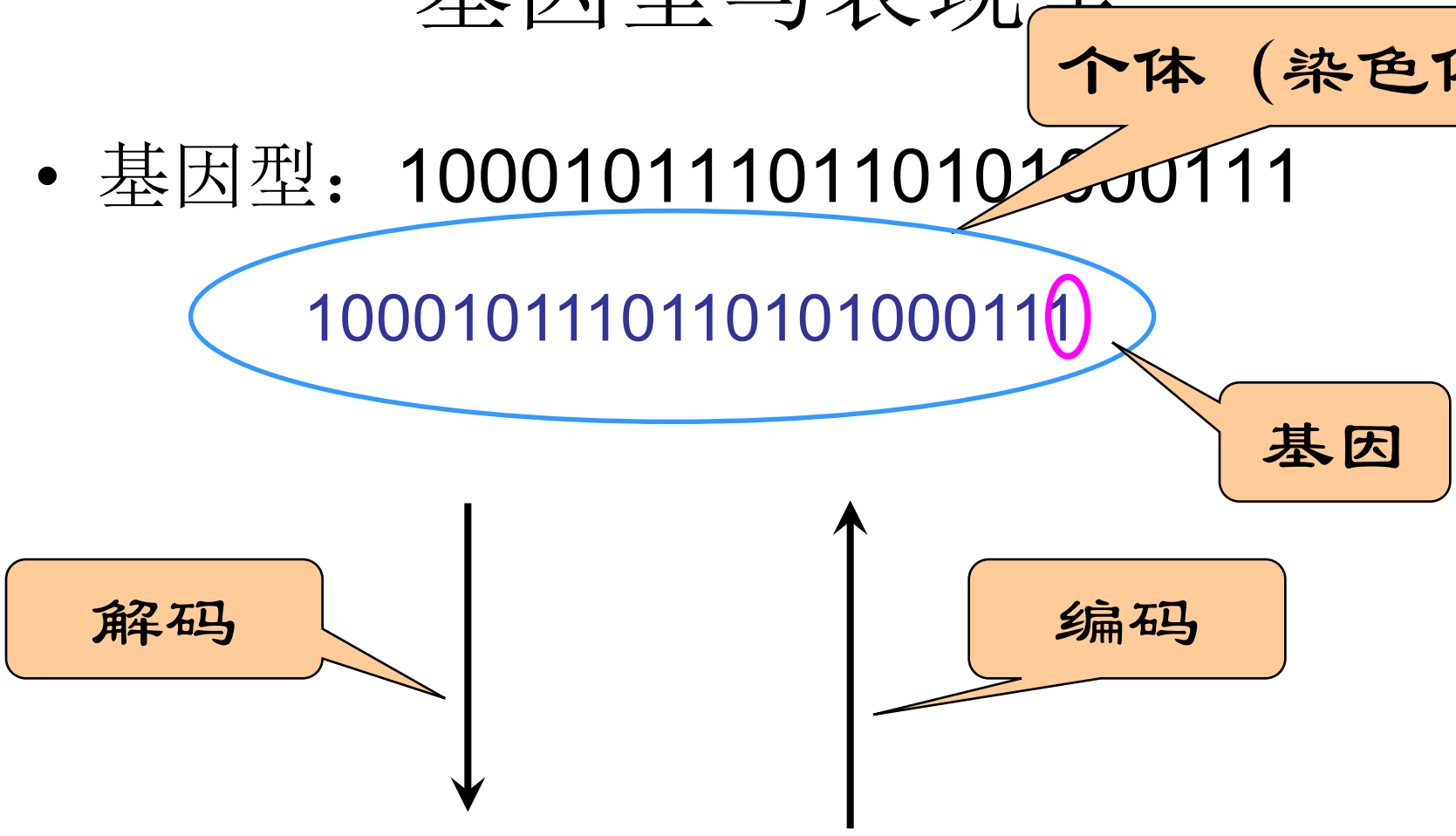
$$x = -1.0 + \frac{2288967}{2^{22}-1} \times 3 = 0.637197$$

