# 遗传算法-入门级最好教程

遗传算法（Genetic Algorithm，GA）是受生物进化论的启发，由J.Holland于1975年提出的[1]，GA是基于“适者生存”的一种高度并行、随机和自适应优化算法，它将问题的求解表示成“染色体”的适者生存过程，通过“染色体”群（population）的一代代不断进化，包括复制（reproduction）、交叉（crossover）和变异（mutation）等操作，最终收敛到“最适应环境”的个体，从而求得问题的最优解或满意可行解。

GA是一种通用的优化算法，其编码技术和遗传操作相对比较简单，优化不受限制性条件的约束，也并不关心待优化问题的梯度问题，其两个最大的显著优点是隐含并行性和全局解空间搜索。随着计算机技术的发展，GA的应用领域也越来越广泛，在机器学习、模式识别、图像处理、神经网络、组合优化、优化控制等领域取得了喜人的成果，当然在生产调度领域也有非常多的成果。

本部分内容将带领大家认识GA的发展历程和优化的一般过程，旨在打造遗传算法的入门级最好教程，后面会陆续再详细介绍涉及的遗传操作，注意是画说哦，敬请期待！

## GA的发展历程

遗传算法的发展历程，大致分为三个时期：萌芽、成长和发展。

### 萌芽期（50年代后期~70年代中期）

早在上世界50年代后期和60年代初期，一些生物学家就已经开始采用电子计算机模拟生物的遗传进化系统，尽管这些工作纯粹服务于研究生物现象，但是他们已经开始使用现代遗传算法的一些标识方式，如1960年，美国A.S.Fraser为了建立生物的表现型方程，用3组5位（共15位）长的0-1字符串表示方程的三个参数[2]。

1962年，美国J.H.Holland教授在研究适应系统时，提出了系统本身与外部环境的相互作用与协调，就已经涉及到了进化算法的思想。之后他又进行了许多适应系统方面的研究工作，在1968年他又提出了模式理论，奠定了遗传算法的主要理论基础。

1967年美国的J.D.Bagay在其关于博弈论的论文中，第一次使用了遗传算法（Genetic algorithm）这个术语，他采用了复制、交换、突变等手段研究轨迹象棋的对弈策略。

1975年，J.H.Holland教授出版了著作《自然界和人工系统适应性（Adaptation in Natural and Artificial Systems）》[1]，比较全面地介绍了遗传算法，人们常把这一事件作为遗传算法得到承认的标志，1975年也就成为了遗传算法的诞生年，Holland也就成为了遗传算法的创始人。

### 成长期（70年代中期~80年代末）

从70年代末至80年代初，许多研究工作者开始从事遗传算法的研究，遗传算法一度成为美国人工智能研究的一个热点。

1987年，美国的D.Lawrence总结了人们长期从事遗传算法的经验，公开出版了《遗传算法与模拟退火（Genetic Algorithm and Simulated Annealing）》[3]一书，以论文集形式用大量的实例介绍了遗传算法的使用技术。

1989年J.H.Holland教授的学生，D.E.Goldberg博士出版专著《遗传算法—搜索、优化及机器学习（Genetic Algorithms—in Search，Optimization and Machine Learning）》[4]，非常全面、系统地介绍了遗传算法的基本原理和应用，使得这一优化技术得到普及与推广，该书从而也被视为遗传算法的基础教科书。

至此，遗传算法已广泛应用于生物、工程、运筹学、计算机科学、图像处理、模式识别和社会科学等领域。1985年在美国举办了第一届遗传算法国际学术会议（International Conference on Genetic Algorithms），此后又举办了很多届。

### 发展期（90年代）

90年代以后，遗传算法不断地向广度和深度发展，1991年，D.Lawrence公开发行《遗传算法手册（Handbook of Genetic Algorithms）》[5]一书，同时随着应用的广泛，遗传算法也暴露出了在表达方面的局限性，在1989年，美国斯坦福大学的J.R.Koza提出了遗传规划新概念，用层次化的计算机程序代替字符串表达问题，这里不对该概念展开，后面会单独介绍该算法。

