A Programming of Genetic Algorithm in Matlab7.0

Cheng Guo (Corresponding author)
Department of Mathematics and Physics, XiaMen University of Technology
LiGong Road 600, JiMei Area, XiaMen, FuJian Province 361024, China
E-mail: guocheng@xmut.edu.cn

Xiaoyong Yang
School of Mathematics and Physics, HuaiHai Institute of Technology
Lian yungang, Jiangsu 222005, China
E-mail:163yangxiaoyong@163.com

Abstract

The genetic algorithm is briefly introduced and its complete programming is provided in detail by MATLAB7.0. In addition, the application in optimization of functions and solution of equation is shown through three examples and the method of avoiding local optimization by increasing the value of pm is also discussed.

Keywords: Population, Encoding, Decoding, Cross over, Mutation, Selection

1. Introduction of genetic algorithm

Genetic Algorithm (GA) is a global optimization algorithm derived from evolution and natural selection. Although genetic algorithm cannot always provide optimal solution, it has its own advantages (Liu yong, Kang lishan & Chen yuping. 1997) and is a powerful tool for solving complex problems (Xi yugeng, Chai tianyou & Yun weimin. 1996).

The basic thought of Genetic algorithm:

1) Randomly producing a original population whose number of individuals is a constant N.

2) Producing next generation by crossing over and mutation among individuals.

3) Forming the new population of N individuals from the generation of 2)

4) Producing the next population by repeating the step2) and 3) until obtaining the individual which satisfies conditions.

2. MATLAB programming for genetic algorithm

In order to understand the sense of the MATLAB programming for genetic algorithm, giving the following instances.

Instance one: seeking the maximum of the function $y = f(x_1, x_2, x_3) = 6 - x_1^2 - x_2^2 - x_3^2$.

2.1 individual and population

For instance one, $X = (x_1, x_2, x_3)$ is called individual, and the function $y = f(x_1, x_2, x_3)$ is written as $y = f(X)$, and $f(X)$ is called fitting value of individual X .For example, the fitting value $f(1,1,1) = 3$ for the individual $(1,1,1)$ and lots of individuals form a population, such as, $(1,2,3), (0,0,0), (-1,0,2.4), (0,-0.5,-1.7)$ is a population of four individuals.

In genetic algorithm, the individuals of next population are chosen by the fitting values of individuals, by maintaining the individuals whose fitting values are greater than others. In addition, the fitting values are also as the extermination end condition of genetic algorithm (Liang jiye. 1999). And when the fitting values don't continue to increase, exterminating the programming and the individual with the greatest value is seen as the optimal solution. The number of the generation of populations is another termination condition for genetic algorithm. For example, when after 100 generation of population, terminate the programming and the individual with the greatest fitting value in the last generation is as the optimal solution.

2.2 encoding

In genetic algorithm, coding is expressing the individual by the binary strings of 0s and 1s. In the instance one, every individual has there dimensions, and every dimension is expressed by a 8- bit string of 0s and 1s, so every individual is expressed as a 24-bit string of 0s and 1s, showing as $\overbrace{01101111}^{x_1}\overbrace{00010101}^{x_2}\overbrace{00110110}^{x_3}$

Programming 1

```
function pop=encoding(popsize,stringlength,dimension)
pop=round(rand(popsize,dimension*stringlength+1));
```

Programming 1 is an encoding function by MATLAB and it randomly produces an encoded original population. Pop in the function encoding is a matrix whose every row indicates an encoded individual and the total number of rows is denoted as popsize. And dimension is the number of dimension of an individual, stringlength is the number of the bits of binary string of encode individual. In the instance one, dimension=3,stringlength=8.In the programming1 ,the last bit of every row record the fitting value of an individual encoded by this row.

2.3 cross-over

Randomly choosing two individuals from pop, and changing the bits of the same section of the two individuals.