显然:(0000000000000000000000)表示 -1

(1111111111111111111111)表示2

基因型与表现型

- 基因型：1000101110110101000111



初始种群

- 基本遗传算法（SGA）采用随机方法生成若干个个体的集合，该集合称为初始种群。
- 初始种群中个体的数量称为种群规模。

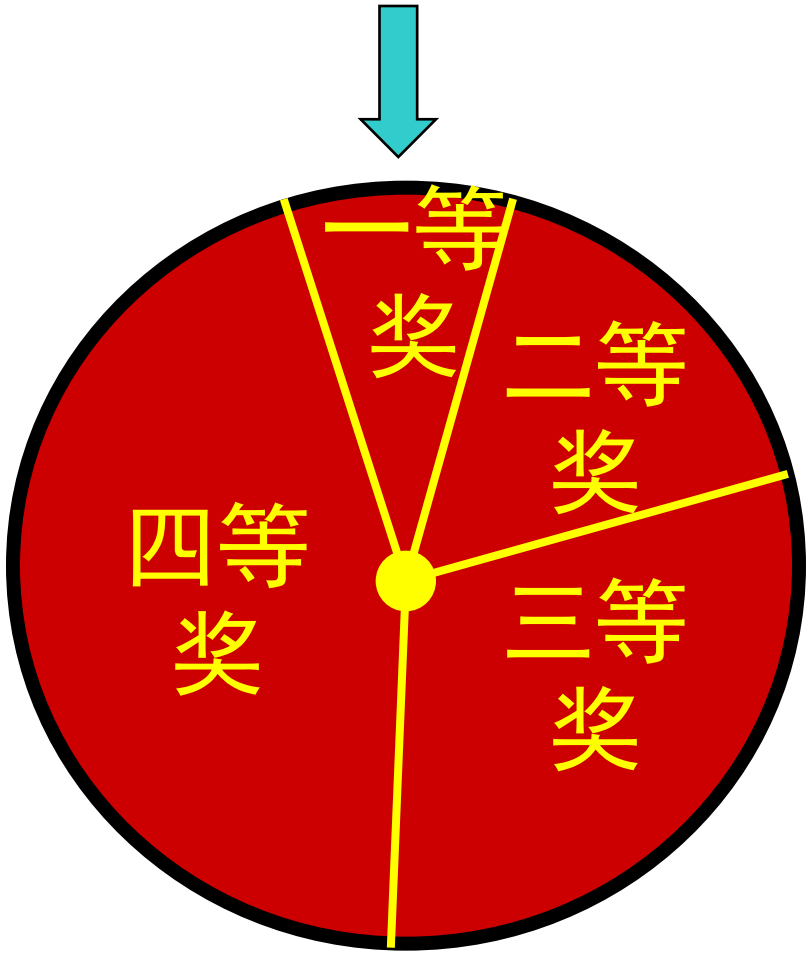
适应度函数

- 遗传算法对一个个体（解）的好坏用适应度函数值来评价，适应度函数值越大，解的质量越好。
- 适应度函数是遗传算法进化过程的驱动力，也是进行自然选择的唯一标准，它的设计应结合求解问题本身的要求而定。

选择算子

- 遗传算法使用选择运算对个体进行优胜劣汰操作。
- 适应度高的个体被遗传到下一代群体中的概率大；适应度低的个体，被遗传到下一代群体中的概率小。
- 选择操作的任务就是从父代群体中选取一些个体，遗传到下一代群体。
