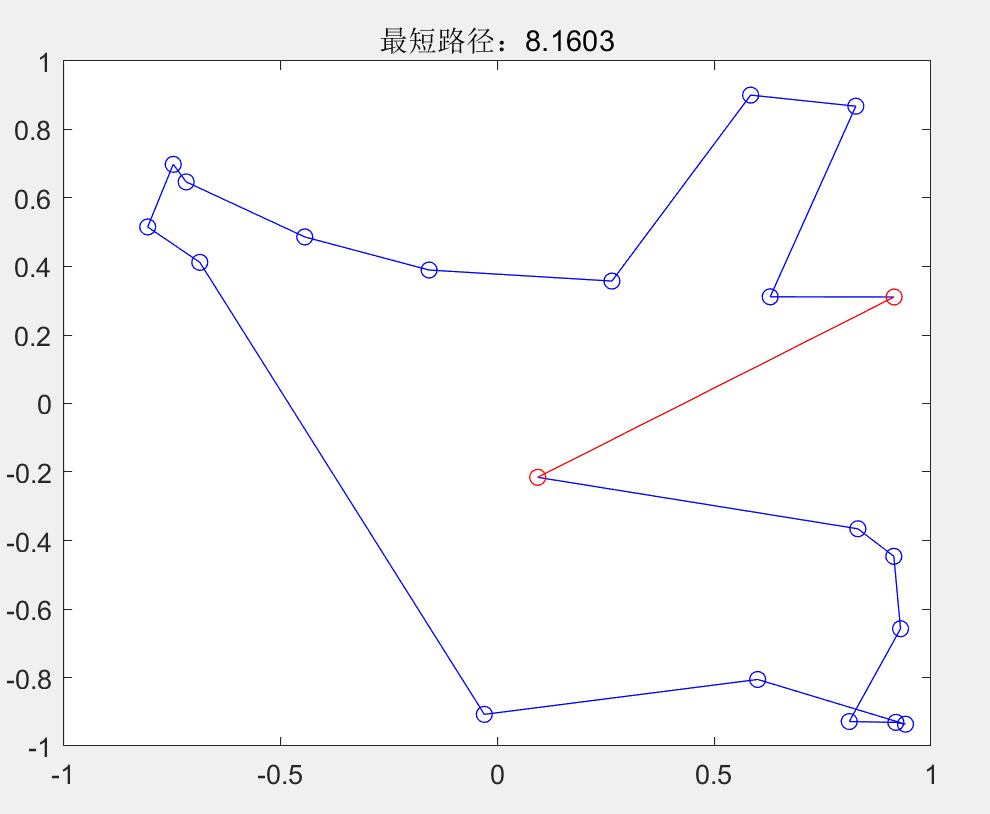
实验七 蚁群算法

姓名：唐川淇 学号：1131190111

# 问题重述

在二维平面中随机生成20个城市的坐标，用蚂蚁算法（蚁群算法）求解TSP问题。TSP问题是指，一个人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。



# 模型的建立与求解

## 初始化参数

首先根据题目要求初始化各类参数，其中包括参数蚂蚁的数量m，信息素的重要程度alpha，启发式因子的重要程度beta，信息素的蒸发系数rho，迭代次数G。同时初始化信息素矩阵Tau，用于存储任意两点之间的信息素的大小，禁忌表用来记录每次迭代每只蚂蚁的路线Tabu,以及变量存储各代最佳路线以及每一代的最佳路线。其中MATLAB代码如下：

|  |
| --- |
| 代码 |
| 介绍：初始化参数 |
| m = 3; % 蚂蚁的个数 一般取10-50  alpha = 1; % 信息素的重要程度  beta = 2; % 启发式英子的重要程度  rho = 0.5; % 信息素蒸发系数  G = 20; % 迭代数  Q = 1; %  Eta = 1./distance; % 启发式因子  Tau = ones(M,M); % 信息素矩阵存储着每两个城市之间的信息素的数值  Tabu = zeros(m,M); % 禁忌表，记录每只蚂蚁走过的路程  R\_best = zeros(G,M); % 各代的最佳路线  L\_best = inf.\*ones(G,1); % 每一代的最佳路径的长度 初始假设为无穷大 |