Programming 2 for cross-over

```
function new_pop=cross_over(pop,popsize,stringlength,dimension)
match=round(rand(1,popsize)*(popsize-1))+1;
for i=1:popsize
    [child1,child2]=cross_running(pop(i,:),pop(match(i),:),stringlength,dimension);
    new_pop(2*i-1:2*i,:)=[child1;child2];
end
function [child1,child2]=cross_running(parent1,parent2,stringlength,dimension)
cpoint=round((stringlength-1)*rand(1,dimension))+1;
for j=1:dimension
child1((j-1)*stringlength+1:j*stringlength)=[parent1((j-1)*stringlength+1:(j-1)*stringlength+cpoint(j))
parent2((j-1)*stringlength+cpoint(j)+1:j*stringlength)];
child2((j-1)*stringlength+1:j*stringlength)=[parent2((j-1)*stringlength+1:(j-1)*stringlength+cpoint(j))
parent1((j-1)*stringlength+cpoint(j)+1:j*stringlength)];
end
```

Programming 2 includes two functions written by MATLAB which complete the course of mutation by change the part of the binary strings of the chosen individuals.

2.4 mutation

Mutation also simulates biologic evolution mechanism. For the individual to mutate, randomly choose the point to mutate which means the bit of the individual encoded binary string, then change 0 to 1 and change 1 to 0. Pm is the probability of mutation and it is not great in nature and in programming 2, pm=0.05. For every individual, a number of probabilities of mutation are randomly given by the computer. If the giver number is not greater than pm, the individual mutates, otherwise don't mutate.

Programming 3 for mutation

```
function new_pop=mutation(new_pop,stringlength,dimension,pm)
new_popsize=size(new_pop,1);
for i=1:new_popsize
    if rand<pm          mpoint=round(rand(1,dimension)*(stringlength-1))+1;
        for j=1:dimension
            new_pop(i,(j-1)*stringlength+mpoint(j))=1-new_pop(i,(j-1)*stringlength+mpoint(j));
        end
    end
end
```

2.5 decoding

Decoding change the encoding binary strings into decimal number, and then computes fitting values for individuals. Decoding is shown as following through instance one above.

In instance one, each dimension of an individual $X=(x_1,x_2,x_3)$ has a boundary, denoting as $x\_bound=[a_1,b_1,a_2,b_2,a_3,b_3]$, $a_1 \le x_1 \le b_1$, $a_2 \le x_2 \le b_2$, $a_3 \le x_3 \le b_3$, an encoded individual :

$$\overbrace{01101111}^{x_1}\overbrace{00010101}^{x_2}\overbrace{00110110}^{x_3}\ .$$

Encoded binary string for x_1 : 01101111.

Decoded decimal number:

$$\frac{0\times2^7+1\times2^6+1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0}{2^8-1}\times(b_1-a_1)+a_1$$

Encoded binary string for x_2 : 00010101

Decoded decimal number:

$$\frac{0\times2^7+0\times2^6+0\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0}{2^8-1}\times(b_2-a_2)+a_2$$

Encoded binary string for x_3 : 00110110

Decoded decimal number:

$$\frac{0\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0}{2^8-1}\times(b_3-a_3)+a_3$$

Programming 4 for decoding

```
function pop=decoding(pop,stringlength,dimension,x_bound)
popsize=size(pop,1);
temp=2.^(stringlength-1:-1:0)/(2^stringlength-1);
for i=1:dimension
    bound(i)=x_bound(i,2)-x_bound(i,1);
end
for i=1:popsize
    for j=1:dimension
        m(:,j)=pop(i,stringlength*(j-1)+1:stringlength*j);
    end
    x=temp*m;
    x=x.*bound+x_bound(:,1)';
    pop(i,dimension*stringlength+1)=funname(x);
end
```

2.6 selection

Selection is the proceeding through which a new population is formed by choosing the individuals with greater fitting values and eliminating the individuals with smaller fitting values. There are two strategies: the first strategy is that maintaining the individuals with greatest fitting values into the next population; the second strategy is that choosing the individuals to next population by bet ring arithmetic(Zhou ming & Sun shudong. 1999) which guarantee direct ratio between chosen probability and fitting value of the individual. Through selection the fitting values of population is increased constantly and isn't decreased.