- 基本遗传算法（SGA）中选择算子采用轮盘赌选择方法。

轮盘赌选择方法



轮盘赌选择方法

- 轮盘赌选择又称比例选择算子，其基本思想是：各个个体被选中的概率与其适应度函数值成正比。
- 设群体大小为N，个体x_i 的适应度为 $f(x_i)$，则个体x_i的选择概率为：

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

轮盘赌选择方法

轮盘赌选择法可用如下过程模拟来实现：

(1)在［0, 1］内产生一个均匀分布的随机数*r*。

(2)若$r \le q_1$,则染色体x_1被选中。

(3)若$q_{k-1} < r \le q_k$($2 \le k \le N$), 则染色体x_k被选中。

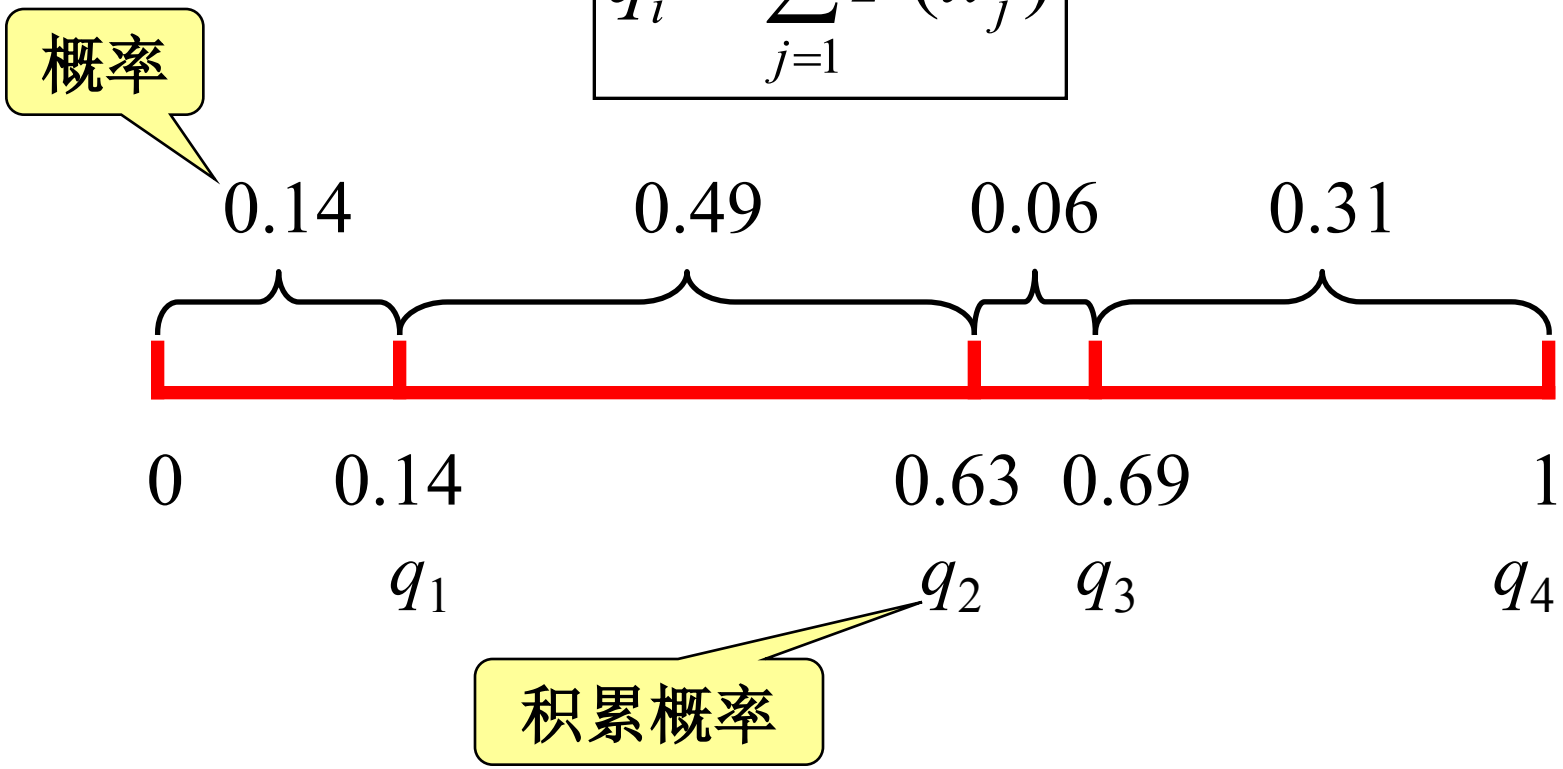
其中的q_i称为染色体x_i (i=1, 2, …, n)的积累概率, 其计算公式为