接下来随机产生20个城市的坐标，并用欧氏距离计算任意两点之间的距离做为他们的距离，其中代码如下：

|  |
| --- |
| 代码 |
| 介绍：初始化距离 |
| C = rands(20,2);  % M为问题的规模 M个城市  [M,N] = size(C);    % 记录任意两个城市之间的距离  % 求任意两个城市之间的距离  distance=count\_distance(C); |

其中count\_distance函数是一个计算距离的函数，它可以计算出所有城市之间的距离，并存储在一个矩阵中，其中函数的定义如下：

|  |
| --- |
| 代码 |
| 介绍：计算距离的函数 |
| function distance=count\_distance(C)  [M,~] = size(C);  distance = zeros(M,M);  for m=1:M  for n=1:M  distance(m,n) = sqrt(sum((C(m,:)-C(n,:)).^2));  end  end  end |

至此，所有提前的准备工作已经完成，接下来需要通过蚁群算法计算出最合适的路径。

## 迭代计算

蚁群算法是一种用来寻找优化路径的概率型算法。它由Marco Dorigo于1992年在他的博士论文中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。蚁群算法应用于解决优化问题的基本思路为：用蚂蚁的行走路径表示待优化问题的可行解，整个蚂蚁群体的所有路径构成待优化问题的解空间。路径较短的蚂蚁释放的信息素量较多，随着时间的推进，较短的路径上累积的信息素浓度逐渐增高，选择该路径的蚂蚁个数也愈来愈多。最终，整个蚂蚁会在正反馈的作用下集中到最佳的路径上，此时对应的便是待优化问题的最优解。

### 生成随机路线、

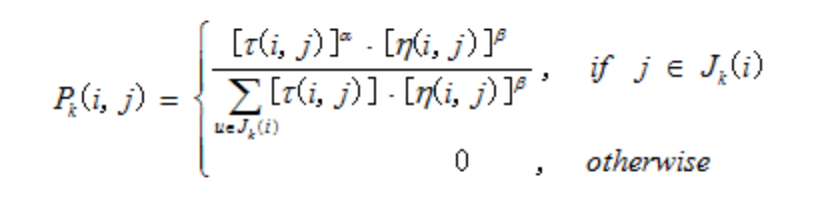
首先将m只蚂蚁随机放到M个城市中，我定义了函数put\_ant\_random，如下：

|  |
| --- |
| 代码 |
| 介绍：放蚂蚁 |
| function T=put\_ant\_random(m,M)  random\_pos = [];  for i=1:(ceil(m/M)) % m只蚂蚁随即放到M座城市  random\_pos = [random\_pos,randperm(M)];  % 在random\_pos 中随机选择m个数，代表蚂蚁的初始城市  end  T=(random\_pos(1,1:m))';  end |

