粒子群算法及其 matlab 实现

PSO 算法相关知识

1、 PSO 算法
   a) 冲突避免
      群体在一定空间移动，个体有自己的移动意志，但不能影响其他个体移动，避免碰撞与争执。
   b) 速度配合
      个体必须配合中心移动速度，不管在方向、距离与速率上都必须配合。
   c) 群体中心
      个体将会向群体中心移动，配合群体中心向目标前进。
2、 PSO 算法基本理论
3、 PSO 算法的约束优化
4、 PSO 算法的优缺点
   a) 不依赖于问题信息，采用实数求解，算法通用性强
   b) 需要调整的参数少，原理简单，易于求解
   c) 协同搜索，同时利用个体局部信息和全局信息知道搜索
   d) 收敛速度快，算法对 CPU 和内存要求不高
   e) 更容易飞跃局部最优信息（局部最优解）
5、 PSO 算法的缺点:
   a) 算法局部搜索能力不高，搜索精度差
   b) 算法不能绝对保证搜索到全局最优解

PSO 算法步骤

1、 初始化粒子群（速度和位置）、惯性因子、加速常数、最大迭代次数和算法终止的最小允许误差
2、 评价每个粒子的初始适应值
3、 将初始适应值作为当前每个粒子的局部最优值，并将各适应值对应的位置作为每个粒子的局部最优值所在的位置。
4、 将最佳初始适应值作为当前全局最优值，并将最佳适应值对应的位置作为全局最优值所在的位置。
5、 依据下式更新每个粒子当前的飞行速度

$$v_i^d = \omega v_i^d + c_1 r_1 (p_i^d - x_i^d) + c_2 r_2 (p_g^d - x_i^d)$$

其中，pdi 为这个粒子局部最优值的位置，pdg 为全局最优值位置，c1 为个体学习常量（加速常量），c2 为群体学习常量，r1，r2 为[0,1]区间上的随机数，w 为惯性因子
6、 对每个粒子的飞行速度进行限幅处理

7、 根据下式更新每个粒子当前所在位置

$$x_i^d = x_i^d + \alpha v_i^d$$

注意：α为约束因子

8、 比较当前每个粒子的适应值是否比历史局部最优值好，如果好，则将当前粒子适应值作为每个粒子的局部最优值所在位置

9、 在当前群众找出全局最优值，并将当前全局最优值对应的位置作为粒子群的全局最优值所在的位置。

10、 重复步骤 5~9，直到满足设定的最小误差或者达到最大迭代次数

11、 输出粒子群全局最优值和其对应的位置以及每个粒子的局部最优值和其对应的位置。

PSO 算法实例

$$\max f(x) = 2.1(1 - x + 2x^2)\exp\left(-\frac{x^2}{2}\right), \qquad x \in [-5,5]$$

根据 PSO 算法思路，编写的求解程序见 P8－1，计算机输出的对应图形如图 8－6 所示。

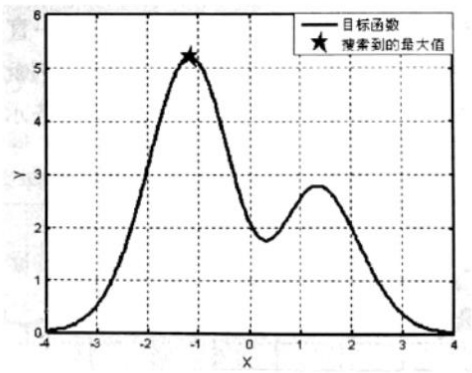

图 8－6 目标函数 PSO 算法搜索到的最大值位置

matlab 程序

```
clc;clear all;close all;
E0=1E-6;
MaxNum=200;%粒子迭代最大数
narvs=1;%自变量个数(一维运动)
particlesize=30;%粒子群规模
c1=1;%个体学习因子
c2=2;%团体学习因子
w=0.6;%惯性因子
a=1;%约束因子
vmax=0.8;%最大速度
x=-5+10*rand(particlesize,narvs);
v=2*rand(particlesize,narvs);
```

```
%评价每个粒子的初始适应度
first_fitness=fx(x);
%初始值作为局部最优解
every_fitness0=first_fitness;
%最大的适应度作为全局最优解
best_sol0=max(first_fitness);


%为个体最好位置和全局最好位置分配内存
personal_best=zeros(particlesize,MaxNum);
all_best=zeros(1,MaxNum);
personal_best(:,1)=every_fitness0;
all_best(:,1)=best_sol0;
for i=2:1:MaxNum
    %搜寻粒子最好位置和全局最好位置
    p_best=zeros(1,MaxNum);%分配内存
    for j=1:i
        for k=1:particlesize
            p_best(k)=max(personal_best(k,:));
        end
        a_best=max(all_best);
    end

    %更新速度
    for j=1:particlesize
        r1=unifrnd(0,1);
        r2=unifrnd(0,1);
        v(j)=w.*v(j)+(c1*r1).*(p_best(j)-x(j))+(c2*r2).*(a_best-x(j));
        if v(j)>vmax
            %不超过最大速度
            v(j)=vmax;
        elseif v(j)<-vmax
            v(j)=-vmax;
        end

    end
    %更新位置
    x=x+a.*v;
    z_ind=find(x>5);
    f_ind=find(x<-5);
    %位置约束
    for j=1:size(z_ind)
        x(z_ind(j))=5;
    end
    for j=1:size(f_ind)
```

```
        x(f_ind(j))=-5;
    end

    personal_best(:,i)=fx(x);
    all_best(:,i)=max(fx(x));

    if abs(a_best+1.1617)<E0
        %满足误差条件，退出
        break;
    end
end
%输出结果：
%画函数图
syms x1;
fx1=2.1*(1-x1+2*x1^2)*exp(-x1^2/2);
fxx=@(x)2.1*(1-x+2*x^2)*exp(-x^2/2);
fplot(fxx,[-5,5]);
%标准值
ff=@(x)2.1*(1-x+2*x^2)*exp(-x^2/2)*(-1);
[x_st,fval_st]=fminsearch(ff,-1);
hold on;
%画标准点
plot(x_st,-fval_st,'r-o');


disp('求解的最好点');
disp(a_best);
xbest=vpasolve(fx1==a_best,x1,-1);
%disp(xbest);
disp('标准的最好点');
disp(x_st);
disp('标准的点对应的最好值');
disp(-fval_st);
%画求解的点
hold on;
plot(xbest,a_best,'b--+');
```

```
function [fval]=fx(x)
%此处应注意.^,.*这些符号的应用
    fval=2.1*(1-x+2*x.^2).*exp(-x.^2/2);
end
```

问题的解

求解的最好点

5.1957

标准的最好点

-1.1617

标准的点对应的最好值

5.1985