Programming 5 for selection

```
function selected=selection(pop,popsize,stringlength,dimension)
popsize_new=size(pop,1);
```

```
r=rand(1,popsize);
fitness=pop(:,dimension*stringlength+1);
fitness=fitness/sum(fitness);
fitness=cumsum(fitness);
for i=1:popsize
    for j=1:popsize_new
        if r(i)<=fitness(j)
            selected(i,:)=pop(j,:);
            break;
        end
    end
end
```

3. Instances

Main programming 6 for solving instances

```
clear;clc;
popsize=10;dimension=3;stringlength=8;x_bound=[-2,3;-2,4;-1,1];pm=0.05;
pop=encoding_guo(popsize,stringlength,dimension);
pop=decoding_guo(pop,stringlength,dimension,x_bound);
[choice_number,choice_k]=max(pop(:,stringlength*dimension+1));
choice=pop(choice_k,:);
for i=1:1000        new_pop=cross_over(pop,popsize,stringlength,dimension);
pop=mutation_guo(new_pop,stringlength,dimension,pm);
pop=decoding_guo(pop,stringlength,dimension,x_bound);
[number,k]=max(pop(:,stringlength*dimension+1));
    if choice_number<number
        choice_number=number;
        choice_k=k;
        choice=pop(choice_k,:);
    end         pop=selection_guo(pop,popsize,stringlength,dimension);
    [number,m]=min(pop(:,stringlength*dimension+1));
    pop(m,:)=choice;
end [value,x]=result_guo(pop,stringlength,dimension,x_bound);
```

Programming 7

```
function [value,x]=result_guo(pop,stringlength,dimension,x_bound)
[value,k]=max(pop(:,stringlength*dimension+1));
temp=2.^(stringlength-1:-1:0)/(2^stringlength-1);
for i=1:dimension
    bound(i)=x_bound(i,2)-x_bound(i,1);
end
for j=1:dimension
    m(:,j)=pop(k,stringlength*(j-1)+1:stringlength*j);
end
x=temp*m;
```

```
x=x.*bound+x_bound(:,1)'
```

programming 8

```
function y=funname(x)
y=6-x(1)^2-x(2)^2-x(3)^2;
```

For different instances, programmings need to be rewritten. And here is only for instance one.

Instance two: finding the maximum of function $y = f(x_1, x_2, x_3) = 6 - x_1 - x_2 - x_3$,

Running result:

```
value = 5.9995   x =   -0.0159         0        0.0159
```

This shows that the maximum is 5.9995, and $x_1 = -0.0159, x_2 = 0, x_3 = 0.0159$ after one hundred generations and it closely approaches the real maximum 6 of the function.

Instance three: finding the maximum of function $y = f(x) = \sin(5x) - 2\cos(3x) + 10, x \in [0,10]$, the figure of the function is as the following Figure 1:

Programming for this instance in MATLAB

```
function y=funname(x)
y=sin(5*x)-2*cos(3*x)+10;
```

in programming 6, changing the following

```
popsize=10;dimension=3;stringlength=8;x_bound=[-2,3;-2,4;-1,1];pm=0.05;
```

as: popsize=10;dimension=1;stringlength=8;x_bound=[0,10];pm=0.5;

running result: value =12.9427   x =5.2941

Through this instance we can validate the power of global optimal seeking and if let pm=0.05, run the programming 6, local optimal solution is found. But increasing pm, let pm=0.5, global optimal solution is obtained. This instance shows that in order to avoid the local optimal solution, pm can be increased. This is the same as biological evolution mechanism: when mutation is fastened, the evolution speed of biological group is increased.

