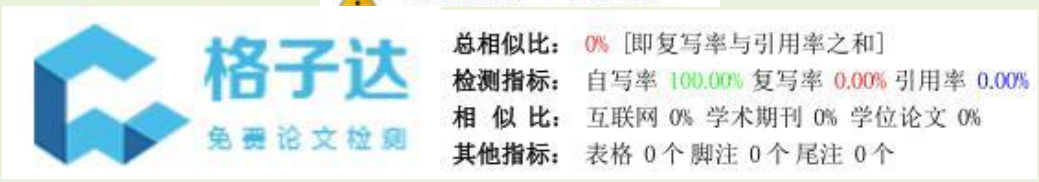
查重（算法部分+程序部分）



改进的蝙蝠算法 1

此算法通过模拟蝙蝠种群利用产生的声波搜索猎物和飞行方向来实现函数的寻优。以一只蝙蝠作为基本单元，且每只蝙蝠都有一个适应值来对函数解空间进行优化。每只蝙蝠可以调整自身发射声波的响度、频率等对空间进行搜索，使整个种群的活动逐步由无序变为有序。但蝙蝠算法在寻优末段容易陷入局部的极值，本文引入变速度权重因子修正系数，使蝙蝠的前期搜索为后期的搜索提供经验，尽可能避免局部极值的困境，从而最后求得全局最优解。经现有研究证明蝙蝠算法在函数极值的求解上优于的遗传、退火和粒子群等函数寻优的优化算法。此外，此算法具有收敛速度快，鲁棒性强等优点。本文引入变速度权重因子的修正系数，使蝙蝠的前期搜索为后期的搜索提供经验，尽可能避免局部极值的困境，从而最后求出全局最优解。

详细步骤如下：

Step1：初始化相关参数

蝙蝠 i 的位置为 X_i，飞行速度 V_i，声音响度为 A_i，频率 y_i 范围，设有目标函数为 $F(x_i)\ (i=1,2,...,n)$。

Step2：更改脉冲频率产生的解并更变蝙蝠位置与飞行速度

蝙蝠 i 在 $t-1$ 时的位置和飞行速度分别表示为 X_i^{t-1} 和 V_i^{t-1}，群体当前找到的最优位置为 X^*。接着根据自身发出不同的音响搜寻猎物，通过接受反馈信息来调整位置 x_i 和飞行速度 $v(i)$。其飞行的速度更变公式如下：

$$w(t)=w_{\min}+(w_{\max}-w_{\min})\exp(-\rho(\frac{t}{T_{\max}})^2) \quad (1)$$

$$y(i)=y(i)_{\min}+[y(i)_{\max}-y(i)_{\min}]\beta \quad (2)$$

$$V_i^t=w(t)V_i^{t-1}+A_i(X_i^{t-1}-X^*)y(i) \quad (3)$$

其中 $w(t)$ 为时刻变速惯性权重因子，作用是使蝙蝠的前期搜索对后期搜索提供参照，$w_{\max}$ 为 $w(t)$ 的最大值、$w_{\min}$ 为 $w(t)$ 的最小值；$1\le\rho\le T_{\max}$，一般取 2，$T_{\max}$ 为最大迭代次数；X^* 为当前位置最优解；$y(i)$ 为频率满足正态均匀分布的一个随机数，β 是一个随机变量，且 $\beta\in[0,1]$。开始运行时，蝙蝠在 $[y_{\min},y_{\max}]$ 随机进行频率分配。

$$X_i^t=X^*+V_i^t \quad (4)$$

为控制蝙蝠所处位置在自变量范围内，本文针对该情况设置了边界规则：如果下次运动的位置超出了自变量范围，那么下次飞行的位置为投影在的边界上的位置。

Step3:搜寻局部最优解 $F(X_i')$

Step4:通过蝙蝠多次飞行产生多个新解，进行全局搜索，若得到的新解 $F'(X_i') > F(X_i')$，那么接受该解

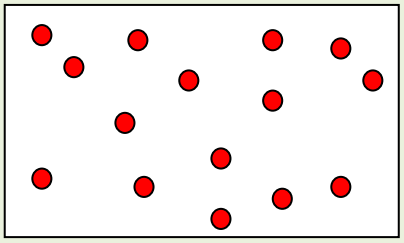
Step5:排列所有蝙蝠的位置，并找出当前最优值 $F'(X_i')$ 及对应的位置

