西安电子科技大学 XIDIAN UNIVERSITY | 人工智能学院 School of Artificial Intelligence

计算智能导论

（遗传算法求解无约束优化）

授课老师：张梦璇

学院：人工智能学院

班级：1820013

专业：智能科学与技术

姓名：刘继垚

学号：18200100176

1 遗传算法描述

基于对自然界中生物遗传与进化机理的模仿，针对不同的问题，很多学者设计了许多不同的编码方法来表示问题的可行解，开发出了许多种不同的遗传算子来模仿不同环境下的生物遗传特性。基于这个特点，Goldberg 总结出厂一种统--的最基本的遗传算法——基本遗传算法（Simple Ge-netic Algorithms，简称 SGA)。基本遗传算法只使用选择算子、交叉算子和变异算子这三种基本遗传算子，其遗传进化操作过程简单，容易理解，是其他-些遗传算法的雏形和基础，它不仅给各种遗传算法提供了-个基本框架，同时也具有一定的应用价值。

1.1 基本遗传算法的构成要素

(1）**染色体编码方法**。基本遗传算法使用固定长度的二进制符号串来表示群体中的个体，其等位基因是由二值符号集{0,1}所组成的。初始群体中各个个体的基因值可用均匀分布的随机数来生成。如:

$$X = 1000101010$$

就可表示一个个体，该个体的染色体长度是 n=10。

(2）**个体适应度评价**。基本遗传算法按与个体适应度成正比的概率来决定当前群体中每个个体遗传到下一代群体中的机会多少。为正确计算这个概率，这里要求所有个体的适应度必须为正数或零。根据不同种类的问题，必须预先确定好由目标函数值到个体适应度之间的转换规则，特别是要预先确定好当目标函数值为负数时的处理方法。

(3）**遗传算子**。基本遗传算法使用下述三种遗传算子:

- 选择运算使用比例选择算子
- 交叉运算使用单点交叉算子
- 变异运算使用基本位变异算子或均匀变异算子

（4）**基本遗传算法的运行参数**。基本遗传算法有下述 4 个运行参数需要提前设定:

- M:群体大小，即群体中所含个体的数量，--般取为 20~100。
- T:遗传运算的终止进化代数，一般取为 100~500。
- p_c:交叉概率，一般取为 0.4~0.99。
- p_m:变异概率，一般取为 0.0001~0.1。

2 基本遗传算法的实现

根据上面对基本遗传算法构成要素的分析和算法描述，我们可以很方便地用计算机语言来实现这个基本遗传算法。附录给出了其 matlab 语言源程序。现对具体实现过程中的何题作以下说明。

2.1 个体适应度评价

在遗传算法中，以个体适应度的大小来确定该个体被遗传到下--代群体中的概率。个体的适应度越大，该个体被遗传到下一代的概率也越大;反之，个体的适应度越小，该个体被遗传到下一代的概率也越小。基本遗传算法使用比例选择算子来确定遗传概率，要求所有个体的适应度必须为正数或零，不能是负数。

对于求目标函数最小值的优化问题，理论上只需简单地对其增加一个负号就可将其转化为求自标函数最大值的优化问题，即:

$$\min f(X) = \max(-f(X))$$

当优化目标是求函数最大值，并且目标函数总取正值时，可以直接设定个体的适应度 F(x）就等干相应的目标函数值 f (x)，即:

$$F(X) = f(X)$$

但实际优化问题中的目标函数值有正也有负， 为满足适应度取非负值的要求，基本遗传算法一般采用下面两种方法之一将目标函数值 f (x）变换为个体的适应度 F(X)。

方法一:对于求目标函数最大值的优化问题，变换方法为:

$$F(X) = \begin{cases} f(X) + C_{\min}, & \text{if } f(X) + C_{\min} > 0 \\ 0, & \text{if } f(X) + C_{\min} \leqslant 0 \end{cases}$$

式中，Cmin 为一个适当地相对比较小的数，它可用下面几种方法一来选取。

- 预先指定的一个较小的数。
- 进化到当前代为止的最小目标函数值。
- 当前代或最近几代群体中的最小旦标函数值。

方法二:对于求目标函数最小值的优化问题，变换方法为:

$$F(X)=\begin{cases} C_{max}-f(X), & \text{if } f(X)<C_{max} \\ 0, & \text{if } f(X)\geqslant C_{max} \end{cases}$$