Instance four: Finding $x, x \in [0,10]$ for function $f(x) = \sin(5x) - 2\cos(3x) + 10$, satisfying $f(x) = 11$. The function in instance 4 is the same as that in instance 3. And by the figure of this function, the value is not only one.

Programming for instance four in MATLAB

```
function y=funname2(x)
y=-abs(sin(5*x)-2*cos(3*x)+10-11);
```

in programming 6, changing the following

```
popsize=10;dimension=3;stringlength=8;x_bound=[-2,3;-2,4;-1,1];pm=0.05;
```

as: popsize=10;dimension=1;stringlength=8;x_bound=[0,10];pm=0.5;

running result: value =-0.0130   x =1.5686

when x_bound=[0,1], value =-0.0035   x =0.9922; $f(0.9922) = 11.0036$

when x_bound=[1,2], value =-0.0106   x =1.5725; $f(1.5725) = 10.9897$

when x_bound=[2,3], value =-0.0150   x =2.6667; $f(2.6667) = 10.9853$

Through changing x_bound, the nine values satisfying $0 \le x \le 10, f(x) = 11$ are:

0.9922,1.5725,2.6667,3.3098,4.9569,5.6196,7.2745,8.9529,9.5922.

This instance shows that genetic algorithm can also be used to solve equations if the boundaries of the variables are given.

4. Conclusion

MATLAB codes given in literature(Liu guohua, Bao hong & Li wenchao. 2001)(Yin ming, Zhang xinghua & Dai xianzhong. 2000) are incomplete, and somewhere have errors. This article provides the complete original codes in MATLAB which can be directly run through MATLAB7.0. The given instances in this article show that the genetic algorithm can be applied to find optimal solution and to solve equations and indicate that the genetic algorithm is a powerful global searching tool. In order to avoid local optimal solution, we can increase individual rate of mutation and increase the hereditary generations of population.

References

Liang jiye. (1999). The research of common problems in genetic algorithm application. R*esearch of Computer Application,* 1999 (7): 20-21.

Liu guohua, Bao hong & Li wenchao. (2001). The genetic algorithm programming in MATLAB. *Research of Computer Aapplication,*2001(8):80-82.

Liu yong, Kang lishan & Chen yuping. (1997). *Nonnumerical parallel algorithm(second volumn)—genetic algorithm.* Beijing:science press,1997.

Xi yugeng, Chai tianyou & Yun weimin. (1996). Summarization of genetic algorithm. *Control Theory and Application,* 1996(6):697-708

Yin ming, Zhang xinghua & Dai xianzhong. (2000). The programming of genetic algorithm in MATLAB. *Application of Electronic Technology,* 2000(1):9-11.

Zhou ming & Sun shudong. (1999). *Theory and Application of Genetic Algorithm*. BeiJing: national independence industry press, 1999.

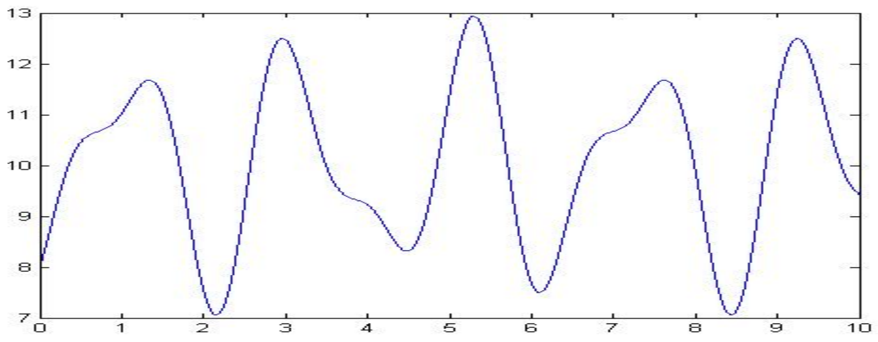

Figure 1. curve for $\sin(5x) - 2\cos(3x) + 10, x \in [0,10]$