将蚂蚁随机的放到不同的城市之后就可以随机生成路线，我定义了函数create\_newroute来生成函数，具体如下：

|  |
| --- |
| 代码 |
| 介绍：循环生成路线 |
| function [Tau,Tabu]=create\_newroute(m,M,Tabu,Tau,alpha,beta,Eta,R\_best,gen)  for i=2:M % 从第二个城市开始  for j=1:m % 每只蚂蚁  visited = Tabu(j,1:(i-1)); % 在访问第i个城市的时候，第j个蚂蚁访问过的城市  unvisited = zeros(1,(M+1-i)); % 待访问的城市  visit\_P = unvisited; % 蚂蚁j访问剩下的城市的概率  count = 1;  for k=1:M % 这个循环是找出未访问的城市  if isempty(find(visited==k)) %还没有访问过的城市 如果成立。则证明第k个城市没有访问过  unvisited(count) = k;  count = count+1;  end  end  % 计算待选择城市的概率  for k=1:length(unvisited) % Tau(visited(end),unvisited(k))访问过的城市的最后一个与所有未访问的城市之间的信息素  visit\_P(k) = ((Tau(visited(end),unvisited(k)))^alpha)\*(Eta(visited(end),unvisited(k))^beta);  end  visit\_P = visit\_P/sum(visit\_P); % 访问每条路径的概率的大小  % 按照概率选择下一个要访问的城市  % 用轮盘赌选择方法  Pcum = cumsum(visit\_P);  selected = find(Pcum>=rand);  to\_visited = unvisited(selected(1));  Tabu(j,i) = to\_visited; % 添加到禁忌表  end  end  if gen>=2  Tabu(1,:) = R\_best(gen-1,:);  end  end |

从第二个城市开始迭代，首先将访问的第一个城市加入禁忌表，之后每轮循环计算选择城市的概率，其中计算访问城市概率的函数定义如下：



使用轮盘赌的方法选择下一次访问的城市，然后将选择的城市加入禁忌表，之后继续迭代，最终可以生成每只蚂蚁的访问路线。

### 记录最佳路线

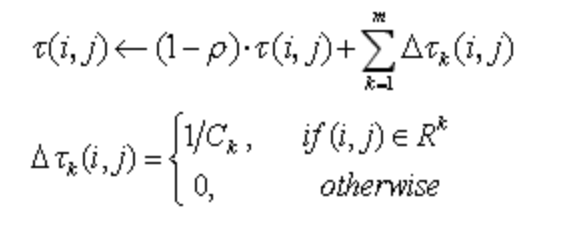
在得到了每只蚂蚁的游览路线之后，需要从所有蚂蚁中找出最优的那条路线，这里我定义了函数find\_best\_distance去找到最优的路线，函数定义如下：

|  |
| --- |
| 代码 |
| 介绍：寻找最优路线 |
| function [L\_best,R\_best,L]=find\_best\_distance(m,M,Tabu,distance,L\_best,R\_best,gen)  L = zeros(1,m);  for i=1:m  R = Tabu(i,:);  L(i) = distance(R(M),R(1)); % 因为要走一周回到原来的地点  for j=1:(M-1)  L(i) = L(i)+distance(R(j),R(j+1));  end  end  L\_best(gen) = min(L); % 记录每一代中路径的最短值  pos = find(L==L\_best(gen));  R\_best(gen,:) = Tabu(pos(1),:); % 最优的路径  end |

