%Killer Whale Optimization Algorithm

clc;

clear;

%% Problem Definition

Dimension = 3; % dimensi diganti sesuai dengan jumlah variabel yang dioptimasi

%Constraint

load ('propco2egr.mat')

CostFunction=@(x) (1/fobjco2egr(x)); % Cost Function

nVar=Dimension; % Number of Decision Variables

VarSize=[1 nVar]; % Size of Decision Variables Matrix

VarMin=LB; % Lower Bound of Variables

VarMax=UB; % Upper Bound of Variables

%% PSO Parameters

MaxIt=50; % Maximum Number of Iterations

nPop=200; % Population Size (Swarm Size)

nTeam = 20; % Team (Number of Leader)

TeamSize = [];

for i=1:nTeam-1

TeamSize(i) = ceil(nPop/nTeam);

end

TeamSize(nTeam) = nPop - sum(TeamSize);

% PSO Parameters

w=1; % Inertia Weight

wdamp=0.99; % Inertia Weight Damping Ratio

c1=1.5; % Personal Learning Coefficient

c2=2; % Global Learning Coefficient

c3=1.0; % Leader Influence Coefficient

Porder=3; % order of Polynomial

% If you would like to use Constriction Coefficients for PSO,

% uncomment the following block and comment the above set of parameters.

% % Constriction Coefficients

% phi1=2.05;

% phi2=2.05;

% phi=phi1+phi2;

% chi=2/(phi-2+sqrt(phi^2-4\*phi));

% w=chi; % Inertia Weight

% wdamp=1; % Inertia Weight Damping Ratio

% c1=chi\*phi1; % Personal Learning Coefficient

% c2=chi\*phi2; % Global Learning Coefficient

% Velocity Limits

VelMax=0.1\*(VarMax-VarMin);

VelMin=-VelMax;

%% Initialization

initial\_whales.Position=[];

initial\_whales.Cost=[];

initial\_whales.Velocity=[];

initial\_whales.Best.Position=[];

initial\_whales.Best.Cost=[];

whales=repmat(initial\_whales,nPop,1);

leader\_whales=repmat(initial\_whales,nTeam,1);

leader\_whales\_poly = [];

leader\_whales\_std = [];

temp\_whales = [];

tempeval\_whales = [];

GlobalBest.Cost=inf;

for i=1:nPop

% Initialize Position

whales(i).Position=unifrnd(VarMin,VarMax,VarSize);

% Initialize Velocity

whales(i).Velocity=zeros(VarSize);

% Evaluation

whales(i).Cost=CostFunction(whales(i).Position);

% temp\_whales(:, i) = whales(i).Position;

end

% tempeval\_whales(1:nPop) = whales.Cost(1:nPop);

for i=1:nPop

% whales(i).Cost = tempeval\_whales(i);

% Store data for polyfit

for j=1:Dimension

whalesPosition(i,j) = whales(i).Position(j);

end

whalesPosition(i,Dimension+1) = whales(i).Cost;

% Update Personal Best

whales(i).Best.Position=whales(i).Position;

whales(i).Best.Cost=whales(i).Cost;

% Update Global Best

if whales(i).Best.Cost<GlobalBest.Cost

GlobalBest=whales(i).Best;

end

end

temp\_whales = [];

tempeval\_whales = [];

for t=1:nTeam

for j=1:Dimension

leader\_whales\_std(t,j) = std2(whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),j));

buffer = polyfit(whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),j),whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),Dimension+1),Porder);

leader\_whales\_poly(t,:,j) = buffer;

syms x;

fun = matlabFunction(poly2sym(buffer));

leader\_whales(t).Position(j) = fminsearch((fun),0);

leader\_whales(t).Position(j) = min(leader\_whales(t).Position(j), UB(j));

leader\_whales(t).Position(j) = max(leader\_whales(t).Position(j), LB(j));

end

temp\_whales(:, t) = leader\_whales(t).Position;

end

% tempeval\_whales = CostFunction(temp\_whales);

for t=1:nTeam

% leader\_whales(t).Cost = tempeval\_whales(t);

leader\_whales(t).Cost = CostFunction(leader\_whales(t).Position);

end

BestCost=zeros(MaxIt,1);

%% PSO Main Loop

for it=1:MaxIt

for t=1:nTeam

temp\_whales = [];

tempeval\_whales = [];

for i=((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t)

% make into cluster

% each cluster will have it's own leader

% each member of cluters should chate their own leader

% leader get information from their member, and draw a polynomial

% equation to map the scanned area

% Update Velocity

% Leader or GlobalBest

if GlobalBest.Cost < leader\_whales(t).Cost % min

ct3 = 0;

ct2 = c2;

else

ct3 = c3;

ct2 = 0;

end;

whales(i).Velocity = w\*whales(i).Velocity ...

