

## 附录 1、标准微粒群算法源程序

/\*该程序用于计算 Goldstein-Price 函数

$$f_1(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ - 2 \leq x_1, x_2 \leq 2$$

的函数最小值 \*/

//库文件

#include"stdio.h"

#include"stdlib.h"

#include"time.h"

#include"math.h"

//随机数定义

#define rdint(i) (rand()%(int)(i))

#define rdft() (float)((double)rdint(16384)/(16383.0))

#define rnd(a,b) (rdint((int)(b)-(int)(a)+1)+(int)(a))

//宏定义

#define POPSIZE 20

#define DIMENSION 2

//全局变量定义

float W=1.0;

float C1=1.8;

float C2=1.8;

float VMAX=2;

float XMIN=-2.0;

float XMAX=2.0;

float P[DIMENSION];

float PBEST;

struct indi

{

float number[DIMENSION];

float best[DIMENSION];

float bestfitness;

float fitness;

float speed[DIMENSION];

}individual[POPSIZE];

void initiate(void);

```

void calculation(int number);
void globalbest(int number);
void localbest(int number);

```

//程序初始化定义

```

void initiate()
{
    int i,j;
    for(i=0;i<POPSIZE;i++)
        for(j=0;j<DIMENSION;j++)
            individual[i].number[j]=rdft()*(XMAX-XMIN)+XMIN;
    for(i=0;i<POPSIZE;i++)
        for(j=0;j<DIMENSION;j++)
            individual[i].speed[j]=VMAX*rdft();
    for(i=0;i<POPSIZE;i++)
        for(j=0;j<DIMENSION;j++)
            individual[i].best[j]=individual[i].number[j];
    for(i=0;i<POPSIZE;i++)
        calculation(i);
    for(i=0;i<POPSIZE;i++)
        individual[i].bestfitness=individual[i].fitness;
    globalbest(0);
}

```

//微粒历史最优位置修改程序

```

void localbest(int number)
{
    int i;
    if(individual[number].bestfitness>individual[number].fitness)
        for(i=0;i<DIMENSION;i++)
            individual[number].best[i]=individual[number].number[i];
    individual[number].bestfitness=individual[number].fitness;
}

```

//种群历史最优位置修改程序

```

void globalbest(int number)
{
    int i,j;
    float s=0;
    int flag=0;
    if(number==0)
    {
        s=individual[0].fitness;
        flag=0;
        for(i=1;i<POPSIZE;i++)
            if(individual[i].fitness<s)

```

```

        {
            s=individual[i].fitness;
            flag=i;
        }
    for(i=0;i<DIMENSION;i++)
        P[i]=individual[flag].number[i];
    PBEST=individual[flag].fitness;
}
else
{
    for(i=0;i<POPSIZE;i++)
        if(individual[i].bestfitness<PBEST)
        {
            for(j=0;j<DIMENSION;j++)
                P[j]=individual[i].best[j];
            PBEST=individual[i].bestfitness;
        }
}
}

```

//适应值函数计算程序

```
void calculation(int num)
```

```

{
    int i;
    float s=0.0,h=0.0;
    s=pow(individual[num].number[0]+individual[num].number[1]+1.0,2.0)*(19.0-14.0*individual[num].number[0]+3.0*pow(individual[num].number[0],2.0)-14.0*individual[num].number[1]+6.0*individual[num].number[0]*individual[num].number[1]+3.0*pow(individual[num].number[1],2.0))+1.0;
    h=30.0+pow(2.0*individual[num].number[0]-3.0*individual[num].number[1],2.0)*(18.0-32.0*individual[num].number[0]+12.0*pow(individual[num].number[0],2.0)+48.0*individual[num].number[1]-36.0*individual[num].number[0]*individual[num].number[1]+27.0*pow(individual[num].number[1],2.0));
    individual[num].fitness=s*h;
}

```

//主程序

```
main()
```

```

{
    int i,j,k,t,total=0;
    float sum=0;
    for(j=0;j<50;j++)
    {
        initiate();
        for(i=0;i<500;i++)
        {
            W=1.0-i*0.6/499;

```

```

if((PBEST-3.0)<0.0001)
{
    total++;
    sum=sum+i;
    break;
}
else
{
    for(k=0;k<POPSIZE;k++)
    {
        for(t=0;t<DIMENSION;t++)
        {
            individual[k].speed[t]=W*individual[k].speed[t]+C1*rdf()*
(individual[k].best[t]-individual[k].number[t])+C2*rdf()*
(P[t]-individual[k].number[t]);
            if(individual[k].speed[t]>VMAX)
                individual[k].speed[t]=VMAX;
            individual[k].number[t]=individual[k].number[t]+individual[k].speed[t];
            if(individual[k].number[t]<XMIN)
                individual[k].number[t]=2*XMIN-individual[k].number[t];
            if(individual[k].number[t]>XMAX)
                individual[k].number[t]=2*XMAX-individual[k].number[t];
        }
        calculation(k);
        localbest(k);
    }
    globalbest(1);
}
}
printf("%d,%f\t",i,PBEST);
}
printf("Total number is %d\n",total);
printf("Sum is %f\n",sum);
printf("Average is %f\n",sum/total);
}

```