根据之前的距离可以计算每条路线的走过的距离，要注意的是，最重要走回起点，所以需要增加最终节点到最初节点的路线。计算出每一条路线的总长度之后，找出其中的最短路线。

### 更新信息素

找到最优路线之后需要更新不同城市之间的信息素，我定义了函数renew\_tau来更新城市之间路线的信息素，具体的公式如下：



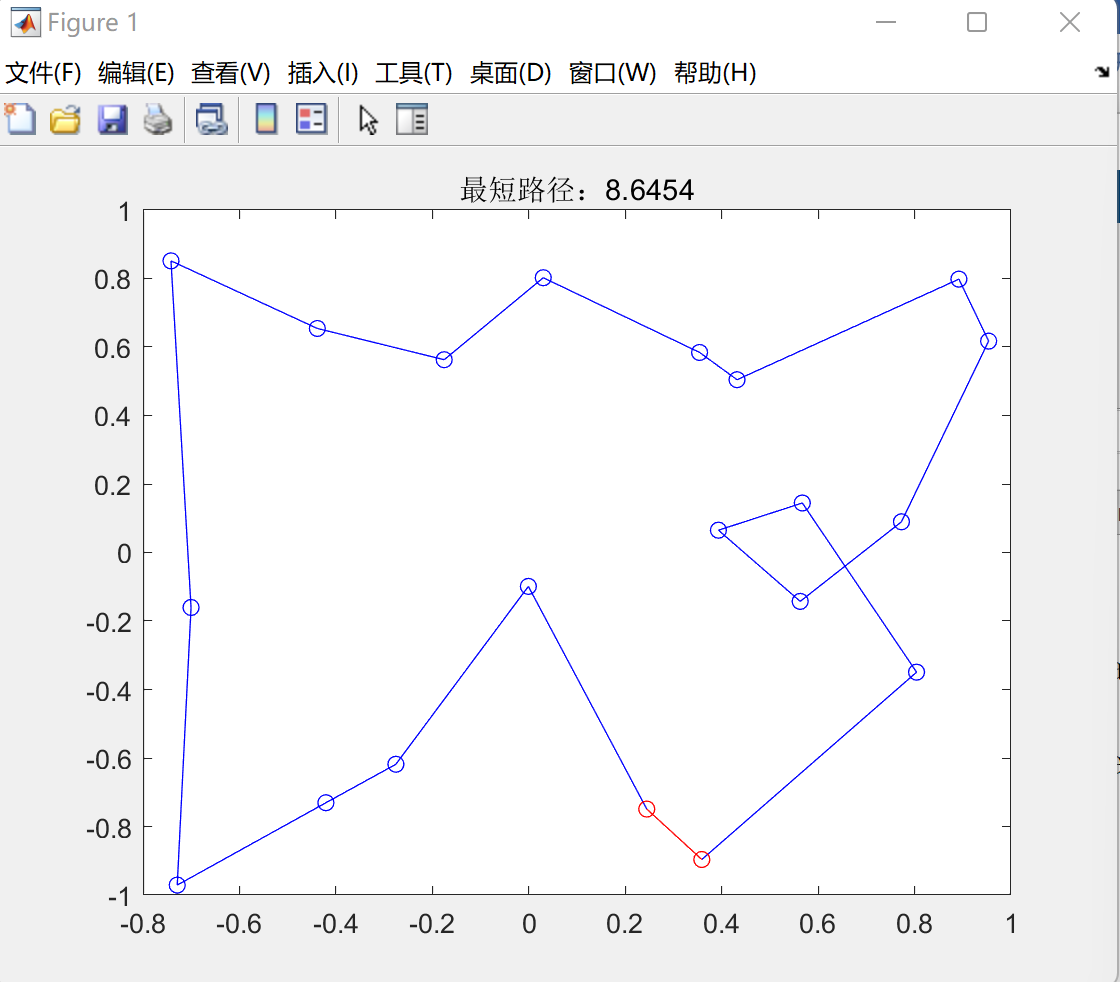
代码如下：

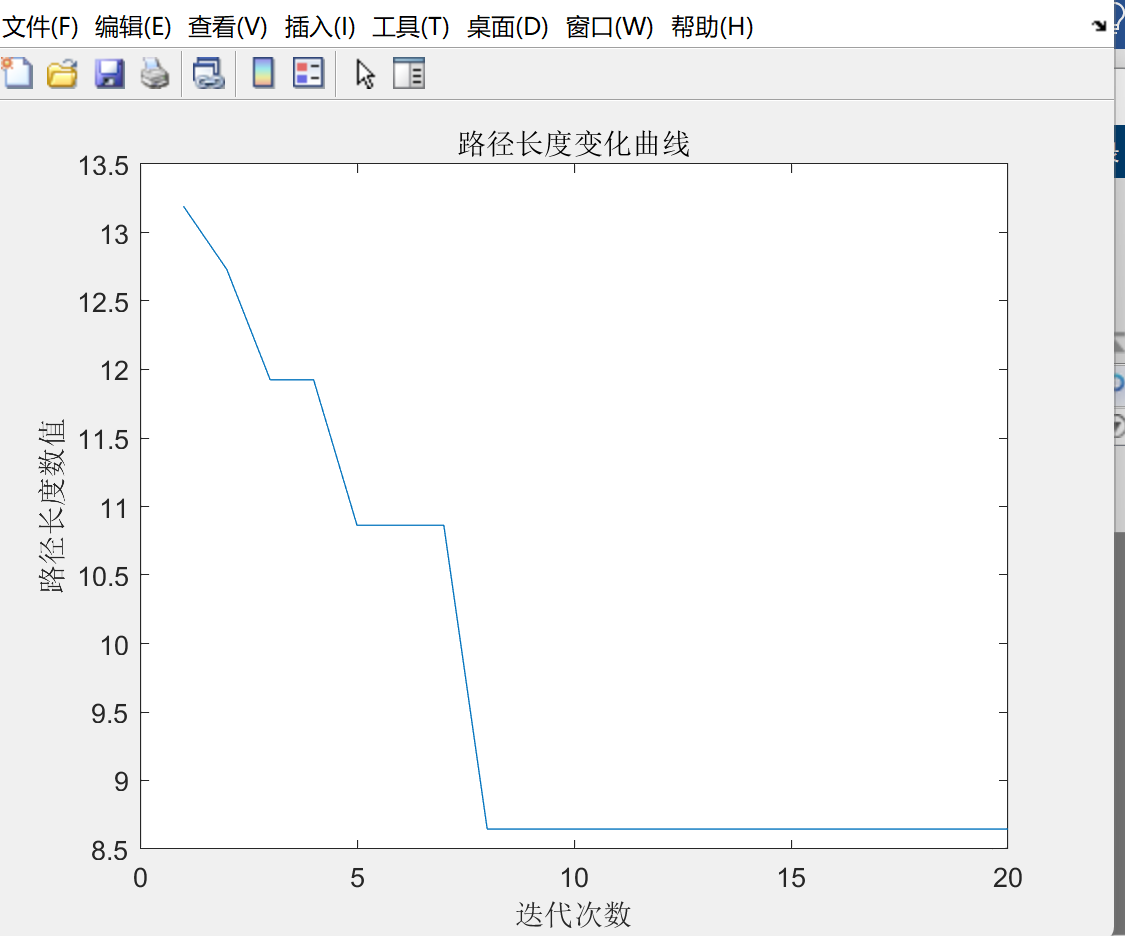
|  |
| --- |
| 代码 |
| 介绍：更新信息素 |
| function Tau=renew\_tau(m,M,Tau,Tabu,Q,L,rho)  Delta\_Tau = zeros(M,M);  for i=1:m % m只蚂蚁  for j=1:(M-1) % M座城市  Delta\_Tau(Tabu(i,j),Tabu(i,j+1)) = Delta\_Tau(Tabu(i,j),Tabu(i,j+1)) + 1/L(i);  end  Delta\_Tau(Tabu(i,M),Tabu(i,1)) = Delta\_Tau(Tabu(i,M),Tabu(i,1)) + Q/L(i);  end  Tau = (1-rho).\*Tau+Delta\_Tau; % 更新路径上的信息素含量  end |

