clc;

clear;

close all;

%% Problem Definition

data = xlsread('infile');

X = data;

k = 2;

CostFunction=@(m) ClusteringCost(m, X); % Cost Function

VarSize=[k size(X,2)]; % Decision Variables Matrix Size

nVar=prod(VarSize); % Number of Decision Variables

VarMin= repmat(min(X),k,1); % Lower Bound of Variables

VarMax= repmat(max(X),k,1); % Upper Bound of Variables

%% PSO Parameters

MaxIt=200; % Maximum Number of Iterations

nPop=70; % Population Size (Swarm Size)

% w=1; % Inertia Weight

% wdamp=0.99; % Inertia Weight Damping Ratio

% c1=2; % Personal Learning Coefficient

% c2=2; % Global Learning Coefficient

% 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

empty\_particle.Position=[];

empty\_particle.Cost=[];

empty\_particle.Out=[];

empty\_particle.Velocity=[];

empty\_particle.Best.Position=[];

empty\_particle.Best.Cost=[];

empty\_particle.Best.Out=[];

particle=repmat(empty\_particle,nPop,1);

BestSol.Cost=inf;

for i=1:nPop

% Initialize Position

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

% Initialize Velocity

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

% Evaluation

[particle(i).Cost, particle(i).Out]=CostFunction(particle(i).Position);

% Update Personal Best

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

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

particle(i).Best.Out=particle(i).Out;

% Update Global Best

if particle(i).Best.Cost<BestSol.Cost

BestSol=particle(i).Best;

end

end

BestCost=zeros(MaxIt,1);

%% PSO Main Loop

for it=1:MaxIt

for i=1:nPop

% Update Velocity

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

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

+c2\*rand(VarSize).\*(BestSol.Position-particle(i).Position);

% Apply Velocity Limits

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

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

% Update Position

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

% Velocity Mirror Effect

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

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

% Apply Position Limits

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

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

% Evaluation

[particle(i).Cost, particle(i).Out] = CostFunction(particle(i).Position);

% Update Personal Best

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

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

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

particle(i).Best.Out=particle(i).Out;

% Update Global Best

if particle(i).Best.Cost<BestSol.Cost

BestSol=particle(i).Best;

end

end

end

BestCost(it)=BestSol.Cost;

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

% Plot Solution

figure(1);

PlotSolution(X, BestSol);

pause(0.01);

w=w\*wdamp;

end

silhouette(data,BestSol.Out.ind)

xlabel('Silhouette Value')

ylabel('Cluster')

s=silhouette(data,BestSol.Out.ind);

%% Results

figure;

plot(BestCost,'LineWidth',2);

xlabel('Iteration');

ylabel('Best Cost');