clear all; close all; clc;

dim = 3;

N = 200; %jumlah air

alfa=360;

G=9.8; %gravitasi

t = 1; %time constant

upbound = [0.625 1300 40];

lowbound = [0.2 1071 30];

iter = 200;

Rpower=1;

min\_flag=1;

minmax =-1;

Rnorm=2;

convergence\_curve=zeros(1,iter);

% Initialize population, position:

if size(upbound,2)==1

X=rand(N,dim).\*(upbound-lowbound)+lowbound;

end

if size(upbound,2)>1

for i=1:dim

high=upbound(i);

low=lowbound(i);

X(:,i)=rand(N,1).\*(high-low)+low;

end

end

Bestpos=zeros(1,dim);

Meanpos=zeros(1,dim);

FBest=zeros(1,dim);

LBest=zeros(1,dim);

Eo=zeros(N,dim);

V=zeros(N,dim);

M = zeros(N);

P = 0;

%% Main Program

while P<iter

for iteration = 1:iter

%% inisialisasi Search Agent dan Objective Function

[N,dim]=size(X);

for i=1:N

%%Agent that go out of the search space, are reinitialized randomly .

Tp=X(i,:)>upbound;

Tm=X(i,:)<lowbound;

X(i,:)=(X(i,:).\*(~(Tp+Tm)))+((rand(1,dim).\*(upbound-upbound)+lowbound).\*(Tp+Tm));

end

for i=1:N

%L is the location of agent number 'i'

L=X(i,:);

%calculation of objective function for agent number 'i'

fobj=@(X)(fobjco2egr(X));

fitness(i)=fobj(X(i,:));

end

if minmax==1

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

else

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

end

if iteration==1

Fbest=best;

Lbest=X(best\_X,:);

end

if minmax==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

Bestpos=[Bestpos Fbest];

Meanpos=[Meanpos mean(fitness)];

%% Hujan jatuh = energi potensial = Ep = 1/2 mv^2

% velocity calculation

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

[i N]=size(fitness);

if Fmax==Fmin

vo=ones(N,1);

else

if minmax==1 %for maximization

best=Fmin;worst=Fmax;

else %for minimation

best=Fmax;worst=Fmin;

end

vo=(fitness-worst)./(best-worst);

end

M= rand(N);

vo=(vo./sum(vo))\*M.\*t;

% velocity calculation berfungsi untuk menentukan butiran hujan yang jatuh

% terlebih dahulu berdasarkan fitness dari setiap agents.

%%

% [N,dim]=size(X);

final\_per=1.5; %In the last iteration, only 1.5 percent of agents

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

kbest=round(N\*kbest/100);

[Ms ds]=sort(vo,'descend');

for i=1:N

for ii=1:kbest

j=ds(ii);

if j~=i

R=norm(X(i,:)-X(j,:),Rnorm); %Euclidian distanse.

for k=1:dim

Eo(i,k)=Eo(i,k)+rand\*(vo(j))\*((X(j,k)-X(i,k))/(R^Rpower+eps));

end

end

end

end

%%acceleration

E = Eo\*exp(-alfa\*iteration/iter);

a=E.\*G;

%movement.

% [N,dim]=size(X);

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

X=X+V;

P = P + 1;

convergence\_curve(P) = Fbest;

jx=plot((1:iter),convergence\_curve,'LineWidth',2);grid on;

title(['Rainfall Algorithm Best Value : ' num2str(Fbest)]);

xlabel('Iteration');

ylabel('Function Value');

end

end