## 遗传算法基础

工业工程中，特别是制造系统中的许多最优化问题性质非常复杂，很难用传统的优化方法（整数规划等）来求解，遗传算法在这类问题上显示除了比较优越的性能，为了更清楚地介绍遗传算法的优化过程，首先给出了生物学术语在遗传算法中的含义对应，方便更好地理解遗传算法，然后描述了基本遗传算法的一般流程，最后通过简单的案例亲眼见证遗传算法的魅力。

### 遗传算法的生物学含义

遗传算法受生物进化与遗传的启发，因此其术语也常常参照生物学的术语。

遗传算法的运算基础是字符串（注意：这里的字符串既可以由字符，也可以由整数、浮点数等构成），相当于生物学中的染色体。字符串中的每个字符都有特定的含义，反映所求解问题的某个特征，相当于基因，亦即染色体DNA的片段。在进行交叉、变异操作时，遗传算法只涉及字符串的某些片段，这就类似于遗传过程只涉及部分基因而不是整个染色体。

遗传学很重视等位基因，它是反映生物某一形态所对应的基因。在遗传算法的字符串中，每个字符都反映问题的某一特性，这也就相当于等位基因，至于等位基因的位置，也就相当于该字符在字符串中的位置。

在遗传学中，杂交产生的子代里显现出亲本的性状，称作显性性状，为显现出来的亲本性状叫做隐性性状。控制显性性状的基因是显性基因，用大写英文字母表示，控制隐性性状的基因是隐性基因，用小写英文字母表示。在遗传算法中，模仿这种大、小写字母表达方式，对显性基因和隐性基因采取不同的操作。

在生物学中，生物个体所表现出来的性状被称作表现型，把反映表现型的基因组成称作基因型，在遗传算法中，表现型就相当于所求解问题的特征，如参数、可行解等，基因型就是字符串的结构。

此外，在生物学中，把细胞中含有两个染色体组的个体称为二倍体，把含有三个及以上染色体组的个体称为多倍体，遗传算法也有相应的处理。

表1 生物学术语在遗传算法中的含义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 生物学 | 遗传算法 | 说明 |
| 1 | 染色体（Chromosome） | 字符串 | 遗传算法的运算基础是字符串（注意：这里的字符串既可以由字符，也可以由整数、浮点数等构成），相当于生物学中的染色体。 |
| 2 | 基因（Gene） | 字符 | 字符串中的每个字符都有特定的含义，反映所求解问题的某个特征，相当于基因，亦即染色体DNA的片段。 |
| 3 | 等位基因（Allele） | 对应的字符 | 在遗传算法的字符串中，每个字符都反映问题的某一特性，这也就相当于等位基因 |
| 4 | 基因位置（Locus） | 字符的位置 | 等位基因的位置，也就相当于该字符在字符串中的位置。 |
| 5 | 表现型（Phenotype） | 字符串含义 | 在生物学中，生物个体所表现出来的性状被称作表现型，在遗传算法中，表现型就相当于所求解问题的特征，如参数、可行解等。 |
| 6 | 基因型（Genotype） | 字符串结构 | 在生物学中，把反映表现型的基因组成称作基因型，在遗传算法中，基因型就是字符串的结构。 |
| 7 | 上位效应（Epistasis） | 非线性作用 |  |
| 8 | 进化（Evolution） | 迭代 | 种群逐渐适应生存环境，品质不断得到改良。生物的进化是以种群的形式进行的。 |
| 9 | 适应度(Fitness) | 适应度函数 | 度量某个物种对于生存环境的适应程度，在遗传算法中就是对染色体解码后得到适应度函数值。 |
| 10 | 选择(Selection) | | 以一定的概率从种群中选择若干个个体。一般，选择过程是一种基于适应度的优胜劣汰的过程。 |
| 11 | 复制(Reproduction) | | 细胞分裂时，遗传物质DNA通过复制而转移到新产生的细胞中，新细胞就继承了旧细胞的基因。 |
| 12 | 交叉(Crossover) | | 两个染色体的某一相同位置处DNA被切断，前后两串分别交叉组合形成两个新的染色体。也称基因重组或杂交，在遗传算法中其实有多种不同的交叉算子。 |
| 13 | 变异(Mutation) | | 复制时可能（很小的概率）产生某些复制差错，变异产生新的染色体，表现出新的性状，在遗传算中也有多种不同的变异算子。 |
| 14 | 编码(Coding) | | DNA中遗传信息在一个长链上按一定的模式排列。遗传编码可看作从表现型到基因型的映射。在遗传算法中，就是将所求解问题表达成染色体的过程。 |
| 15 | 解码(Decoding) | | 基因型到表现型的映射，在遗传算法中就是通过染色体计算得到适应度值的过程。 |
| 16 | 个体（Individual） | | 指带有染色体特征的实体，在遗传算中，每个个体代表所求问题的一个解。 |
| 17 | 种群（Population） | | 个体的集合，该集合内个体数称为种群的大小。 |