$$q_i = \sum_{j=1}^{i} P(x_j)$$

轮盘赌选择方法

积累概率实例：

$$q_i = \sum_{j=1}^{i} P(x_j)$$



轮盘赌选择方法

- 轮盘赌选择方法的实现步骤:

（1）计算群体中所有个体的适应度值；

（2）计算每个个体的选择概率；

（3）计算积累概率；

（4）采用模拟赌盘操作（即生成0到1之间的随机数与每个个体遗传到下一代群体的概率进行匹配）来确定各个个体是否遗传到下一代群体中。

轮盘赌选择方法

- 例如，有染色体

  s1= 13 (01101)

  s2= 24 (11000)

  s3= 8　(01000)

  s4= 19 (10011)

  假定适应度为$f(s)=s^2$，则

  $f(s1) = f(13) = 13^2 = 169$

  $f(s2) = f(24) = 24^2 = 576$

  $f(s3) = f(8) = 8^2 = 64$

  $f(s4) = f(19) = 19^2 = 361$

染色体的选择概率为

$$P(s_1)=\frac{f(s_1)}{\sum_{j=1}^{N}f(s_j)}=\frac{169}{169+576+64+361}=0.14$$

$$P(s_2)=\frac{f(s_2)}{\sum_{j=1}^{N}f(s_j)}=\frac{576}{169+576+64+361}=0.49$$

$$P(s_3)=\frac{f(s_3)}{\sum_{j=1}^{N}f(s_j)}=\frac{64}{169+576+64+361}=0.06$$

$$P(s_4)=\frac{f(s_4)}{\sum_{j=1}^{N}f(s_j)}=\frac{361}{169+576+64+361}=0.31$$

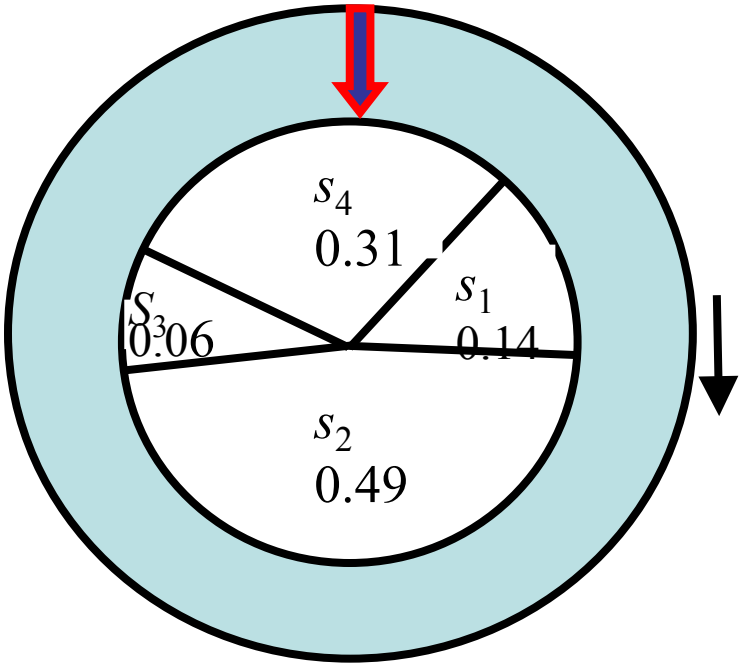
染色体的累计概率为

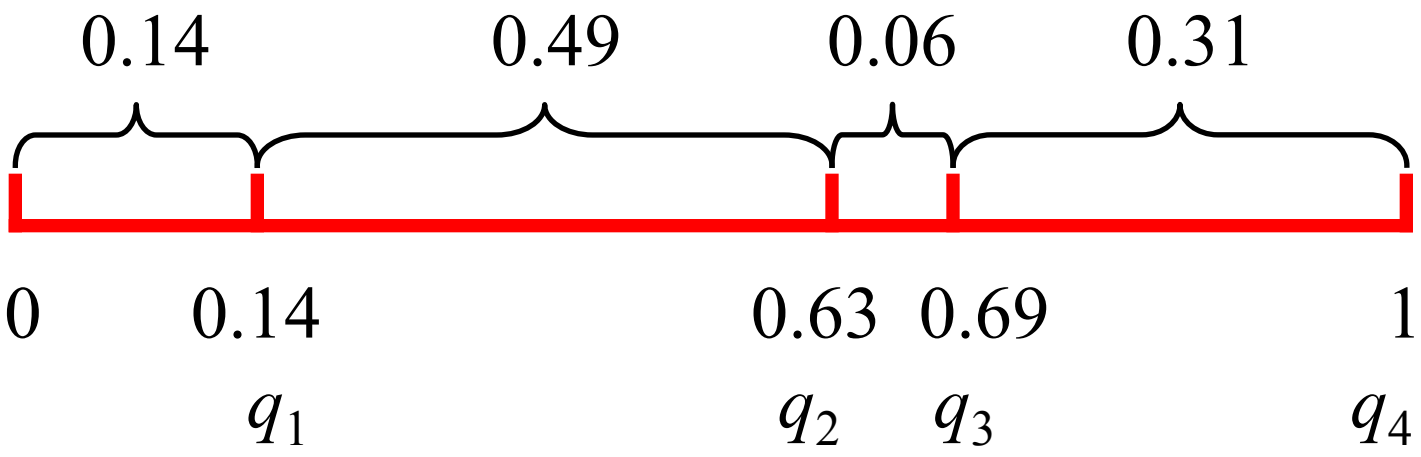
$$q(s_1)=\sum_{j=1}^{N}p(s_j)=0.14$$

$$q(s_2)=\sum_{j=1}^{N}p(s_j)=0.14+0.49=0.63$$

$$q(s_3)=\sum_{j=1}^{N}p(s_j)=0.14+0.49+0.06=0.69$$

$$q(s_4)=\sum_{j=1}^{N}p(s_j)=0.14+0.49+0.06+0.31=1$$


s_4
0.31
s_1
0.14
s_3
0.06
s_2
0.49


0.14
0.49
0.06
0.31
0
0.14
0.63
0.69
1
q_1
q_2
q_3
q_4

例如，从区间［0, 1］中产生4个随机数:

$$r_1 = 0.450126, \quad r_2 = 0.110347$$

$$r_3 = 0.572496, \quad r_4 = 0.98503$$

染色体	适应度	选择概率	积累概率	选中次数
s_1=01101	169	0.14	0.14	1
s_2=11000	576	0.49	0.63	2
s_3=01000	64	0.06	0.69	0
s_4=10011	361	0.31	1.00	1

交叉算子

- 交叉运算，是指对两个相互配对的染色体依据交叉概率 Pc 按某种方式相互交换其部分基因，从而形成两个新的个体。
- 交叉运算是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起关键作用，是产生新个体的主要方法。
- 基本遗传算法（SGA）中交叉算子采用单点交叉算子。

单点交叉运算

交叉前：

01000|0111000000010000

11100|0000011111000101

交叉后：

01000|0000011111000101 （孩子1）

11100|0111000000010000 （孩子2）

变异算子

- 变异运算，是指改变个体编码串中的某些基因值，从而形成新的个体。
- 变异运算是产生新个体的辅助方法，决定遗传算法的局部搜索能力，保持种群多样性。
- 交叉运算和变异运算的相互配合，共同完成对搜索空间的全局搜索和局部搜索。
- 基本遗传算法（SGA）中变异算子采用基本位变异算子。

