**重力搜尋演算法(Gravitational Search Algorithm, GSA)**

重力搜尋演算法是一種可以用來解決最佳化問題的演演算法，將問題等化成物理上的重力，將所有粒子當作有品質的物體，每個粒子會受到解空間中其它粒子的萬有引力的影響，並產生加速度向品質更大的粒子運動，品質小的粒子在朝品質大趨近的過程中逐漸逼近優化問題中的最佳解。

**實驗步驟**

1. 產生原始族群:隨機產生搜尋範圍內的原始族群所在位置，這裡的原始族群即為物體質量。
2. 計算適應度:代入testbench function 求出各質量適應的程度
3. 求出各質量所佔比例:使用式(1)求出個別質量，再利用式(2)求出每個質量所佔的比例。若是要找出最大質量，則Best為適應度最大的，worst則為適應度最小的。

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

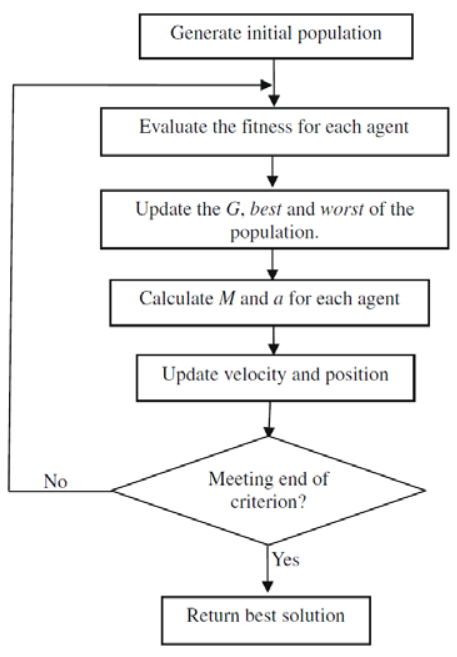
1. 算出吸引力:將彼此間的質量與距離代入改良式重力公式，如式(3)，即可求得彼此間的吸引力。 其中，G(t)值大小會隨時間改變，變化方式如式(4)。F 取最好的值數量也會隨著時間減 少，如式(5)。Ma為影響別人的強度(Active gravitational mass)，Mp為被影響的強度(Passive gravitational mass)，而Mi為自己的質量，Mii為第i個族群的慣性質量，本實驗將這些質量都設定為自己的質量，即Mi = Mp = Ma = Mii。其中R為兩個族群間的歐幾里得距離，Ԑ=8.85x10-12。

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |
|  | (5) |

1. 算出加速度並更新速度及位置:由式(6)即可算出加速度，並藉由式(7)及式(8)分別更新速度及位置。

|  |  |
| --- | --- |
|  | (6) |
|  | (7) |
|  | (8) |

1. 重力搜尋演算法流程圖

****

**實驗結果**

**Problem 1**

1. F2



1. F6



1. F9



1. F11



1. F13



1. F15



1. F17



**Problem 2**

1. F2



1. F6



1. F9



1. F11



1. F13



1. F15



1. F17



**Problem 3**

1. F2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 1.527031e-10 | 3.896542e-09 | 2.848142e-09 | 3.730713 |
| 2500 | 5.520515e-11 | 3.155687e-09 | 2.167612e-09 | 22.057448 |

1. F6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 0 | 0 | 0 | 3.612199 |
| 2500 | 0 | 0 | 0 | 46.728425 |

1. F9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 4.402978e-02 | 3.183869e-01 | 3.552714e-15 | 3.717849 |
| 2500 | 5.859624e-03 | 2.188910e-01 | 0 | 50.409793 |

1. F11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 3.819355e-02 | 5.330402e-02 | 5.917818e-02 | 4.081675 |
| 2500 | 3.366389e-03 | 1.780105e-02 | 1.971949e-02 | 56.996778 |

1. F13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 9.055836e-21 | 4.033246e-17 | 5.470016e-18 | 5.871517 |
| 2500 | 4.520691e-07 | 4.394946e-04 | 2.013140e-18 | 84.581480 |

1. F15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 3.987150e-03 | 2.614802e+02 | 4.968678e-03 | 5.161207 |
| 2500 | 1.947248e-03 | 9.096636e+02 | 1.439662e-03 | 57.221885 |