### 禁忌表清0

由于要重新生成路线，所以最后需要将禁忌表清零。

## 结果展示

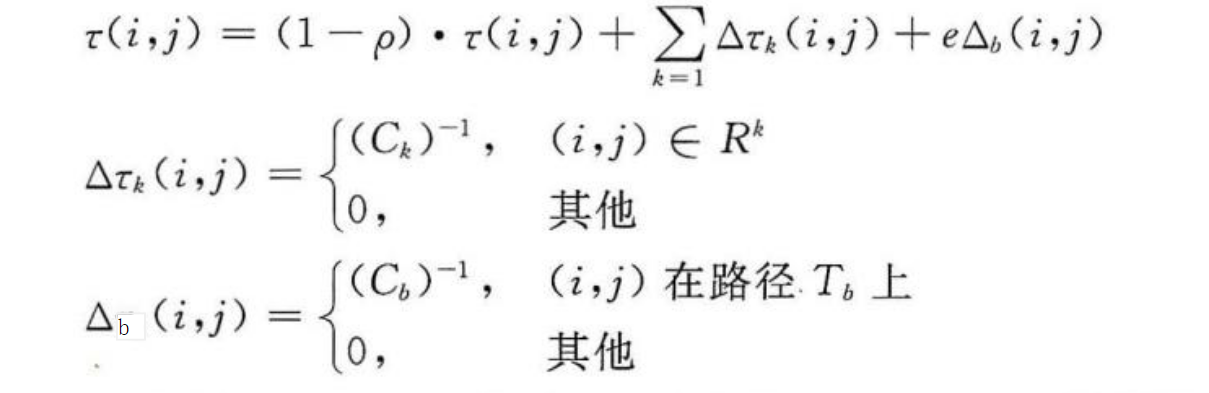




# 模型的改进

EAS是对基础 AS 的第一次改进，它在原 AS 信息素更新的基础上，增加了一个对至今最优路径的强化手段。在每轮信息素更新完毕后，搜索到至今最优路径( Tb )的那只蚂蚁会为这条路径添加额外的信息素。

