%% 清空环境

clc;clear all;close all;

random\_num=zeros(1,10)

t=[0,17,16,26,33,43,51,38;17,0,14,10,25,33,36,30;16,14,0,21,18,29,38,23;26,10,21,0,26,30,29,28;33,25,18,26,0,11,25,6;43,33,29,30,11,0,18,6;51,36,38,29,25,18,0,22;38,30,23,28,6,6,22,0] % 各岛屿间转移时间矩阵

for i=1:8

random\_num (i)= t(randperm(numel(t(i,:),1),i);

end

disp(random\_num);

a=1/60\*[-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,500,500,500,500,500,500,500,500,500,500,-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,500,500,500,500,500,500,500,500,500,500];

a1=a(1,11:20)

a2=a(1,1:10)

a3=a(1,21:30)

a4=a(1,31:40)

b=ones(40);

fun= @(X)(a\*X);

cons1= @(X)(300<=a\*X<=2700);

cons2= @(X)(0<b\*X<540);

%% 设置种群参数

sizepop = 100; % 初始种群个数

dim = 40; % 空间维数

ger = 500; % 最大迭代次数

xlimit\_max = 60\*ones(dim,1); % 设置位置参数限制(矩阵的形式可以多维)

xlimit\_min = 0\*ones(dim,1);

vlimit\_max = 1\*ones(dim,1); % 设置速度限制

vlimit\_min = -1\*ones(dim,1);

c\_1 = 0.8; % 惯性权重

c\_2 = 1.5; % 自我学习因子

c\_3 = 1.5; % 群体学习因子

%% 生成初始种群

% 首先随机生成初始种群位置

% 然后随机生成初始种群速度

% 然后初始化个体历史最佳位置，以及个体历史最佳适应度

% 然后初始化群体历史最佳位置，以及群体历史最佳适应度

for i=1:dim

for j=1:sizepop

pop\_x(i,j) = xlimit\_min(i)+(xlimit\_max(i) - xlimit\_min(i))\*rand; % 初始种群的位置

pop\_v(i,j) = vlimit\_min(i)+(vlimit\_max(i) - vlimit\_min(i))\*rand; % 初始种群的速度

end

end

gbest = pop\_x; % 每个个体的历史最佳位置

for j=1:sizepop

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

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

fitness\_gbest(j) = fun(pop\_x(:,j)); % 每个个体的历史最佳适应度

else

fitness\_gbest(j) = 10^(-10);

end

else

fitness\_gbest(j) = 10^(-10);

end

end

zbest = pop\_x(:,1); % 种群的历史最佳位置

fitness\_zbest = fitness\_gbest(1); % 种群的历史最佳适应度

for j=1:sizepop

if fitness\_gbest(j) > fitness\_zbest % 如果求最小值，则为<; 如果求最大值，则为>;

zbest = pop\_x(:,j);

fitness\_zbest=fitness\_gbest(j);

end

end

%% 粒子群迭代

% 更新速度并对速度进行边界处理

% 更新位置并对位置进行边界处理

% 进行自适应变异

% 进行约束条件判断并计算新种群各个个体位置的适应度

% 新适应度与个体历史最佳适应度做比较

% 个体历史最佳适应度与种群历史最佳适应度做比较

% 再次循环或结束

iter = 1; %迭代次数

record = zeros(ger, 1); % 记录器

while iter <= ger

for j=1:sizepop

% 更新速度并对速度进行边界处理

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

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

% 更新位置并对位置进行边界处理

pop\_x(:,j) = pop\_x(:,j) + pop\_v(:,j);% 位置更新

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

% 进行自适应变异

if rand > 0.85

i=ceil(dim\*rand);

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

end

% 进行约束条件判断并计算新种群各个个体位置的适应度

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

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

fitness\_pop(j) = fun(pop\_x(:,j)); % 当前个体的适应度

else

fitness\_pop(j) = 10^-(10);

end

else

fitness\_pop(j) = 10^(-10);

end

% 新适应度与个体历史最佳适应度做比较

if fitness\_pop(j) > fitness\_gbest(j) % 如果求最小值，则为<; 如果求最大值，则为>;

gbest(:,j) = pop\_x(:,j); % 更新个体历史最佳位置

fitness\_gbest(j) = fitness\_pop(j); % 更新个体历史最佳适应度

end

% 个体历史最佳适应度与种群历史最佳适应度做比较

if fitness\_gbest(j) > fitness\_zbest % 如果求最小值，则为<; 如果求最大值，则为>;

zbest = gbest(:,j); % 更新群体历史最佳位置

fitness\_zbest=fitness\_gbest(j); % 更新群体历史最佳适应度

end

end

record(iter) = fitness\_zbest;%最大值记录

iter = iter+1;

end

%% 迭代结果输出

plot(record);title('收敛过程')

disp(['最优值：',num2str(fitness\_zbest)]);

disp('变量取值：');

disp(zbest);

-60,-60；,-60,-60,-60,-60,-60,-60,-60,-60,500,500,500,500,500,500,500,500,500,500,-60,-60,-60,-60,-60,-60,-60,-60,-60,-60,500,500,500,500,500,500,500,500,500,500