1. F17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 3.978874e-01 | 3.978874e-01 | 3.978874e-01 | 3.676623 |
| 2500 | 3.978874e-01 | 3.978874e-01 | 3.978874e-01 | 46.735922 |

**Problem 4**

1. F2

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 3.896542e-09 | 4.360152e-10 |
| 2500 | 3.155687e-09 | 3.626069e-10 |

1. F6

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 0 | 0 |
| 2500 | 0 | 0 |

1. F9

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 3.183869e-01 | 1.259798e-15 |
| 2500 | 2.188910e-01 | 6.542746e-16 |

1. F11

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 5.330402e-02 | 1.699675e-17 |
| 2500 | 1.780105e-02 | 9.887893e-18 |

1. F13

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 4.033246e-17 | 2.311773e-17 |
| 2500 | 4.394946e-04 | 1.024613e-17 |

1. F15

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 2.614802e+02 | 1.646270e+03 |
| 2500 | 9.096636e+02 | 2.237296e+03 |

1. F17

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | 3.978874e-01 | 1.384944e-16 |
| 2500 | 3.978874e-01 | 6.442496e-17 |

**Problem 5**

吸引力大小取值順序:1為從最小的開始取，0為從最大的開始取。min\_flag=1

重力公式參:alpha = 20,G0 = 100

最後一次迭代agent參與的比例: final\_per = 2

**GSA程式碼**

**main.m**

clear;clc;close all;

format long e;

% benchmark function select

fid1 = fopen('problem1.txt','w');

% fid2 = fopen('problem2.txt','w');

fid3 = fopen('problem3.txt','w');

fid4 = fopen('problem4.txt','w');

% fid5 = fopen('problem5.txt','w');

for i = 1:7

func\_index = [2 6 9 11 13 15 17] ; %testbench function編號

index\_F = func\_index(i);

tic

%可調參數

% index\_F=12; %testbench function編號

max\_iter=2500; %迭代次數

run = 50;

N=50;%agent數量

min\_flag=1; % 1: minimization, 0: maximization

% alpha = 20;G0 = 100;%p41重力公式參數

final\_per = 2;%最後一次迭代agent參與的比例

for runn = 1:run

[last\_fitness(runn,:),Fbest,Lbest,BestData(runn,:),MeanData(runn,:),MedianData(runn,:)]=GSA(index\_F,N,max\_iter,min\_flag,final\_per);

end

toc

time = toc;%存在problem 2

last\_fitness = mean(last\_fitness,1);

%將資料存入檔案中

%problem 1

% p1 = mean(BestData,1);

p1 = BestData(1,:);

figure();

plot(1:1:max\_iter,p1);

title(['function-',num2str(index\_F),'-searching results of one of the 50 runs,iteration =',num2str(max\_iter)]);

% fprintf(fid1,'%d\t',index\_F);fprintf(fid1,'%d\t',max\_iter);

for kk = 1:max\_iter

fprintf(fid1,'%f\t',p1(kk));

end

fprintf(fid1,'\r\n');

%problem 2

p2 = mean(BestData,1);

% average\_best\_so\_far = p2(1,max\_iter);

figure();

plot(1:1:max\_iter,p2);

title(['function-',num2str(index\_F),'-average best-so-far solution,iteration =',num2str(max\_iter)]);

%problem 3

fprintf(fid3,'%d\t',index\_F);

fprintf(fid3,'%d\t',max\_iter);

fprintf(fid3,'%d\t',time);

average\_best\_so\_far = p2(1,max\_iter);

average\_mean\_fitness = mean(last\_fitness);

median\_best\_so\_far = MedianData(1,max\_iter);

% time

fprintf(fid3,'%e\t',average\_best\_so\_far);

fprintf(fid3,'%e\t',average\_mean\_fitness);

fprintf(fid3,'%e\t',median\_best\_so\_far);

fprintf(fid3,'\r\n');

%problem 4

fprintf(fid4,'%d\t',index\_F);

fprintf(fid4,'%d\t',max\_iter);

average = average\_mean\_fitness;

stddd = std(last\_fitness);

fprintf(fid4,'%e\t',average);

fprintf(fid4,'%e\t',stddd);

fprintf(fid4,'\r\n');

end

fclose(fid1);

% fclose(fid2);

fclose(fid3);

fclose(fid4);

% fclose(fid5);