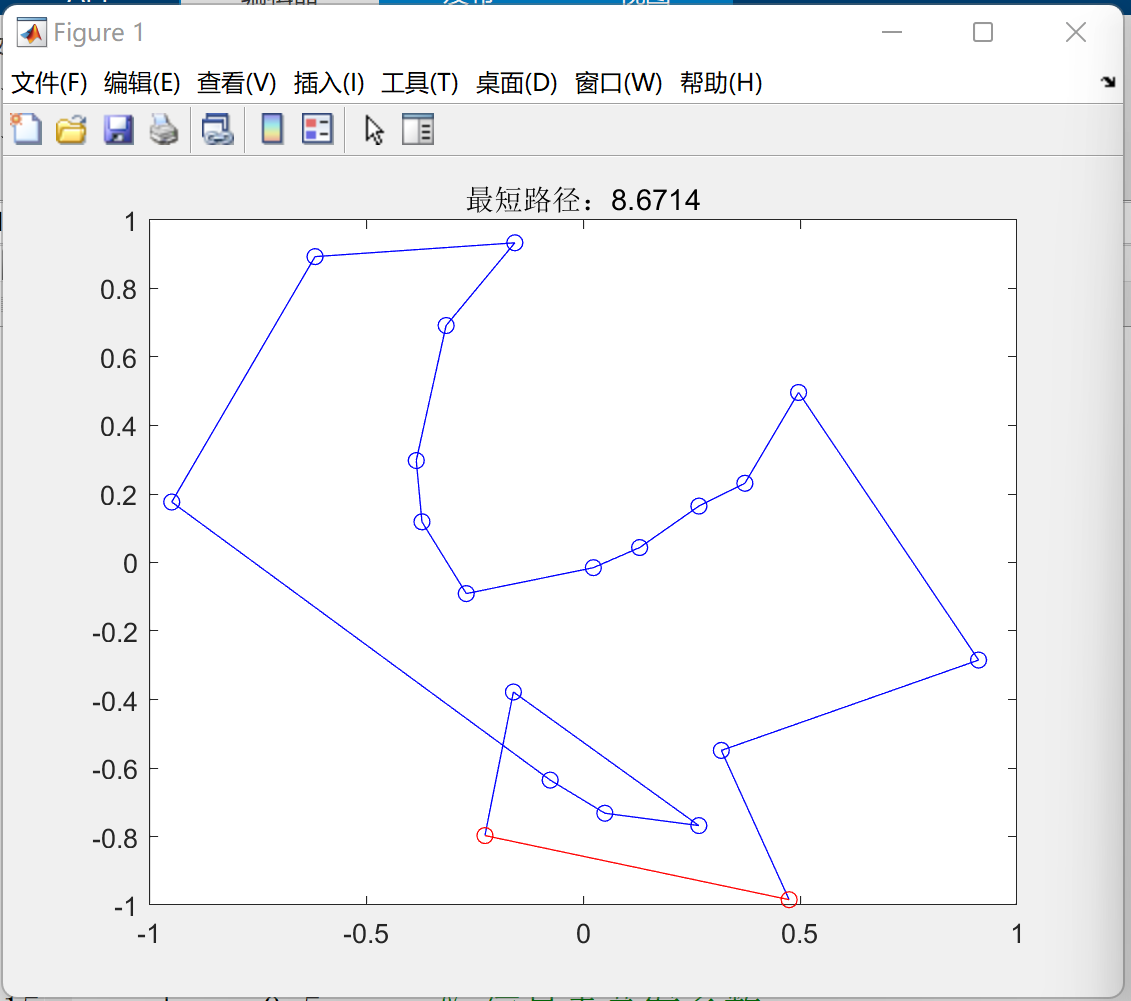
其中信息素更新公式如下所示：

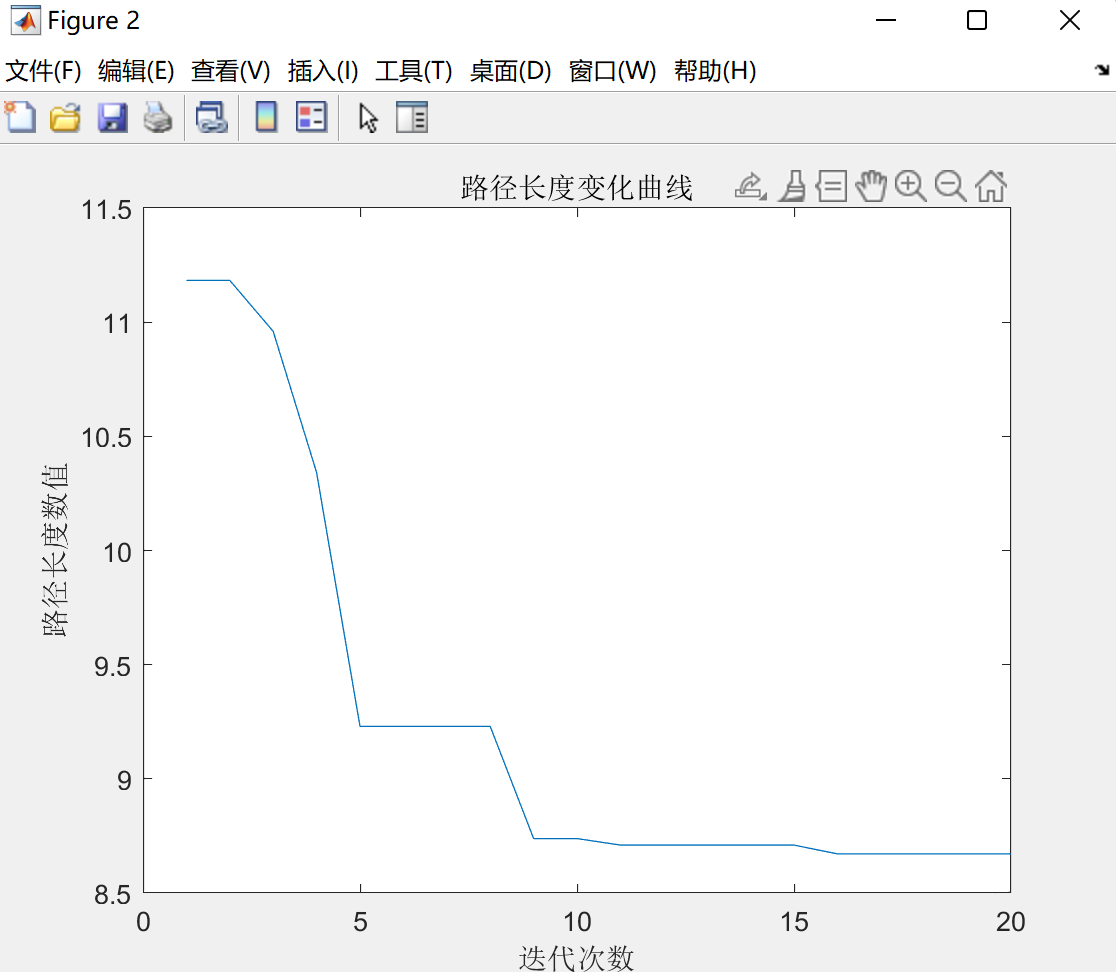


修改后的信息素计算代码，其中的Q为公式中的e，具体如下：

|  |
| --- |
| 代码 |
| 介绍：完整主函数 |
| function Tau=renew\_tau(m,M,Tau,Tabu,Q,L,rho)  Delta\_Tau = zeros(M,M);  for i=1:m % m只蚂蚁  for j=1:(M-1) % M座城市  Delta\_Tau(Tabu(i,j),Tabu(i,j+1)) = Delta\_Tau(Tabu(i,j),Tabu(i,j+1)) + Q/L(i);  end  Delta\_Tau(Tabu(i,M),Tabu(i,1)) = Delta\_Tau(Tabu(i,M),Tabu(i,1)) + Q/L(i);  end  Tau = (1-rho).\*Tau+Delta\_Tau; % 更新路径上的信息素含量  end |

最终得到结果：





附录

|  |
| --- |
| 代码 |
| 介绍：完整主函数 |
| clear all;  close all;  clc;  C = rands(20,2);  % M为问题的规模 M个城市  [M,N] = size(C);    % 记录任意两个城市之间的距离  % 求任意两个城市之间的距离  distance=count\_distance(C);    m = 3; % 蚂蚁的个数 一般取10-50  alpha = 1; % 信息素的重要程度  beta = 2; % 启发式英子的重要程度  rho = 0.5; % 信息素蒸发系数  G = 20; % 迭代数  Q = 1; %  Eta = 1./distance; % 启发式因子  Tau = ones(M,M); % 信息素矩阵 存储着每两个城市之间的信息素的数值  Tabu = zeros(m,M); % 禁忌表，记录每只蚂蚁走过的路程  R\_best = zeros(G,M); % 各代的最佳路线  L\_best = inf.