基本位变异示例

变异前：

000001110000000010000

变异后：

000001110001000010000

运行参数

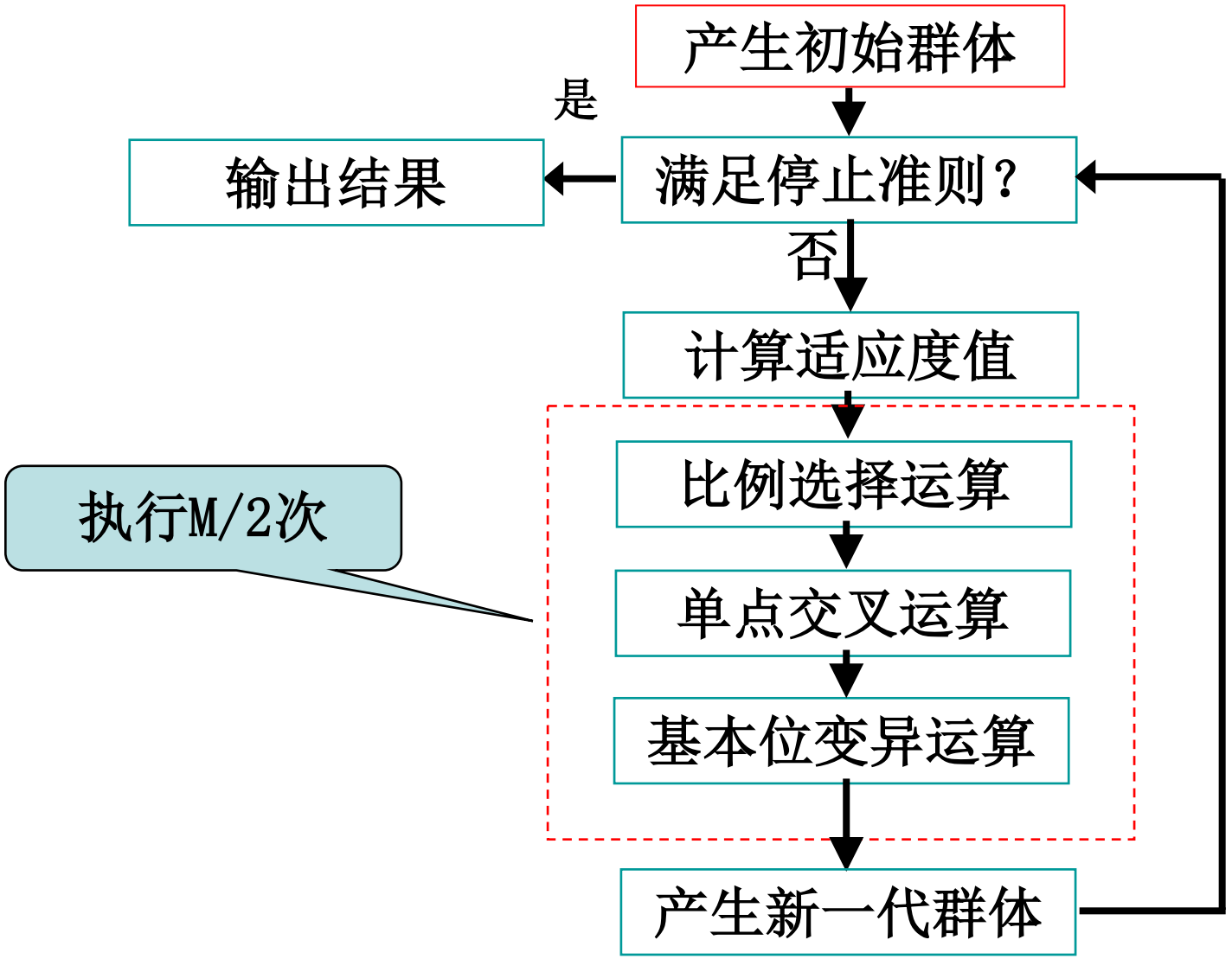
（1）M ：种群规模

（2）T ： 遗传运算的终止进化代数

（3）Pc ：交叉概率

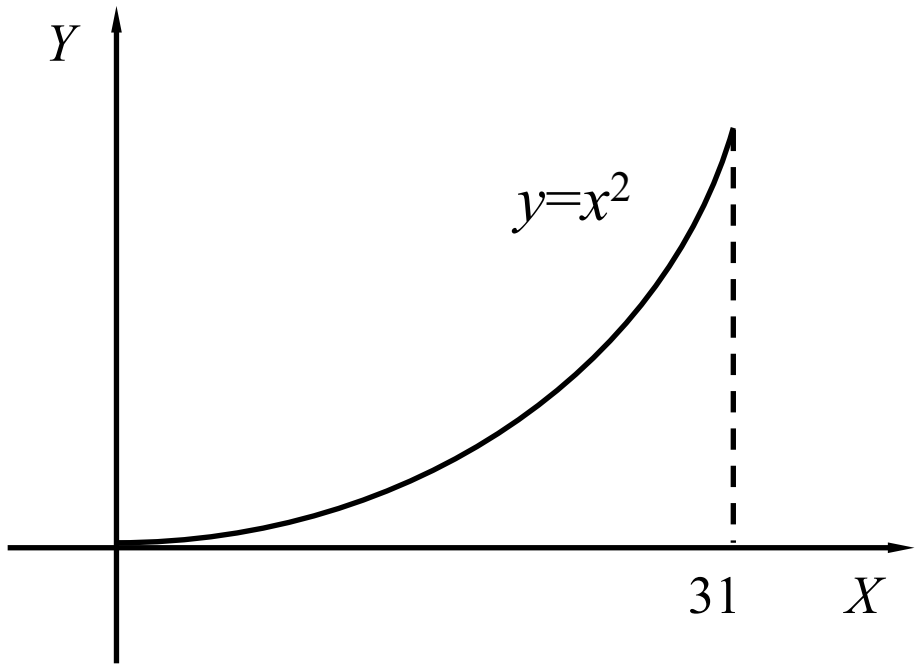
（4）Pm：变异概率

基本遗传算法的框图


产生初始群体
是
输出结果
满足停止准则？
否
计算适应度值
比例选择运算
单点交叉运算
基本位变异运算
产生新一代群体
执行M/2次

遗传算法的应用举例

已知x为整数，利用遗传算法求解区间［0，31］上的二次函数$y=x^2$的最大值。



[分析]

