Function [R\_best,L\_best,L\_ave,Shortest\_Route,Shortest\_Length]=ACATSP(C,NC\_max,m,Alpha,Beta,Rho,Q)

%%-------------------------------------------------------------------------

%% 主要符号说明

%% C n个城市的坐标，n×2的矩阵

%% NC\_max 最大迭代次数

%% m 蚂蚁个数

%% Alpha 表征信息素重要程度的参数

%% Beta 表征启发式因子重要程度的参数

%% Rho 信息素蒸发系数

%% Q 信息素增加强度系数

%% R\_best 各代最佳路线

%% L\_best 各代最佳路线的长度

%%=========================================================================

%%第一步：变量初始化

n=size(C,1);%n表示问题的规模（城市个数）

D=zeros(n,n);%D表示完全图的赋权邻接矩阵

for i=1:n

for j=1:n

if i~=j

D(i,j)=((C(i,1)-C(j,1))^2+(C(i,2)-C(j,2))^2)^0.5;

else

D(i,j)=eps;      %i=j时不计算，应该为0，但后面的启发因子要取倒数，用eps（浮点相对精度）表示

end

D(j,i)=D(i,j);   %对称矩阵

end

end

Eta=1./D;          %Eta为启发因子，这里设为距离的倒数

Tau=ones(n,n);     %Tau为信息素矩阵

Tabu=zeros(m,n);   %存储并记录路径的生成

NC=1;               %迭代计数器，记录迭代次数

R\_best=zeros(NC\_max,n);       %各代最佳路线

L\_best=inf.\*ones(NC\_max,1);   %各代最佳路线的长度

L\_ave=zeros(NC\_max,1);        %各代路线的平均长度

while NC<=NC\_max        %停止条件之一：达到最大迭代次数，停止

%%第二步：将m只蚂蚁放到n个城市上

Randpos=[];   %随即存取

for i=1:(ceil(m/n))

Randpos=[Randpos,randperm(n)];

end

Tabu(:,1)=(Randpos(1,1:m))';    %此句不太理解？

%%第三步：m只蚂蚁按概率函数选择下一座城市，完成各自的周游

for j=2:n     %所在城市不计算

for i=1:m

visited=Tabu(i,1:(j-1)); %记录已访问的城市，避免重复访问

J=zeros(1,(n-j+1));       %待访问的城市

P=J;                      %待访问城市的选择概率分布

Jc=1;

for k=1:n

if length(find(visited==k))==0   %开始时置0

J(Jc)=k;

Jc=Jc+1;                         %访问的城市个数自加1

end

end

%下面计算待选城市的概率分布

for k=1:length(J)

P(k)=(Tau(visited(end),J(k))^Alpha)\*(Eta(visited(end),J(k))^Beta);

end

P=P/(sum(P));

%按概率原则选取下一个城市

Pcum=cumsum(P);     %cumsum，元素累加即求和

Select=find(Pcum>=rand); %若计算的概率大于原来的就选择这条路线

to\_visit=J(Select(1));

Tabu(i,j)=to\_visit;

end

end

if NC>=2

Tabu(1,:)=R\_best(NC-1,:);

end

%%第四步：记录本次迭代最佳路线

L=zeros(m,1);     %开始距离为0，m\*1的列向量

for i=1:m

R=Tabu(i,:);

for j=1:(n-1)

L(i)=L(i)+D(R(j),R(j+1));    %原距离加上第j个城市到第j+1个城市的距离

end

L(i)=L(i)+D(R(1),R(n));      %一轮下来后走过的距离

end

L\_best(NC)=min(L);           %最佳距离取最小

pos=find(L==L\_best(NC));

R\_best(NC,:)=Tabu(pos(1),:); %此轮迭代后的最佳路线

L\_ave(NC)=mean(L);           %此轮迭代后的平均距离

NC=NC+1                      %迭代继续

%%第五步：更新信息素

Delta\_Tau=zeros(n,n);        %开始时信息素为n\*n的0矩阵

for i=1:m

for j=1:(n-1)

Delta\_Tau(Tabu(i,j),Tabu(i,j+1))=Delta\_Tau(Tabu(i,j),Tabu(i,j+1))+Q/L(i);

%此次循环在路径（i，j）上的信息素增量

end

Delta\_Tau(Tabu(i,n),Tabu(i,1))=Delta\_Tau(Tabu(i,n),Tabu(i,1))+Q/L(i);

%此次循环在整个路径上的信息素增量

end

Tau=(1-Rho).\*Tau+Delta\_Tau; %考虑信息素挥发，更新后的信息素

%%第六步：禁忌表清零

Tabu=zeros(m,n);             %%直到最大迭代次数

end

%%第七步：输出结果

Pos=find(L\_best==min(L\_best)); %找到最佳路径（非0为真）

Shortest\_Route=R\_best(Pos(1),:) %最大迭代次数后最佳路径

Shortest\_Length=L\_best(Pos(1)) %最大迭代次数后最短距离

subplot(1,2,1)                  %绘制第一个子图形

DrawRoute(C,Shortest\_Route)     %画路线图的子函数

subplot(1,2,2)                  %绘制第二个子图形

plot(L\_best)

hold on                         %保持图形

plot(L\_ave,'r')

title('平均距离和最短距离')     %标题

function DrawRoute(C,R)

%%=========================================================================

%% DrawRoute.m

%% 画路线图的子函数

%%-------------------------------------------------------------------------

%% C Coordinate 节点坐标，由一个N×2的矩阵存储

%% R Route 路线

%%=========================================================================

N=length(R);

scatter(C(:,1),C(:,2));

hold on

plot([C(R(1),1),C(R(N),1)],[C(R(1),2),C(R(N),2)],'g')

hold on

for ii=2:N

plot([C(R(ii-1),1),C(R(ii),1)],[C(R(ii-1),2),C(R(ii),2)],'g')

hold on

end

title('旅行商问题优化结果 ')