+c1\*rand(VarSize).\*(whales(i).Best.Position-whales(i).Position) ...

+ct2\*rand(VarSize).\*(GlobalBest.Position-whales(i).Position) ...

+ct3\*rand(VarSize).\*(leader\_whales(t).Position);

% Apply Velocity Limits

whales(i).Velocity = max(whales(i).Velocity,VelMin);

whales(i).Velocity = min(whales(i).Velocity,VelMax);

% Update Position

whales(i).Position = whales(i).Position + whales(i).Velocity;

% Velocity Mirror Effect

IsOutside=(whales(i).Position<VarMin | whales(i).Position>VarMax);

whales(i).Velocity(IsOutside)=-whales(i).Velocity(IsOutside);

% Apply Position Limits

whales(i).Position = max(whales(i).Position,VarMin);

whales(i).Position = min(whales(i).Position,VarMax);

temp\_whales(:, i) = whales(i).Position;

end

% tempeval\_whales = CostFunction(temp\_whales);

for i=((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t)

% Evaluation

whales(i).Cost = CostFunction(whales(i).Position);

% whales(i).Cost = tempeval\_whales(i);

% Store data for polyfit

for j=1:Dimension

whalesPosition(i,j) = whales(i).Position(j);

end

whalesPosition(i,Dimension+1) = whales(i).Cost;

% Update Personal Best

if whales(i).Cost<whales(i).Best.Cost

whales(i).Best.Position=whales(i).Position;

whales(i).Best.Cost=whales(i).Cost;

% Update Global Best

if whales(i).Best.Cost<GlobalBest.Cost

GlobalBest=whales(i).Best;

end

end

end

end

polycheck = 0;

leader\_whales\_poly = [];

temp\_whales = [];

tempeval\_whales = [];

for t=1:nTeam

for j=1:Dimension

leader\_whales\_std(t,j) = std2(whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),j));

if leader\_whales\_std < 0.01

polycheck = 1;

else

buffer = polyfit(whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),j),whalesPosition(((t-1)\*ceil(nPop/nTeam))+1:((t-1)\*ceil(nPop/nTeam))+TeamSize(t),Dimension+1),Porder);

leader\_whales\_poly(t,:,j) = buffer;

if max(isnan(buffer)) < 1

syms x;

fun = matlabFunction(poly2sym(buffer));

leader\_whales(t).Position(j) = fminsearch((fun),0);

end

leader\_whales(t).Position(j) = min(leader\_whales(t).Position(j), UB(j));

leader\_whales(t).Position(j) = max(leader\_whales(t).Position(j), LB(j));

end

end

% temp\_whales(:, t) = leader\_whales(t).Position;

end

% tempeval\_whales = CostFunction(temp\_whales);

for t=1:nTeam

% leader\_whales(t).Cost = tempeval\_whales(t);

leader\_whales(t).Cost = CostFunction(leader\_whales(t).Position);

end

BestCost(it)=1/GlobalBest.Cost;

disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it)) ': Datacheck = ' num2str(polycheck)]);

w=w\*wdamp;

end

BestSol = GlobalBest;

%% Results

% min\_variable\_design = BestSol.Position(1,:)

% min\_objective\_function = BestSol.Cost(1,:)

figure(gcf)

title('Grafik Nilai Maksimum KWA','color','b')

xlabel('Jumlah Iterasi')

ylabel('Nilai Fungsi Obyektif')

hold on

grid on

% plot(efitnessmax, 'DisplayName', 'efitnessmax', 'YDataSource', 'efitnessmax');

plot(BestCost,'LineWidth',2);

hold on

% semilogy(BestCost,'LineWidth',2);

% xlabel('Jumlah Iterasi');

% ylabel('Nilai Fungsi Obyektif');

% grid on;