原问题可转化为在区间［0, 31］中搜索能使 y 取最大值的点 a 的问题。

个体：［0, 31］中的任意点x

适应度：函数值$f(x)=x^2$

解空间：区间［0, 31］

这样, 只要能给出个体x的适当染色体编码, 该问题就可以用遗传算法来解决。

[解]

(1) 设定种群规模,编码染色体，产生初始种群。

将种群规模设定为**4**；用**5**位二进制数编码染色体；取下列个体组成初始种群S_1

s_1= 13 (01101), s_2= 24 (11000)

s_3= 8 (01000), s_4= 19 (10011)

(2) 定义适应度函数, 取适应度函数

$$f(x)=x^2$$

(3) 计算各代种群中的各个体的适应度，并对其染色体进行遗传操作,直到适应度最高的个体，即31（11111）出现为止。

首先计算种群S1中各个体

s1= 13(01101),　s2= 24(11000)

s3= 8(01000),　s4= 19(10011)

的适应度f (si), 容易求得

$f(s1) = f(13) = 13^2 = 169$

$f(s2) = f(24) = 24^2 = 576$

$f(s3) = f(8) = 8^2 = 64$

$f(s4) = f(19) = 19^2 = 361$

再计算种群S_1中各个体的选择概率。

$$P(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

由此可求得 $P(s_1)=P(13)=0.14$

$P(s_2)=P(24)=0.49$

$P(s_3)=P(8)=0.06$

$P(s_4)=P(19)=0.31$

再计算种群S_1中各个体的积累概率

$$q_i = \sum_{j=1}^{i} P(x_j)$$

$$q(s_1) = \sum_{j=1}^{N} p(s_j) = 0.14$$

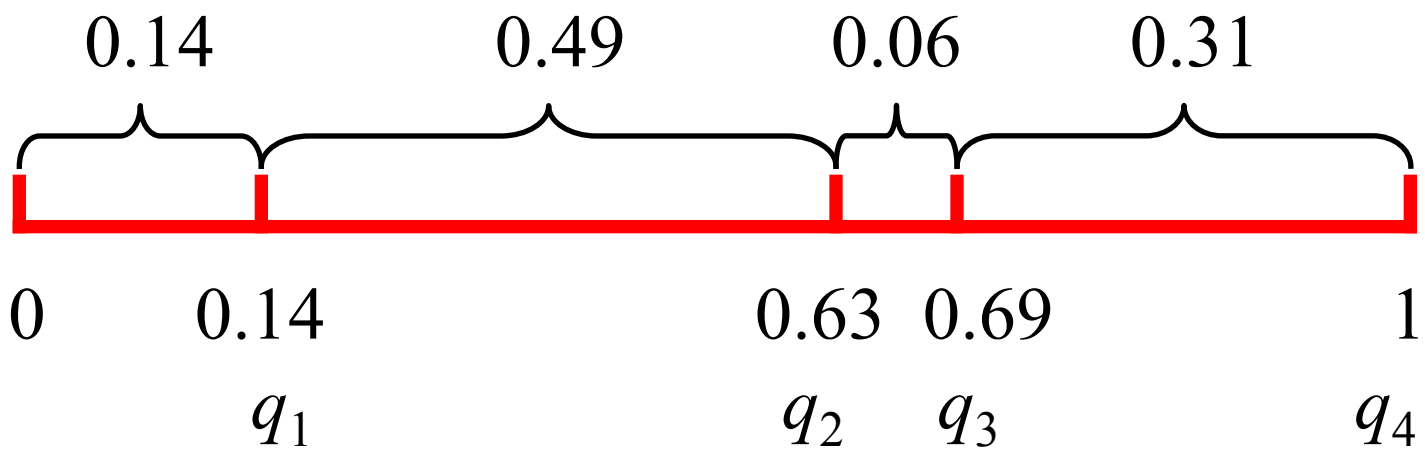
$$q(s_2) = \sum_{j=1}^{N} p(s_j) = 0.14 + 0.49 = 0.63$$

$$q(s_3) = \sum_{j=1}^{N} p(s_j) = 0.14 + 0.49 + 0.06 = 0.69$$

$$q(s_4) = \sum_{j=1}^{N} p(s_j) = 0.14 + 0.49 + 0.06 + 0.31 = 1$$

种群S_1中各个体的积累概率

$$q_i = \sum_{j=1}^{i} P(x_j)$$



选择-复制：设从区间［0, 1］中产生4个随机数:

$r_1 = 0.450126, \quad r_2 = 0.110347$

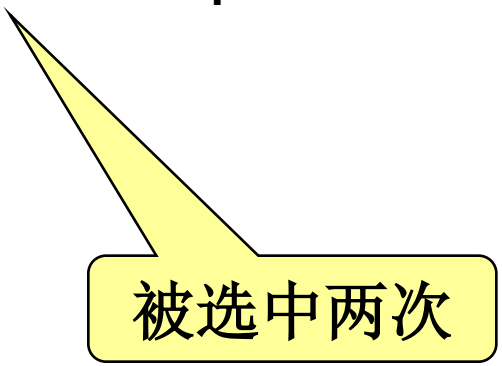
$r_3 = 0.572496, \quad r_4 = 0.98503$

染色体	适应度	选择概率	积累概率	选中次数
s_1=01101	169	0.14	0.14	1
s_2=11000	576	0.49	0.63	2
s_3=01000	64	0.06	0.69	0
s_4=10011	361	0.31	1.00	1