\*ones(G,1); % 每一代的最佳路径的长度 初始假设为无穷大  % 开始迭代计算  for gen=1:G  % 将m只蚂蚁放到n个城市上  Tabu(:,1) = put\_ant\_random(m,M);  % 迭代  [Tau,Tabu]=create\_newroute(m,M,Tabu,Tau,alpha,beta,Eta,R\_best,gen);  % 记录最佳路线  [L\_best,R\_best,L]=find\_best\_distance(m,M,Tabu,distance,L\_best,R\_best,gen);  % 更新信息素  Tau=renew\_tau(m,M,Tau,Tabu,Q,L,rho);  % 禁忌表清零  Tabu = zeros(m,M);  % 画图  for i=1:(M-1)  plot([C(R\_best(gen,i),1),C(R\_best(gen,i+1),1)],[C(R\_best(gen,i),2),C(R\_best(gen,i+1),2)],'bo-');  hold on;  end  plot([C(R\_best(gen,M),1),C(R\_best(gen,1),1)],[C(R\_best(gen,M),2),C(R\_best(gen,1),2)],'ro-');  title(['最短路径：',num2str(L\_best(gen))]);  hold off;  pause(0.05);  gen = gen+1;  end  figure(2);  plot(L\_best);  title('路径长度变化曲线');  xlabel('迭代次数');  ylabel('路径长度数值'); |