### 遗传算法一般流程

遗传算法的常用形式是Goldberg提出的，遗传算法是一种基于生物自然选择与遗传机理的随机搜索算法，和传统的搜索算法不同，遗传算法从一组随机产生的初始解（称为种群）开始搜索过程。种群中的每个个体是问题的一个解，称为染色体，这些染色体在后续迭代中不算进化，称为遗传，在每一代中用适应度来评价每个染色体的好坏，生成的下一代染色体称为子代（Offspring）。子代是由父代染色体（Parent）通过交叉或变异操作形成的。新一代形成中，根据适应度值的大小选择部分子代，淘汰部分子代，从而保持种群大小不变。适应度值高的个体被选中的概率更高。这样经过若干代之后，算法收敛于最优的染色体，它可能就是问题的最优解或次有解。

设Gen达标遗传的代数，M表示种群大小，i表示已处理个体的累计数，当i=M时表示这一代的所有个体已全部处理完成，需要转入下一代群体。pr、pc、pm分别表示复制、交叉和变异的概率。遗传算法的一般流程如图1所示。



图1 遗传算法一般框架

遗传算法的基本运算过程如下：

a)初始化：设置进化代数计数器Gen=0，设置最大进化代数T，随机生成M个个体作为初始群体P(0)。

b)个体评价：计算群体P(Gen)中各个个体的适应度。

c)选择运算：将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

d)交叉运算：将交叉算子作用于群体。遗传算法中起核心作用的就是交叉算子。

e)变异运算：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。群体P(Gen)经过选择、交叉、变异运算之后得到下一代群体P(Gen+1)。

f)终止条件判断：若Gen=T,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

## 遗传算法举例

本节将用一个简单的例子，来详细说明遗传算法是如何工作的。

对于以下无约束优化问题：

 (1)