于是，经复制得群体：

s_1'=11000（24），s_2'=01101（13）

s_3'=11000（24），s_4'=10011（19）



交叉

设交叉率p_c=100%，即S_1中的全体染色体都参加交叉运算。

设s_1'与s_2'配对，s_3'与s_4'配对。

s_1' =11000（24），s_2' =01101（13）

s_3' =11000（24），s_4' =10011（19）

分别交换后两位基因，得新染色体：

s_1''=11001（25），s_2''=01100（12）

s_3''=11011（27），s_4''=10000（16）

变异

设变异率p_m=0.001。

这样，群体S_1中共有

$$5\times4\times0.001=0.02$$

位基因可以变异。

0.02位显然不足1位，所以本轮遗传操作不做变异。

于是，得到第二代种群S_2：

s_1=11001（25），s_2=01100（12）

s_3=11011（27），s_4=10000（16）

第二代种群S_2中各染色体的情况

染色体	适应度	选择概率	积累概率	估计选中次数
s_1=11001	625	0.36	0.36	1
s_2=01100	144	0.08	0.44	0
s_3=11011	729	0.41	0.85	2
s_4=10000	256	0.15	1.00	1

假设这一轮选择-复制操作中，种群S_2中的4个染色体都被选中，则得到群体：

s_1'=11001（25），s_2'= 01100（12）

s_3'=11011（27），s_4'= 10000（16）

做交叉运算，让s_1'与s_2'，s_3'与s_4'分别交换后三位基因，得

s_1''=11100（28），s_2'' = 01001（9）

s_3''=11000（24），s_4'' = 10011（19）

这一轮仍然不会发生变异。

于是，得第三代种群S_3：

s_1=11100（28），s_2=01001（9）

s_3=11000（24），s_4=10011（19）

第三代种群S_3中各染色体的情况

染色体	适应度	选择概率	积累概率	估计的选中次数
s_1=11100	784	0.44	0.44	2
s_2=01001	81	0.04	0.48	0
s_3=11000	576	0.32	0.80	1
s_4=10011	361	0.20	1.00	1

设这一轮的选择-复制结果为：

被选中两次

s_1'=11100（28），s_2'=11100（28）

s_3'=11000（24），s_4'=10011（19）

做交叉运算，让s_1'与s_4'，s_2'与s_3'分别交换后两位基因，得

s_1''=11111（31），s_2''=11100（28）

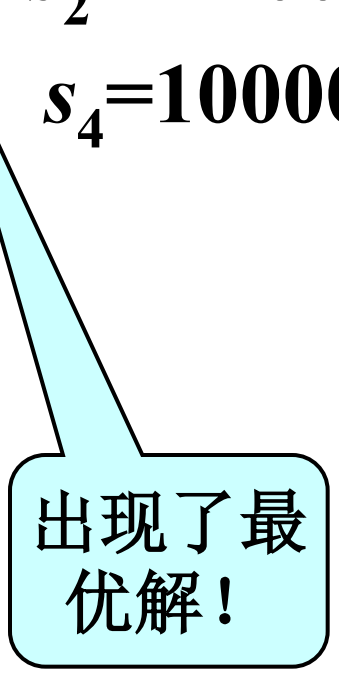
s_3''=11000（24），s_4''=10000（16）

这一轮仍然不会发生变异。

于是，得第四代种群S_4:

s_1=11111（31），s_2=11100（28）

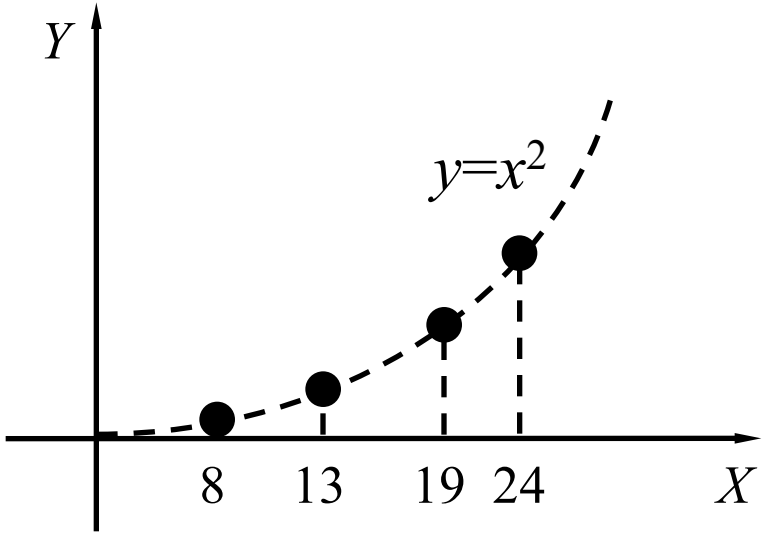
s_3=11000（24），s_4=10000（16）



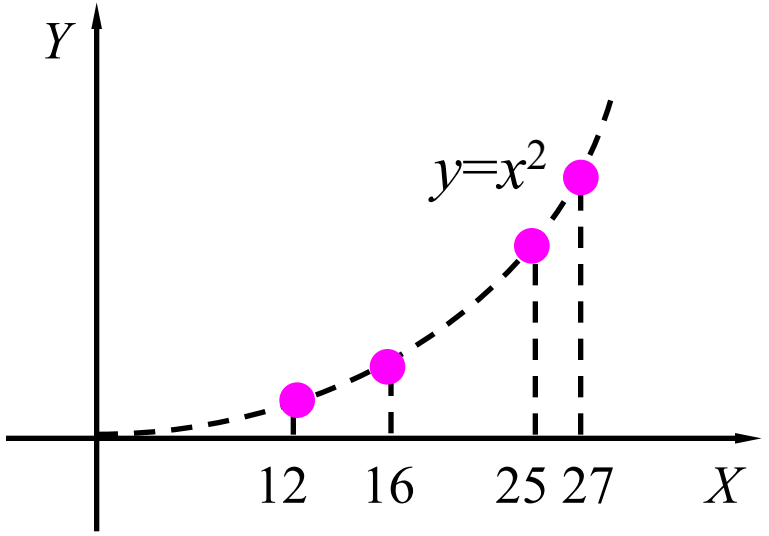
显然，在这一代种群中已经出现了适应度最高的染色体s_1=11111。于是，遗传操作终止，将染色体（11111）作为最终结果输出。

然后，将染色体“11111”解码为表现型，即得所求的最优解：31。

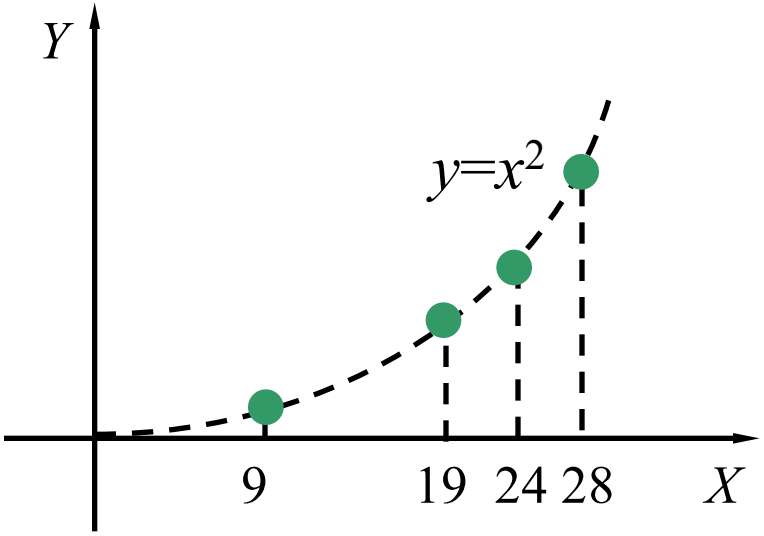
将31代入函数$y=x^2$中，即得原问题的解，即函数$y=x^2$的最大值为961。



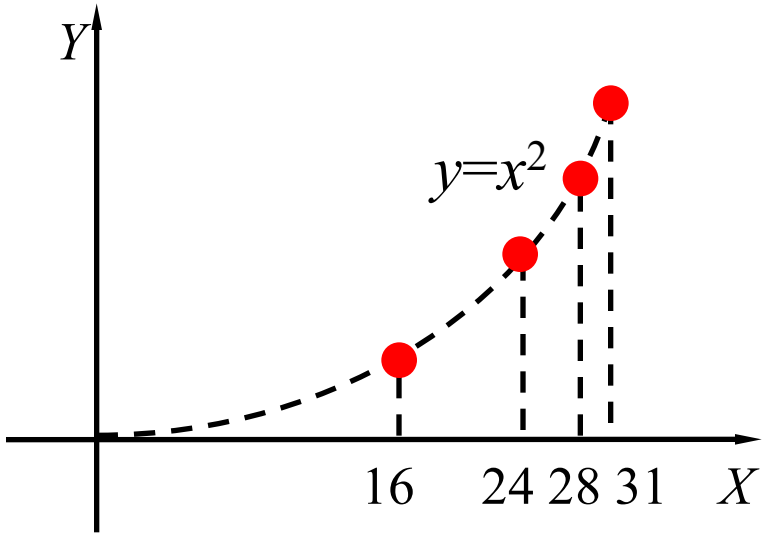
第一代种群及其适应度



第二代种群及其适应度



第三代种群及其适应度



第四代种群及其适应度

刚刚产生四代种群及其适应度。你知道这四代种群之间有何发展规律吗？

四代种群的发展规律就是适应度的整体水平越来越好，直到最优解出现！

总结

- 粒子群优化算法和遗传算法不需要导数信息,计算简单,全局搜索能力强,特别适合寻找全局最优解. 工程中应用广泛.
- 缺点是: 计算量很大,精度不高.