Step6:设当前最优解为 F^*，然后使所有蝙蝠继续向下一时刻运动，并返回步骤 2 重新计算。

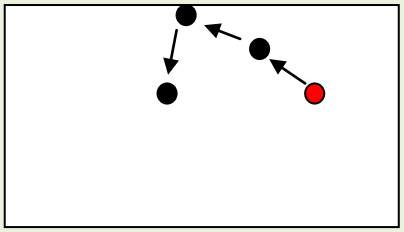
Step7:时刻结束，输出全局最优解

蝙蝠算法原理分析：

首先在自变量范围内产生蝙蝠的随机位置；



然后每个蝙蝠向周围某处移动，飞行靠声波反馈来更变方向，又向周围移动是随机的，蝙蝠下一时刻移动到某一位置有一定的出现频率，所以本文在运动公式中加了声波和频率两个因素。每一时刻的位移看作是一次飞行，每次飞行的距离很短（距离长短反映出搜素精度）。



每只蝙蝠初始位置是随机的，在初始位置时刻其中一只蝙蝠对应的函数最值，算法开始会记录该位置，然后所有蝙蝠逐渐向该位置靠近，飞行方向大致向当前最值方向（即该方向的位置该蝙蝠的出现频率更高），但是在飞行方向也是随机飞行的，相当于逐步搜索过去。蝙蝠飞行过程中不会因为当前飞行出现的最大值位置而改变种群的飞行趋势，这样可以尽量避免算法陷入局部极值。

算法每飞行一次就记录一次种群所处位置的最值，最后找出记录中的最值。

算法有两个参数可以影响最终的结果：种群数量和飞行次数。其中种群数量的影响是最大的。

案例

以一个二元非线性函数为例：

$$z = 21.5 + u_1 \sin(4\pi u_1) + u_2 \sin(20\pi u_2)$$

定义 u_1 自变量范围【-3，12.1】，u_2 自变量【4.1，5.8 范围】，求最大值

算法过程见**详细步骤**，程序见下页

标红部分为可更改部分，其余程序固定就好

改进的蝙蝠算法	运行环境：matlab2011a 以上

```
clear
wmax=0.9;%惯性权重最大值
wmin=0.4;%惯性权重最小值
n=10000; %  群体大小
A=rand(1,n); %  声音响度  (不变或者减小)
%  频率范围
Qmin=0; %  最低频率
Qmax=1; %  最高频率
d=2;%  搜索变量的维数（即频率和速度）
%  初始矩阵
Q=zeros(n,1); %  频率矩阵初始化
v=zeros(n,d); %  速度矩阵初始化，初始化意义就是产生一个初始矩阵
% x 自变量范围
u=-3;
o=12.1;
% y 自变量范围
p=4.1;
l=5.8;
%  初始化群体/解
for i=1:n
Sol(i,1)=-3+(12.1+3)*rand(1,1);%x 自变量范围【-3，12.1】
Sol(i,2)=4.1+(5.8-4.1)*rand(1,1);%y 自变量【4.1，5.8 范围】
%将随机生成的两个自变量带入函数式
Fitness(i)=Fun(Sol(i,:));%函数值
end
%  寻找当前最优解
[fmax,I]=max(Fitness);
best=Sol(I,:);
T=100;%飞行次数
%  开始飞行
for t=1:T
    for i=1:n,
        Q(i)=Qmin+(Qmin-Qmax)*rand;%rand 均匀分布的随机数
        %v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);（原速度）
        w=(wmax-wmin)*exp(-2*(t/T)^2)+wmin;%惯性权重因子
        v(i,:)=w*v(i,:)+(Sol(i,:)-best)*A(i)*Q(i);%更改后的速度
        S(i,:)=Sol(i,:)+v(i,:);%位置移动
        %边界问题，如果下次飞行超出自变量范围外了，那么下次飞行的位置为投影在的边界上的位置
        %x 轴
        if S(i,1)>o
            S(i,1)=o;
```

```
            end
            if S(i,1)<u
                S(i,1)=u;
            end
            %y 轴
            if S(i,2)>l
                S(i,2)=l;
            end
            if S(i,2)<p
                S(i,2)=p;
            end
            %评估该次飞行后产生的新解
            Fnew(i)=Fun(S(i,:));
        end
        [Fmax,Z]=max(Fnew);%找出该次飞行后产生的最大值
        C(t,:)=S(Z,:);
        FFnew(t)=Fmax;
end
[Ffmax,N]=max(FFnew);%找出整个飞行过程中的最大值
M=C(N,:)
Ffmax
```

```
%目标函数
function z=Fun(u)
    z=21.5+u(1)*sin(4*pi*u(1))+u(2)*sin(20*pi*u(2));
```