式中，Cmax 为一个适当地相对比较大的数，它可用下面几种方法来选取。

- 预先指定的一个较大的数。
- 进化到当前代为止的最大目标函数值。
- 当前代或最近几代群体中的最大目标函数值。

2.2 比例选择算子

选择算子或复制算子的作用是从当前代群体中选择出一些比较优良的个体，并将其复制到下一代群体中。最常用和最基本的选择算子是比例选择算子。所谓比例选择算子，是指个体被选中许遗传到下一代群体中的概率与该个体的适应度大小成正比。

比例选挥实际上是一种有退还随机选择，也叫做赌盘(Roulette Wheel）选择,因为这种选择方式与赌博中的赌盘操作原理颇为相似。

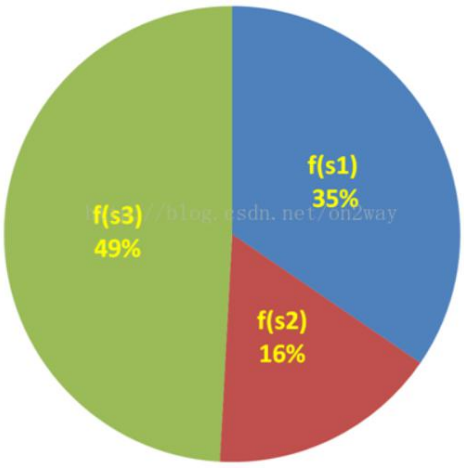


图 1 适应度轮盘

如图 1 所示为一赌盘示意图。整个赌盘被分为大小不同的-些扇面，分别对

应着价值各不相同的一些赌博物品。指针指向各个扇面的概率却是可以估计的，它与各个扇面的圆心角大小成正比:圆心角越大，停在该扇面的可能性也越大;圆心角越小，停在该扇面的可能性也越小。与此类似，在遗传算法中，整个群体被各个个体所分割，各个个体的适应度在全部个体的适应度之和中所占比例也大小不一，这个比例值瓜分了整个赌盘盘面，它们也决定了各个个体被遗传到下一代群体中的概率。

比例选择算子的具体执行过程是:

(1）先计算出群体中所有个体的适应度的总和。

(2）其次计算出每个个体的相对适应度的大小，它即为各个个体被遗传到下--代群体中的概率。

(3）最后再使用模拟赌盘操作（即 0 到 1 之间的随机数）来确定各个个体被选中的次数。

2.3 单点交叉算子

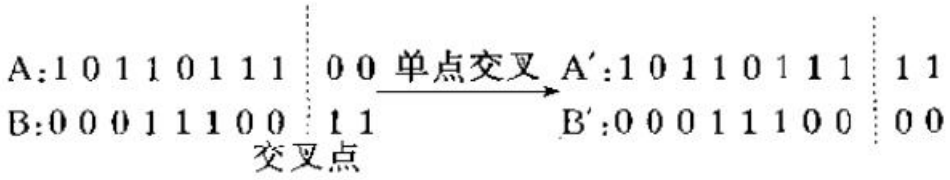
单点交叉算手是最常用和最基本的交叉操作算子。单点交叉算子的具体执行过程如下。

(1）对群体中的个体进行两两随机配对。若群体大小为 M，则共有⌊$M/2$⌋对相互配对的个体组。

(2）对每---对相互配对的个体，随机设置某一基因座之后的位置为交叉点。染色体的长度为 n，则共有（n-1）个可能的交叉点位置。

(3）对每一对相互配对的个体，依设定的交叉概率 p_c 在其交叉点处相互交换两个个体的部分染色体，从而产生出两个新的个体。

单点交叉运算的示意如下所示:



2.4 基本位变异算子

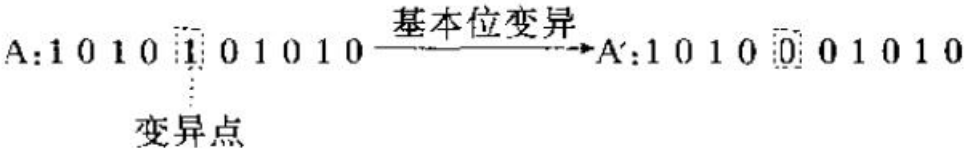
基本位变异算子是最简单和最基本的变异操作算子。对于基本遗传算法中用二进制编码符号串所表示的个体，若需要进行变异操作的某--基因座上的原有基

