% clear all

% close all

% clc

%tg=Best\_pos

VarName=xlsread("ekim.xlsx")

VarName1=VarName(:,1);

VarName2=VarName(:,2);

VarName3=VarName(:,3);

VarName4=VarName(:,4);

VarName5=VarName(:,5);

VarName6=VarName(:,6);

nVar = 5;

ub = [1 1 1 1 50]\*2;

lb = [-1 -1 -1 -1 -50]\*2;

fobj = @energyPred;

%% GRO parameter initialization

sigma\_initial = 2;

sigma\_final = 1 / Max\_iter ;

% Initialize best position X\* (global best)

best\_pos=zeros(1, dim);

best\_score=inf; %change this to -inf for maximization problems

%Initialize the gold prospectors’ population Xi, i = 1, 2, . . . , N

Positions=initialization(N, dim, lb, ub);

Fit = inf(1,N);

%Initialize the gold prospectors’ new positions Xnewi = Xi , i = 1, 2, . . . , N

X\_NEW = Positions;

Fit\_NEW = Fit;

Convergence\_curve=zeros(1, Max\_iter);

Convergence\_curve(1) = min(Fit);

iter = 1;% Loop counter

%% Main loop

while iter <= Max\_iter

for i= 1:N

%Calculate fitness of current search agent at new position XNewi

Fit\_NEW(i) = fobj(X\_NEW(i,:));

%Update position of current search agent Xi according to Equation (13)

if Fit\_NEW(i) < Fit(i)

Fit(i) = Fit\_NEW(i);

Positions(i,:) = X\_NEW(i,:);

end

%Update best search agent X\*

if Fit(i) < best\_score

% new gold mine is found

best\_score = Fit(i);

best\_pos = Positions(i,:);

end

end

%Update l1, l2 by Equation (7)

l2 = ((Max\_iter - iter)/(Max\_iter-1) )^2 \* (sigma\_initial - sigma\_final) + sigma\_final;

l1 = ((Max\_iter - iter)/(Max\_iter-1) )^1 \* (sigma\_initial - sigma\_final) + sigma\_final;

%calculate the next position of current search agent XNewi with one of

%... the migration, mining or collaboration methods

for i = 1:size(Positions,1)

coworkers = randperm(N-1,2);

diggers = 1:N;

diggers(i) = [];

coworkers = diggers(coworkers);

digger1 = coworkers(1); %random prospector g1

digger2 = coworkers(2); %random prospector g2

m = rand;

%collaboration

if m < 1/3

for d = 1:dim

r1 = rand; % r1 is a random number in [0,1]

D3 = Positions(digger2,d) - Positions(digger1,d); % Equation (11)

X\_NEW(i,d) = Positions(i,d) + r1 \* D3; % Equation (12)

end

%mining method

elseif m < 2/3

for d = 1:dim

r1 = rand; % r1 is a random number in [0,1]

A2 = 2\*l2\*r1 - l2 ; % Equation (10)

D2 = Positions(i,d) - Positions(digger1,d) ; % Equation (8)

X\_NEW(i,d) = Positions(digger1,d) + A2\*D2; % Equation (9)

end

%migartion method

else

for d = 1:dim

r1 = rand; % r1 is a random number in [0,1]

r2 = rand; % r2 is a random number in [0,1]

C1 = 2 \* r2; % Equation (6)

A1 = 1 + l1 \* (r1 - 1/2); % Equation (5)

D1 = C1 \* best\_pos(d) - Positions(i,d) ; % Equation (3)

X\_NEW(i,d) = Positions(i,d) + A1 \* D1; % Equation (4)

end

end

%Domain control

X\_NEW(i,:) = boundConstraint(X\_NEW(i,:),Positions(i,:), lb , ub);

end

Convergence\_curve(iter) = best\_score;

iter = iter+1;

end

end