程序详解

n=10000; % 群体大小和 T=100;%飞行次数是提高算法结果准确的关键参数，两个参数越大，最后得到的结果越接近实际值，对于三维的函数飞行次数最好 50 次以上，种群规模这就要看自变量范围是多少了，matlab 中 rand 函数一般产生的随机数精度为 0.0001，本文小编设置的是 10000 只，那么对于这个函数的运算次数是一百万次，但是运行时间仅十多秒。蒙特卡洛求函数最优中也提到当随机计算一百万次的时候，得到的结果基本接近实际的极值了。所以在设置这两个参数时，最好两个参数相乘在一百万次左右。

% 初始矩阵

Q=zeros(n,1); % 频率矩阵初始化

v=zeros(n,d); % 速度矩阵初始化，初始化意义就是产生一个初始矩阵

初始化矩阵是为了方便后面的矩阵赋值，初始化矩阵也不一定要用到 zeros，ones 也可以的，但是尽量不要用=[]空矩阵，可能会发生错误。

% x 自变量范围

u=-3;

o=12.1;

% y 自变量范围

p=4.1;

```
l=5.8;
```
这里的自变量范围是为了后面的边界条件设置的
```
Sol(i,1)=-3+(12.1+3)*rand(1,1);%x 自变量范围【-3，12.1】
Sol(i,2)=4.1+(5.8-4.1)*rand(1,1);%y 自变量【4.1，5.8 范围】
```
这里的自变量范围才是要用于函数计算的，其他自变量设置参考上述案例
```
% 初始化群体/解
for i=1:n
    Sol(i,1)=-3+(12.1+3)*rand(1,1);%x 自变量范围【-3，12.1】
    Sol(i,2)=4.1+(5.8-4.1)*rand(1,1);%y 自变量【4.1，5.8 范围】
    %将随机生成的两个自变量带入函数式
    Fitness(i)=Fun(Sol(i,:));%函数值
end
```
这个循环是对每个蝙蝠依次随机安排一个初始位置，然后产生初始解
```
[fmax,I]=max(Fitness);
```
max 这个形式中的 I 记录的是最大值对应的自变量，fmax 为当前蝙蝠所处位置的函数值 Fitness 矩阵中的最大值，后面的[1,2]=max(3)形式均同理
```
Q(i)=Qmin+(Qmin-Qmax)*rand;%rand 均匀分布的随机数
%v(i,:)=v(i,:)+(Sol(i,:)-best)*Q(i);（原速度）
w=(wmax-wmin)*exp(-2*(t/T)^2)+wmin;%惯性权重因子
v(i,:)=w*v(i,:)+(Sol(i,:)-best)*A(i)*Q(i);%更改后的速度
S(i,:)=Sol(i,:)+v(i,:);%位置移动
```
上四行程序对应的是公式（1-4）
```
%x 轴
if S(i,1)>o
    S(i,1)=o;
end
if S(i,1)<u
    S(i,1)=u;
end
%y 轴
if S(i,2)>l
    S(i,2)=l;
end
if S(i,2)<p
    S(i,2)=p;
end
```
这是边界条件，下次飞行可能超出自变量范围，因此设立边界值避免得到自变量范围外的函数值

该案例求得的是最大值，如果是求最小值呢？

将程序中的 max 全部换成 min 就可以了（已用黄底纹标出）

程序更改

如果是其他函数怎么办呢？

函数 z=21.5+u(1)*sin(4*pi*u(1))+u(2)*sin(20*pi*u(2)) 中的两个自变量对应程序中的是 Sol(i,1)=-3+(12.1+3)*rand(1,1);%x 自变量范围【-3，12.1】和 Sol(i,2)=4.1+(5.8-4.1)*rand(1,1);%y 自变量【4.1，5.8 范围】，两自变量产生的是列矩阵，而程序中 Sol(i,:) 提取的是矩阵中的行，所以**如果对该函数增减自变量，或者想求其他含有多个自变量的函数，程序中只用修改以下程序部分**，其他参数也可以自行更改（注：自变量的范围和个数增加了，那么种群个数和飞行次数务必要增加）：

```
Sol(i,1)=-3+(12.1+3)*rand(1,1);%x 自变量范围【-3，12.1】
Sol(i,2)=4.1+(5.8-4.1)*rand(1,1);%y 自变量【4.1，5.8 范围】
```

和

```
% x 自变量范围
u=-3;
o=12.1;
% y 自变量范围
p=4.1;
l=5.8;
```