因值为0，则变异操作将该基因值变为1；反之，若原有基因值为1，则变异操作将其变为Q。

基本位变异算子的具体执行过程是:

(1）对个体的每一个基因座，依变异概率p_m。指定其为变异点。

(2)对每一个指定的变异点,对其基因值做取反运算或用其他等位基因值来代替，从而产生出一个新的个体。基本位变异运算的示意如下所示:



2.5 基本遗传算法应用步骤

第一步:确定决策变量及其各种约束条件，即确定出个体的表现型和问题的解空间。

第二步:建立优化模型，即确定出目标函数的类型（是求目标函数的最大值还是求目标函数的最小值?）及其数学描述形式或量化方法。

第三步:确定表示可行解的染色体编码方祛，也即确定出个体的基因型及遗传算法的搜索空间。

第四步:确定解码方法，即确定出由个体基因型x到个体表现型X的对应关系或转换方法。

第五步:确定个体适应度的量化评价方法，即确定出由目标函数值f（x）到个体适应度F(X）的转换规则。

第六步:设计遗传算子，即确定出选择运算、交叉运算、变异运算等遗传算子的具体操作方法。

第七步:确定遗传算法的有关运行参数，M，T，p_c，p_m等参数。

由上述构造步骤可以看出，可行解的编码方法、遗传算子的设计是构造遗传算法时需要考虑的两个主要问题，也是设计遗传算法时的两个关键步骤。对不同的优化问题需要使用不同的编码方法和不同操作的遗传算子,它们与所求解的具体问题密切相关,因而对所求解问题的理解程度是遗传算法应用成功与否的关键。图2所示为遗传算法的主要构造过程示意图。