其三维图形如图2所示。

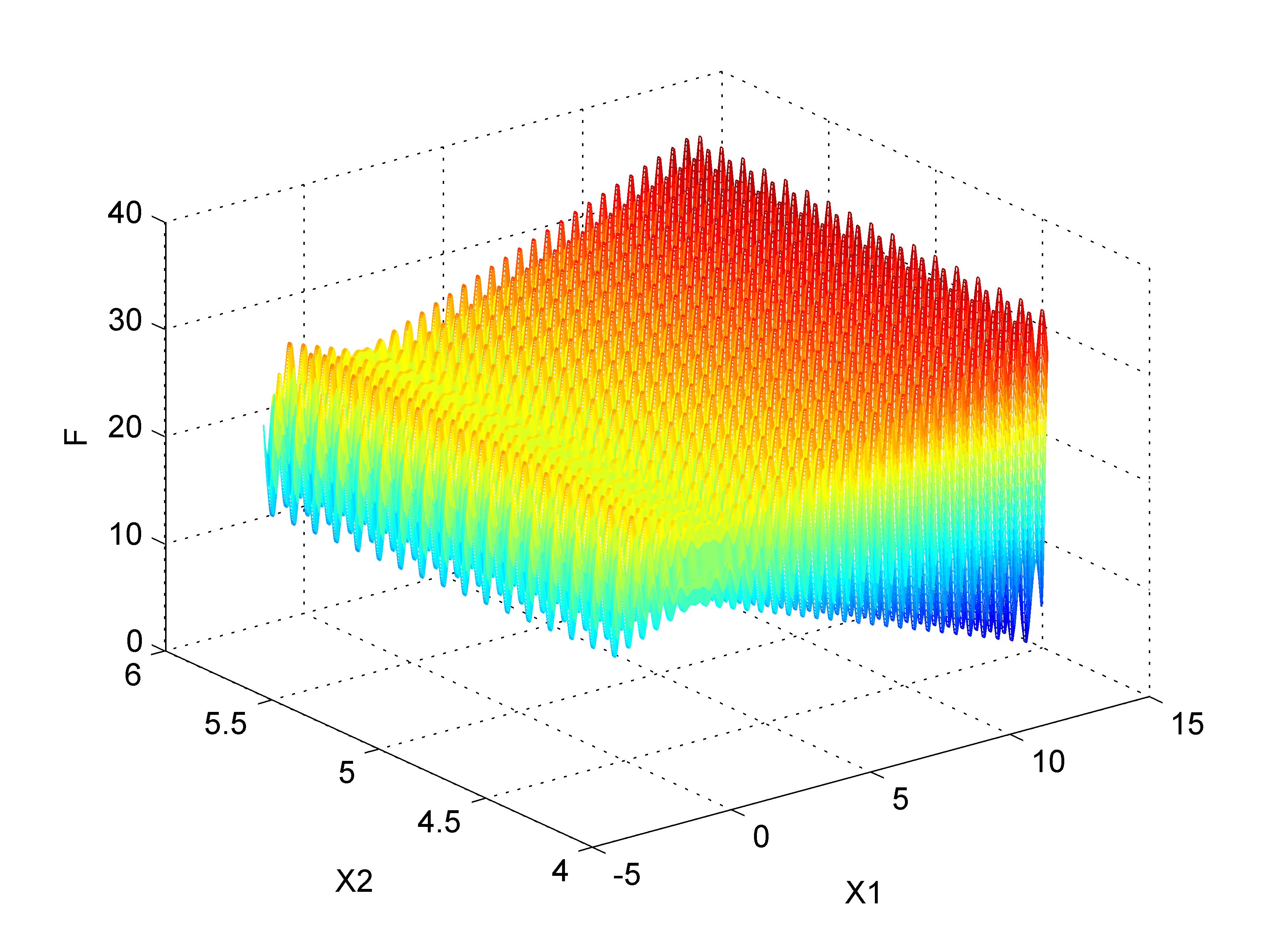


图2 目标函数曲面图

### 基因表达

首先，将决策变量编码为二进制串，串长取决于需要的精度，例如，xj的值域是[aj,bj]，而需要的精度是小数点后5位，这就要求xj的值域至少要分为(bj-aj)x106份。设xj所需要的子串长为mj，则有



将xj由二进制转换为十进制可按下式得到：



decimal(substringj)表示变量xj的子串substringj的十进制值。

假设x1和x2需要的精度都是小数点后5位，两个变量需要的总串长按下面计算：

(12.1-(-3.0))x10000=151000

217<151000≤218

所以m1=18。

(5.8-4.1)x10000=17000

214<17000≤215

m2=15。

染色体的总长为18+15=33位，可表示如下：



对应的变量x1和x2的值为



对应的十进制值为：

x1=-3.0+5417x(12.1-(-3.0))/(218-1)=-2.687969

x2=4.1+24318x(5.8-4.1)/(215-1)=5.361653

### 种群初始化

随机产生的初始种群如下：

v1=[000001010100101001101111011111110]

v2=[001110101110011000000010101001000]

v3=[111000111000001000010101001000110]

v4=[100110110100101101000000010111001]

v5=[0000101111011000100011100011010000]

v6=[111110101011011000000010110011001]

v7=[110100010011111000100110011101101]

v8=[001011010100001100010110011001100]

v9=[111110001011101100011101000111101]

v10=[111101001110101010000010101101010]