**GSA.m**

function [last\_fitness,Fbest,Lbest,BestData,MeanData,MedianData]=GSA(index\_F,N,max\_iter,min\_flag,final\_per)

[down,up,dim]=test\_functions\_range\_modify(index\_F);

%初始化

X=rand(N,dim).\*(up-down)+down;

V=zeros(N,dim);

last\_fitness = zeros(1,N);

%存數據

BestData=[];

MeanData=[];

MedianData=[];

for iter=1:max\_iter

%檢查是否越界，越界則該X重新初始化!!!

i=1;

while i<=N

% for j = 1:dim

% X(i,j)

if any(X(i,:)>up) || any(X(i,:)< down)

X(i,:) = rand(1,dim).\*(up-down)+down;

end

% end

i=i+1;

end

%計算agent的fitness

for i=1:N

fitness(i)=test\_functions\_modify(X(i,:),index\_F,dim);

end

if i==N

last\_fitness = fitness;

end

if min\_flag==1

[best, best\_X]=min(fitness); %minimization.

else

[best, best\_X]=max(fitness); %maximization.

end

if iter==1%第一代

Fbest=best;Lbest=X(best\_X,:);

end

if min\_flag==1

if best<Fbest %minimization.

Fbest=best;Lbest=X(best\_X,:);

end

else

if best>Fbest %maximization

Fbest=best;Lbest=X(best\_X,:);

end

end

BestData=[BestData Fbest];

MeanData=[MeanData mean(fitness)];

MedianData =[MedianData median(fitness)];

%eq.14-20計算

[M]=massCal(fitness,min\_flag);

%參考投影片p41公式

G = Gconstant(iter,max\_iter);

%Calculation of accelaration in gravitational field. eq.7-10,21.

a=Gfield(M,X,G,iter,max\_iter,final\_per);

%Agent movement. eq.11-12

[X,V]=move(X,a,V);

end %iteration

**massCal.m**

function [M]=massCal(fit,min\_flag)

Fmax=max(fit); Fmin=min(fit); Fmean=mean(fit);

N = size(fit,2);

if Fmax==Fmin

M=ones(N,1);

else

if min\_flag==1 %for minimization eq. 19、20

best=Fmin;worst=Fmax;

else %for maximization

best=Fmax;worst=Fmin;

end

M=(fit-worst)./(best-worst); %eq.15,

end

M=M./sum(M); %eq. 16.

**Gconstant.m**

function G = Gconstant(iter,max\_iter)

%p41重力公式參數

alpha = 20;

G0 = 100;

G=G0\*exp(-alpha\*iter/max\_iter);

**Gfield.m**

function a=Gfield(M,X,G,iter,max\_iter,final\_per)

[N,dim]=size(X);

kbest=final\_per+(1-iter/max\_iter)\*(100-final\_per);

kbest=round(N\*kbest/100);%四捨五入

[~, index]=sort(M,'descend');

for i=1:N

F(i,:)=zeros(1,dim);

for ii=1:kbest

j=index(ii);

if j~=i

R=norm(X(i,:)-X(j,:),2); %eq.8

for k=1:dim

F(i,k)=F(i,k)+rand\*(M(j))\*((X(j,k)-X(i,k))/(R+eps));%eq.7 、9

end

end

end

end

a=F.\*G; %eq. 10

**move.m**

function [X,V]=move(X,a,V)

%movement.

[N,dim]=size(X);

V=rand(N,dim).\*V+a; %eq. 11.

X=X+V; %eq. 12.

**Problem1**

****

**Problem 2**

****

**Problem 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterations | average best-so-far | average mean fitness | median best-so-far | Time(s) |
| 500 | 6.177638e+01 | -1.031242 | -6.071559e-01 | 61.771699 |
| 2500 | -1.031587 | -8.083419e-01 | -1.031628 | 219.796982 |

**Problem 4**

|  |  |  |
| --- | --- | --- |
| iterations | mean | standard deviation |
| 500 | -6.071559e-01 | 8.547866e-02 |
| 2500 | -8.083419e-01 | 5.830381e-02 |

在GSA演算法中，會藉由質量間的吸引力來搜尋出最佳解，而最佳解會接近於質量較大的位置，當質量越大時，移動的速度就會越慢，搜索的速度就會越慢，類似適應性的學習速率，比較可以找到區域性的解。缺點是可以調整的參數相對其他演算法來說較少，因此在搜尋範圍上有所限制。