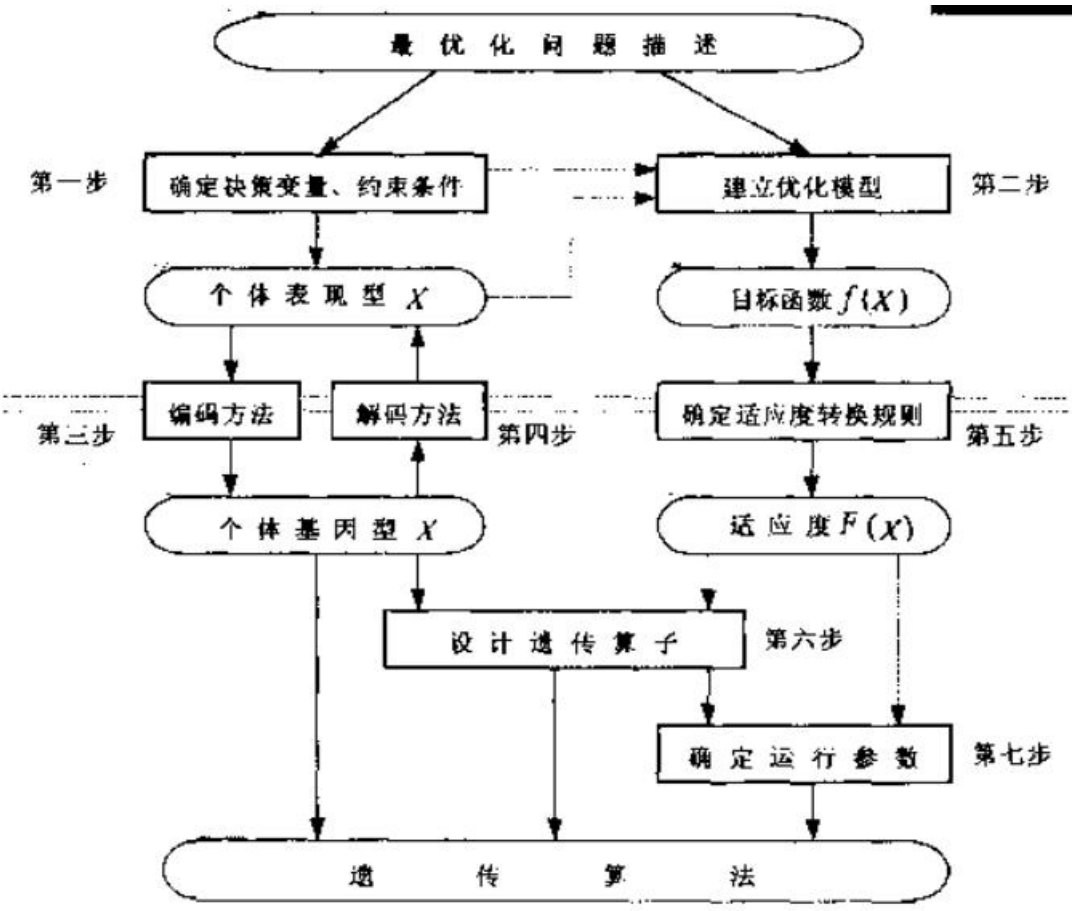


图 2 遗传算法的主要构造过程

3.遗传算法在多峰函数中的应用

参数设置：

表 1 参数设置

迭代次数	100
种群大小	100
二进制编码长度	10
交叉概率	0.6
变异概率	0.001

3.1 函数 1：

函数图像：

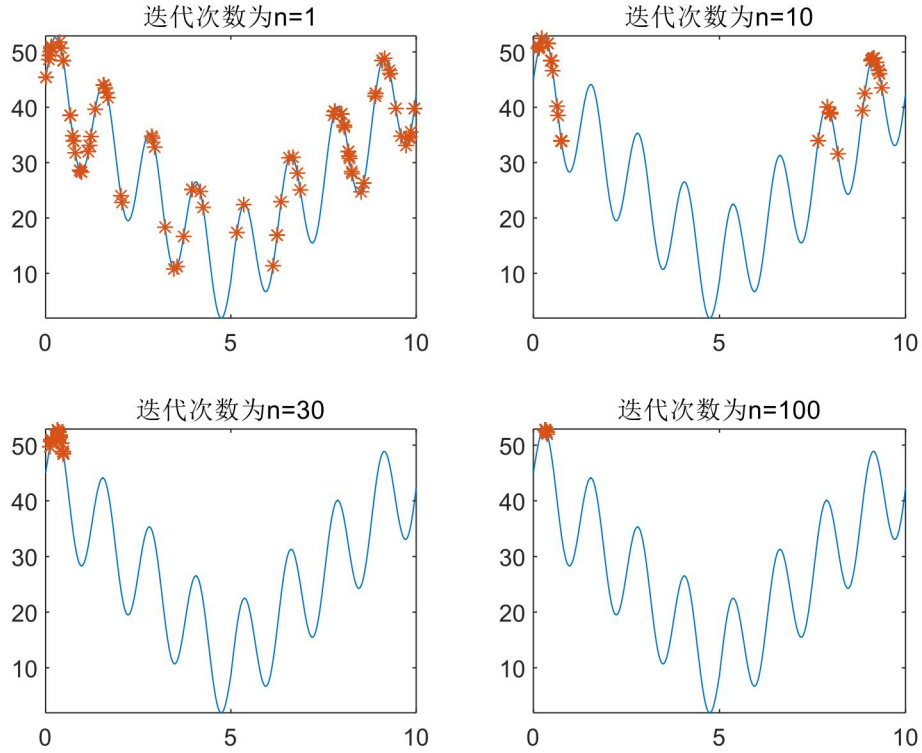


图 3 函数 1 的遗传算法搜索结果

最优点：0.28 最优值：52.90

3.2 函数 2：

$$y=5\sin(3x)-7|5-0.5x|+40,\quad x\in[0,20]$$

函数图像：

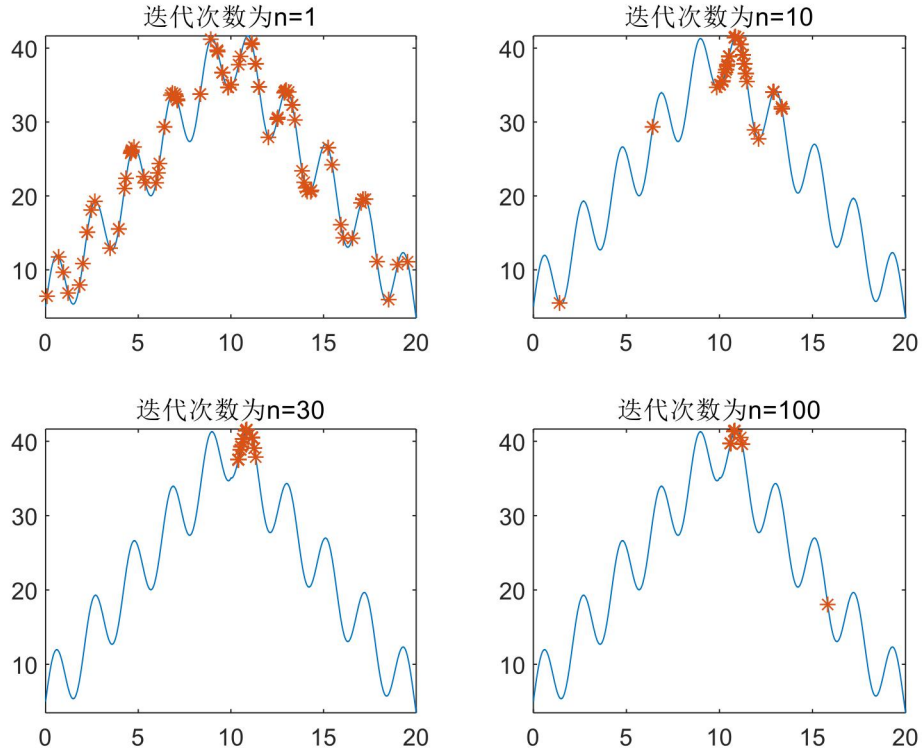


图 4 函数 2 的遗传算法搜索结果

最优点：10.81 最优值：41.42

3.3 函数 3：

$$y=\frac{4\sin(3x)}{x}+10,\quad x\in[0,10]$$

优化函数图像：

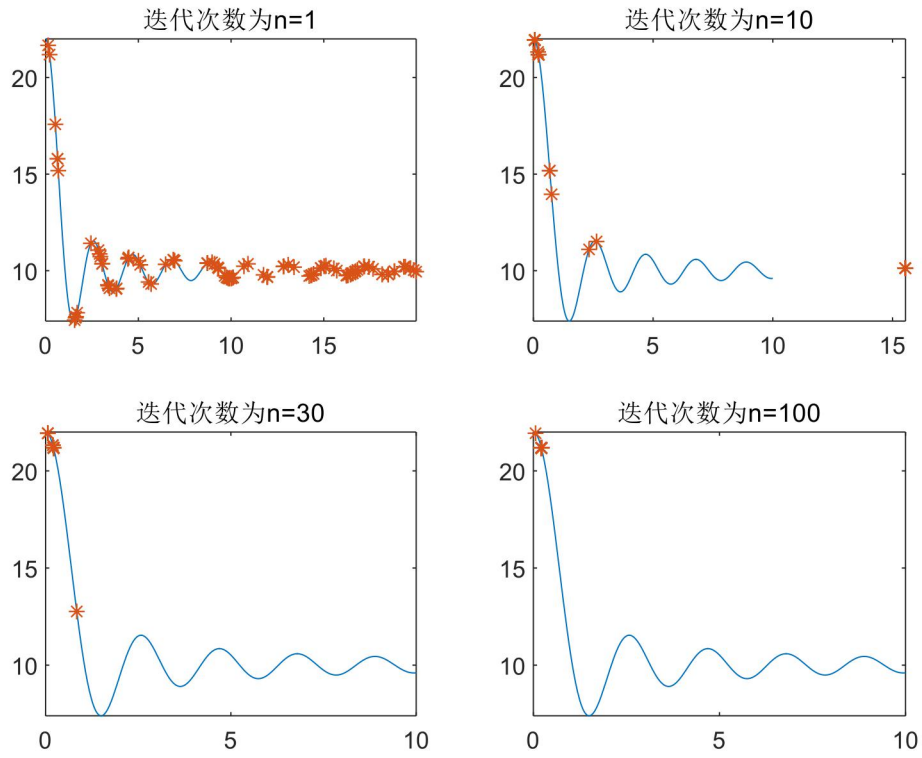


图 5 函数 3 的遗传算法搜索结果

最优点：0.06 最优值：21.94

3.4 函数 4：

$$y = x^2 - 4x^2\sin(0.4x) + 50000,\quad x \in [0, 100]$$

优化函数图像：

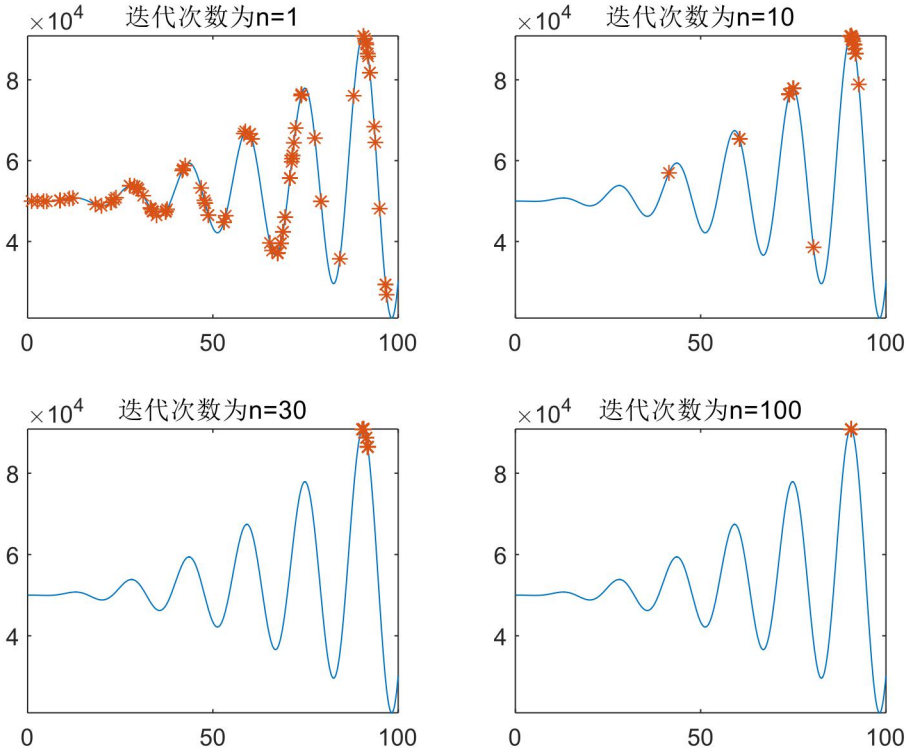


图 6 函数 4 的遗传算法搜索结果

最优点：90.62 最优值：90827.67