对应的十进制值为

v1=[x1,x2]=[-2.687969,5.361653]

v2=[x1,x2]=[0.474101,4.170144]

v3=[x1,x2]=[10.419457,4.661461]

v4=[x1,x2]=[6.159951,4.109598]

v5=[x1,x2]=[-2.301286,4.455282]

v6=[x1,x2]=[11.788.84,4.174346]

v7=[x1,x2]=[9.342067,5.121702]

v8=[x1,x2]=[-0.330256,4.694977]

v9=[x1,x2]=[11.671267,4.873501]

v10=[x1,x2]=[11.446273,4.171908]

### 评估

计算染色体适应度值的过程由以下三步构成：

Step1：将染色体的基因型转换为表现型，即将二进制串转换为十进制值：xk=(x1k,x2k),k=1,2,...,M。

Step2：计算目标函数值f(xk)。

Step3：将目标哈数值转换为适应度值，对于最大化问题，可简单地取目标值为适应度值。

以上初始种群中染色体的适应度值为

eval(v1)=f(-2.687969,5.361653)=19.805119

eval(v2)=f(0.474101,4.170144)=17.370896

eval(v3)=f(10.419457,4.661461)=9.590546

eval(v4)=f(6.159951,4.109598)=29.406122

eval(v5)=f(-2.301286,4.455282)=15.686091

eval(v6)=f([11.788.84,4.174346)=11.90541

eval(v7)=f(9.342067,5.121702)=17.958717

eval(v8)=f(-0.330256,4.694977)=19.763190

eval(v9)=f(11.671267,4.873501)=26.401669

eval(v10)=f(11.446273,4.171908)=10.252480

显然，染色体v4是最好的，而染色体v3是最差的。

### 选择

大多数情况下采用轮盘赌法作为选择方法，它是一种正比选择策略，能够根据与适应度值成正比的概率选择新的种群。轮盘赌法由以下四步构成：

1． 对各个染色体vk计算适应度值eval(vk)



2. 计算种群中所有染色体的适应度之和



3. 对各染色体**v**k，计算选择概率pk



4. 对各染色体**v**k，计算累积概率qk（想必大家可能有点疑惑，这个地方都已经有了选择概率了但是为什么还要计算累积概率呢，这是因为随机产生的随机数只能将[0,1]区间分割成两部分，所以当种群中有多个染色体时，必须通过随机数位于哪个区间确定选择对应的染色体，此时这些选择概率在数轴上就需要进行累加求和。）



选择过程就是旋转轮盘M次，每次按如下方式选出一个染色体来构造新的种群。

Step1：在[0,1]区间内产生一个均匀分布的伪随机数r。

Step2：若r≤q1，则选第一个染色体v1，否则，选择第k个染色体vk（2≤k≤M），使得qk-1<r≤qk成立。

案例中，种群的适应度值和F为178.135372。

各染色体vk（k=1,2,...,10）的选择概率pk为

p1=0.111180，p2=0.097515，p3=0.053839，p4=0.165077，p5=0.088057，p6=0.066806，p7=0.100815，p8=0.110945，p9=0.148211，p10=0.057554

由此得到如下的轮盘赌：



各染色体的累积概率qk为

q1=0.111180，q2=0.208695，q3=0.262534，q4=0.427611，q5=0.515668，q6=0.582475，q7=0.683290，q8=0.794234，q9=0.942446，q10=1.000000

