clear

clc

cof = [0.8739 0.8739 0.3261 0.3261...

0.5943 0.5943 0.4057 0.4057...

0.7213 0.7213 0.2787 0.2787 ...

0.8156 0.8156 0.1844 0.1844...

0.6539 0.6539 0.3461 0.3461];

a = [1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0];

b = [0 1 -1 1 0 1 -1 1 0 1 -1 1 0 1 -1 1 0 1 -1 1];

re = [17 17 0.7 0.7 3 3 0.6 0.6 4 4 0.6 0.6 2 2 0.5 0.5 4.5 4.5 0.7 0.7];

P = 18.139970195;

Lp = 0;

M = 21.46730000;

Lm = 0;

fun = @(x)((cof \* x));

cons1 = @(x,P)(a \* x <= P - Lp);

cons2 = @(x,M)(b \* x <= M - Lm);

cons3 = @(x)(x> 0);

%% Set population Parameters

sizepop = 2000; % Initial population number

dim = 20; % The spatial dimension

ger = 700; % Maximum iteration

% xlimit\_max = 25\*ones(dim,1); % Set position parameter limits

xlimit\_max = re';

xlimit\_min = zeros(dim,1);

vlimit\_max = 0.001\*ones(dim,1); % Set speed limit

vlimit\_min = zeros(dim,1);

c\_1 = 0.1; % Inertia weight

c\_2 = 0.5; % Self-learning factor

c\_3 = 0.5; % Group learning factor

%% Generating initial population

% First, the initial population position is generated randomly

% Then the initial population velocity is generated randomly

% It then initializes the individual historical best position, as well as the individual historical best fitness

% Then initialize the population history best position, and the population history best fitness

for i=1:dim

for j=1:sizepop

pop\_x(i,j) = xlimit\_min(i)+(xlimit\_max(i) - xlimit\_min(i))\*rand; % The location of the initial population

pop\_v(i,j) = vlimit\_min(i)+(vlimit\_max(i) - vlimit\_min(i))\*rand; % The velocity of the initial population

end

end

gbest = pop\_x; % The historical best position of each individual

for j=1:sizepop

if cons1(pop\_x(:,j),P)

if cons2(pop\_x(:,j),M)

if cons3(pop\_x(:,j))

fitness\_gbest(j) = fun(pop\_x(:,j)); % The historical best position of each individual

else

fitness\_gbest(j) = 0;

end

else

fitness\_gbest(j) = 0;

end

else

fitness\_gbest(j) = 0;

end

end

zbest = pop\_x(:,1); % The historical best location of the population

fitness\_zbest = fitness\_gbest(1); % Historical optimum fitness of the population

for j=1:sizepop

if fitness\_gbest(j) > fitness\_zbest

zbest = pop\_x(:,j);

fitness\_zbest=fitness\_gbest(j);

end

end

%% Particle swarm iteration

% Update speed and perform speed boundary processing

% Updates the position and bounds the position

% Adaptive mutation is performed

% To judge the constraints and calculate the fitness of each individual position in the new population

% The new fitness was compared with the individual's historical best fitness

% The best fitness in individual history was compared with the best fitness in population history

% Loop again or end

iter = 1;

record = zeros(ger, 1);

while iter <= ger

for j=1:sizepop

% Update speed and perform speed boundary processing

pop\_v(:,j)= c\_1 \* pop\_v(:,j) + c\_2\*rand\*(gbest(:,j)-pop\_x(:,j))+c\_3\*rand\*(zbest-pop\_x(:,j));

for i=1:dim

if pop\_v(i,j) > vlimit\_max(i)

pop\_v(i,j) = vlimit\_max(i);

end

if pop\_v(i,j) < vlimit\_min(i)

pop\_v(i,j) = vlimit\_min(i);

end

end

% Updates the position and bounds the position

pop\_x(:,j) = pop\_x(:,j) + pop\_v(:,j);

for i=1:dim

if pop\_x(i,j) > xlimit\_max(i)

pop\_x(i,j) = xlimit\_max(i);

end

if pop\_x(i,j) < xlimit\_min(i)

pop\_x(i,j) = xlimit\_min(i);

end

end

% Adaptive mutation is performed

if rand > 0.85

i=ceil(dim\*rand);

pop\_x(i,j)=xlimit\_min(i) + (xlimit\_max(i) - xlimit\_min(i)) \* rand;

end

% To judge the constraints and calculate the fitness of each individual position in the new population

if cons1(pop\_x(:,j),P)

if cons2(pop\_x(:,j),M)

if cons3(pop\_x(:,j))

fitness\_pop(j) = fun(pop\_x(:,j)); % Fitness of the current individual

else

fitness\_pop(j) = 0;

end

else

fitness\_pop(j) = 0;

end

else

fitness\_pop(j) = 0;

end

% The best fitness in individual history was compared with the best fitness in population history

if fitness\_pop(j) > fitness\_gbest(j)

gbest(:,j) = pop\_x(:,j); % Update group history best location

fitness\_gbest(j) = fitness\_pop(j); % Update population history for optimum fitness

end

% The best fitness in individual history was compared with the best fitness in population history

if fitness\_gbest(j) > fitness\_zbest

zbest = gbest(:,j); % Update group history best location

fitness\_zbest=fitness\_gbest(j); % Update population history for optimum fitness

end

end

record(iter) = fitness\_zbest;%Maximum record

iter = iter+1;

end

%% µü´ú½á¹ûÊä³ö

plot(record,'Linewidth',1.5);title('The convergence process')

ylabel('The objective function')

xlabel('The number of iterations')

disp(['The optimal value£º',num2str(fitness\_zbest)]);

disp('State variable£º');

disp(zbest);

for i = 1:10

c(i) = zbest(2\*i) + zbest(2\*i-1);

end

for i = 1:5

C(i,1) = zbest(4\*i-2) + zbest(4\*i-3);

C(i,2) = zbest(4\*i) + zbest(4\*i-1);

end

r = [17 0.7 3 0.6 4 0.6 2 0.5 4.5 0.7];

for i = 1:10

if c(i) <= r(i)

delter\_c(i) = r(i) - c(i);

Delter(i) = 0;

else

delter\_c(i) = 0;

Delter(i) = c(i) - r(i);

end

end

for i = 1:5

u(i,1) = zbest(4\*i-3);

u(i,2) = zbest(4\*i-1);

d(i,1) = zbest(4\*i-2);

d(i,2) = zbest(4\*i);

end

if P + M < 33

rem = sum(delter\_c);

Rem = sum(Delter);

else

rem = 33 -(P+M);

end

figure

bar((d+u))

hold on

bar((d))

bar([24,0.7;2,0.6;4,0.6;2,0.5;4,0.7],'FaceColor','none','EdgeColor','k','LineWidth',1.5)

legend('GU from Powell','EP from Powell','GU from Mead','EP from Mead','require for GU','require for EP');

set(gca, 'XTickLabel',{'CA','AZ','WY','NM','CO'})

set(gca,'ylim',[0,30],'ytick',[0:5:30]);

set(gca, 'Ygrid','on');

ylabel('Water allocation')

title('Water resources allocation scheme')