3.5 函数 5：

$$y=-7|x-500|-10|x-800|+200\log_2(x)\cos(1-0.05x),\quad x\in[0,1000]$$

优化函数图像：

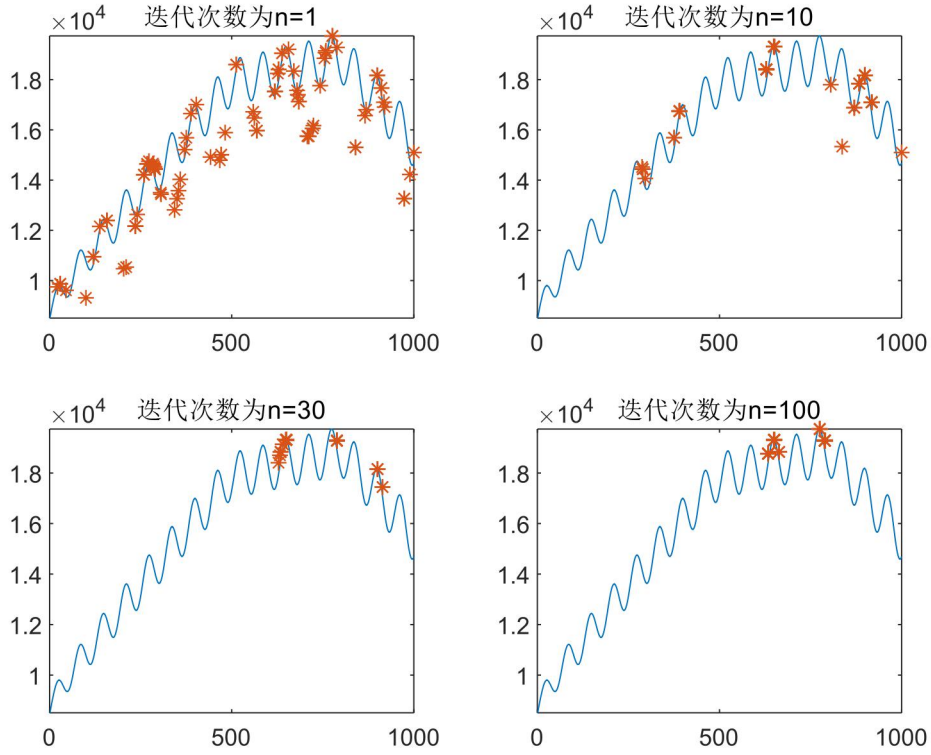


图 7 函数 5 的遗传算法搜索结果

```
The best X is --->>663.73
The best Y is --->>19741.80
```

3.6 基本遗传算法结果分析及总结

由上述实验可以看出，基本遗传算法搜素多峰函数的最值具有较高的有效性，能够较为准确的找出最优点。在本次实验中，我们只针对二维平面函数进行编码求解，实验结果表明我们的基本遗传算法是非常有效的。如果要针对高维函数进行遗传算法求解，只需要对代码稍加修改每一维进行编码，之后进化求解其最优值。

4. 遗传算法实现代码（matlab）

代码说明：对于不同函数，只需要修改主函数中注明的地方和子函数Fit_value中需要修改的地方

4.1 主函数

```
clear;
clc;
%% 设定参数
count=100;%迭代次数
N=100; %种群大小
length=10;%二进制编码长度
pc=0.6;%交叉概率
pm=0.001;%变异概率
%% 初始化种群
herd=Init(N,length) %初始化二进制表示的种群矩阵（矩阵一行为一个个体）
herd2 = binary2decimal(herd) %十进制表示的种群
j=1;
for i=1:count
    %计算每个个体适应度