下面旋转轮盘10次，每次选一个染色体来构造新种群，设产生的[0,1]区间内的10个随机数分别为：

0.301431,0.322062,0.766503,0.881893,0.350871,

0.583392,0.177618,0.3434242,0.032685,0.197577

第一个数r1=0.301431大于q3小于q4，这说明染色体v4被选择出来构造新种群，重复以上操作，选出如下的新种群：

v1’=[100110110100101101000000010111001](v4)

v2’=[100110110100101101000000010111001](v4)

v3’=[001011010100001100010110011001100](v8)

v4’=[111110001011101100011101000111101](v9)

v5’=[100110110100101101000000010111001](v4)

v6’=[110100010011111000100110011101101](v7)

v7’=[001110101110011000000010101001000](v2)

v8’=[100110110100101101000000010111001](v4)

v9’=[000001010100101001101111011111110](v1)

v10’=[001110101110011000000010101001000](v2)

### 交叉

这里先采用单点交叉，更详细的交叉操作后面会单独介绍。该方法在染色体中随机选择一个断点，交换两个父代染色体上断点的右端，对于如下两个染色体，若随机断点选在第17个基因之后：



交换父代染色体断点右端的部分将得到：



设交叉率pc=0.25，即平均有25%的染色体进行交叉，于是有：

交叉过程

begin

k←0；

while k≤10 do

rk←[0,1]均匀分布随机数；

if rk<0.25 then

选择vk作为交叉的父代之一；

end

k←k+1；

end

end

设随机数列如下：

0.625721,0266823,0.288644,0.295114,0.163274

0.567461,0.085940,0.392865,0.770714.0.548656

于是染色体v5’和v7’被选择进行交叉，设断点在第一个基因之后，则交叉得到的子代染色体为：



### 变异

变异以变异率的概率改变一个或几个基因，假设v1的第16个基因被选择进行变异，该基因为1，则变异后卫0，于是变异后的染色体为：



设变异率为pm=0.01，即种群中平均有1%的基因会发生变异，种群中共有33x10=330个基因，假设如下基因发生变异：

|  |  |  |
| --- | --- | --- |
| 染色体 | 基因位 | 随机数 |
| 4 | 6 | 0.009857 |
| 5 | 32 | 0.003113 |
| 7 | 1 | 0.000946 |
| 10 | 32 | 0.001282 |

变异后的新种群如下：

v1’=[100110110100101101000000010111001]

v2’=[100110110100101101000000010111001]

v3’=[001011010100001100010110011001100]

v4’=[111111001011101100011101000111101]

v5’=[100110110100101101000000010111011]

v6’=[110100010011111000100110011101101]

v7’=[101110101110011000000010101001000]

v8’=[100110110100101101000000010111001]

v9’=[000001010100101001101111011111110]

v10’=[001110101110011000000010101001010]

相应的十进制值和适应度值为：

f(6.159951,4.109598)=29.406122

f(6.159951,4.109598)=29.406122

f(-0.330256,4.694977)=19.763190

f(11.907206,4.873501)=5.702781

f(8.024130,4.170248)=19.91025

f(9.342067,5.121702)=17.958717

f(6.159951,4.109598)=29.406122

f(6.159951,4.109598)=29.406122

f(-2.687969,5.361653)=19.805119

f(0.474101,4.170248)=17.370896

至此完成了完整的一次迭代，最终循环1000代，得到以下最优染色体：

v\*=[111110000000111000111101001010110]

eval(v\*)=f(11.631407,5.724824)=38.818208

x1\*=11.631407,x2\*=5.724824,f(x1\*,x2\*)=38.818208

1. Holland, J., *Adaptation In Natural And Artificial Systems.* University of Michigan Press, 1975.

2. Fraser, A.S., *Simulation of Genetic Systems by Automatic Digital Computers VII. Effects of Reproductive Ra'l'e, and Intensity of Selection, on Genetic Structure.* Australian Journal of Biological Sciences, 1960. **13**(3).

3. Castillo, O. and P. Melin, *Genetic Algorithms and Simulated Annealing*, in *Soft Computing for Control of Non-Linear Dynamical Systems*, O. Castillo and P. Melin, Editors. 2001, Physica-Verlag HD: Heidelberg. p. 63-84.

4. Goldberg, D., *Genetic Algorithm in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989. **xiii**.

5. Davis, L., *Handbook Of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991. **115**.