    fitvalue=Fit_value(herd);
    %选择操作
    newherd = Selection(herd,fitvalue);
    %交叉操作
    newherd = Crossover(newherd,pc);
    %变异操作
    newherd = Mutation(newherd,pm);
    %更新种群
    herd = newherd;
    %选择最优个体及其最优适应度
    [x2,bestfit] = Best(herd,fitvalue);
    bestindividual = binary2decimal(x2);
    %更新后种群的个体和适应度
    x1 = binary2decimal(newherd);
    y1 = Fit_value(newherd);

    %绘制图像
%    if mod(i-1,10) == 0 || i==100
    if i==1|| i==10 || i==30||i==100
        %figure;
        subplot(2,2,j);
        %=======不同函数修改这里(函数值必须>0)========
        %fplot(@(x)10*sin(5*x)+7*abs(x-5)+10,[0 10]); %函数 1
        %fplot(@(x)5*sin(3*x)+7*(-abs(5-0.5*x))+40,[0 20]);%函数 2
        %fplot(@(x)4.*sin(3*x)./(x)+10,[0 10]);%函数 3
```

```
        %fplot(@(x)x.^2-4*x.^2.*sin(0.4*x)+50000,[0 100]);%函数 4

fplot(@(x)-7*abs(x-500)-10*abs(x-800)+200*log2(x).*(cos(1-0.05*x)).^2+2
0000,[0 1000]); %函数 5
        %=======不同函数修改这里(函数值必须>0)========
        hold on;
        plot(x1,y1,'*');
        title(['迭代次数为 n=' num2str(i)]);
        %plot(x1,y1,'*');
        j=j+1;
    end
end
hold off;
fprintf('The best X is --->>%5.2f\n',bestindividual);
fprintf('The best Y is --->>%5.2f\n',bestfit);
```

4.2 初始化一个种群

```
%% 初始化种群
function herd=Init(N,length)
    %N:种群大小
    %length：染色体长度-->>转化的二进制长度
    %herd：就是每行是一个个体，列数是染色体长度
    herd=round(rand(N,length)) ;
    %round 就是四舍五入;rand(3,4)生成 3 行 4 列的 0-1 之间的随机数
end
```

4.3 计算每个个体适应度

```
function [objvalue] = Fit_value(pop)
    x = binary2decimal(pop);
    %转化二进制数为 x 变量的变化域范围的数值
   %===============不同函数修改这里(函数值必须>0)↓===========
   %objvalue=10*sin(5*x)+7*abs(x-5)+10;%目标函数 1
   %objvalue=5*sin(3*x)+7*(-abs(5-0.5*x))+40;%目标函数 2
   %objvalue= 4.*sin(3*x)./(x)+10;%函数 3
   %objvalue=x.^2-4*x.^2.*sin(0.4*x)+50000;%目标函数 4

objvalue=-7*abs(x-500)-10*abs(x-800)+200*log2(x).*(cos(1-0.05*x))+20000;
%目标函数 5
   %objvalue=sin(x)+2*x;%目标函数 2
   %===============不同函数修改这里(函数值必须>0)↑===========
```

4.4 进行自然选择，选出基因好的个体进行复制

```
%% 选择算法
function [newherd]=Selection(herd,fitvalue)
```

```
%herd:种群矩阵
%fit_value：适应度列向量
%轮盘构造
[m,n]=size(herd);
sum_value=sum(fitvalue);
p=fitvalue/sum_value;
p = cumsum(p);%第 n 行的值为前 n 行的累加值
ms = sort(rand(m,1));%生成从小到大排列 0~1 随机向量
j = 1;
i = 1;
while i<=m
    if(ms(i))<p(j)
        newherd(i,:)=herd(j,:);
        i= i+1;
    else
        j=j+1;
    end
end
end
```

4.5 交叉算子

```
%% 交叉算子
function [newherd] = Crossover(herd,pc)
%交叉变换
%输入变量：pop：二进制的父代种群数，pc：交叉的概率
%输出变量：newpop：交叉后的种群数
[m,n] = size(herd);
newherd = ones(size(herd));
for i = 1:2:m-1
    if(rand<pc) %rand 表示 0~1 的一个随机数
        cpoint = round(rand*n);
        newherd(i,:) = [herd(i,1:cpoint),herd(i+1,cpoint+1:n)];
        newherd(i+1,:) = [herd(i+1,1:cpoint),herd(i,cpoint+1:n)];
    else
        newherd(i,:) = herd(i,:);
        newherd(i+1,:) = herd(i+1,:);
    end
end
```

4.6 变异算子

```
%% 变异算子
%函数说明
%输入变量：herd：二进制种群，pm：变异概率
%输出变量：newherd 变异以后的种群
```

```
function [newherd] = Mutation(herd,pm)
[m,n] = size(herd);
newherd = ones(size(herd));
for i = 1:m
    if(rand<pm)
        mpoint = round(rand*n);
        if mpoint <= 0;
            mpoint = 1;
        end
        newherd(i,:) = herd(i,:);
        if newherd(i,mpoint) == 0
            newherd(i,mpoint) = 1;
        else newherd(i,mpoint) == 1
            newherd(i,mpoint) = 0;
        end
    else newherd(i,:) = herd(i,:);
    end
end
```

4.7 求最优适应度

```
%% 求最优适应度函数
%输入变量：herd:种群，fitvalue:种群适应度
%输出变量：bestindividual:最佳个体，bestfit:最佳适应度值
function [bestindividual,bestfit] = Best(herd,fit_value)
[m,n] = size(herd);
bestindividual = herd(1,:);
bestfit = fit_value(1);
for i = 2:m
    if fit_value(i)>bestfit
        bestindividual = herd(i,:);
        bestfit = fit_value(i);
    end
end
```

4.8 二进制转十进制

```
%% 二级制种群转十进制种群
function herd2 = binary2decimal(herd)
    %herd：一个种群二进制矩阵
    %二进制转十进制 转为列向量
    [px,py]=size(herd);
    for i = 1:py
        herd1(:,i) = 2.^(py-i).*herd(:,i);
    end
    %sum(.,2)对行求和，得到列向量
```

```
    temp = sum(herd1,2);
    %=========修改测试函数的取值范围============
    %herd2 = temp*10/1023;  %范围为0~10  %测试函数1
    % herd2 = temp*20/1023;  %范围为0~20  %测试函数2
    %herd2 = temp*10/1023;  %范围为0~10  %测试函数3
    herd2 = temp*100/1023;  %范围为0~100  %测试函数4
    herd2 = temp*1000/1023;  %范围为0~1000  %测试函数5
    %=======================================
    %转换到0~10
```

参考文献

[1] 周明等.遗传算法原理及应用[m]. 国防工业出版社出版,1999,(3):1-64.

1999 年国防工业出版社出版