南京邮电大学

《数据结构与算法课程设计》

课程设计报告

（ 2016 / 2017 学年 第 2 学期）

题 目： 蚁群算法在旅行商问题中的应用

--算法思想解读、Matlab 代码实现、优化方法的提出

专 业 网络工程

姓 名 刘美含

学 号 B15070204

指导教师 许 斌

指导单位 物联网学院

日 期 2017 年 6 月 3 日

评分细则	评分项	成绩
	遵守机房规章制度（5 分）	
	上机时的表现（5 分）	
	学习态度（5 分）	
	程序准备情况（5 分）	
	程序设计能力（10 分）	
	团队合作精神（5 分）	
	课题功能实现情况（10 分）	
	算法设计合理性（10 分）	
	用户界面设计（10 分）	
	报告书写认真程度（5 分）	
	内容详实程度（10 分）	
	文字表达熟练程度（10 分）	
	回答问题准确度（10 分）	
简短评语	教师签名： 年　月　日	
评分等级		
备注	评分等级有五种：优秀、良好、中等、及格、不及格	

蚁群算法在旅行商问题中的应用

--算法思想解读、Matlab 代码实现、优化方法的提出

一、课题内容和要求

1、问题描述

蚁群算法是在对真实蚂蚁的观测基础上提出的，单个蚂蚁是不具智能的，但生活在一起的蚁群却总是能够在蚁穴和食物间找到一条几乎是最短的路径。这就要靠蚁群的智能，蚁群算法就基于这一智能。图 1-1 给出蚁群智能的基本思想。

蚂蚁会在自己走过的路径上留下生化信息素，同时它也会选择地面上信息素较多的路径行走。对于下图的第二种情况，因为路上有了障碍物，所以蚂蚁需要绕过障碍物，该怎么样绕过障碍物，由于没有信息素作为指导，所以起先只能是随机的选择从左或从右走（第三个图所示）；但是随着行走次数的增加，较短的路径上留下的信息素就会较多，就有更多的蚂蚁行走，信息素进一步增多，最终较短的路就成为了选择的路（第四个图）。这就是蚁群智能的基本原理。

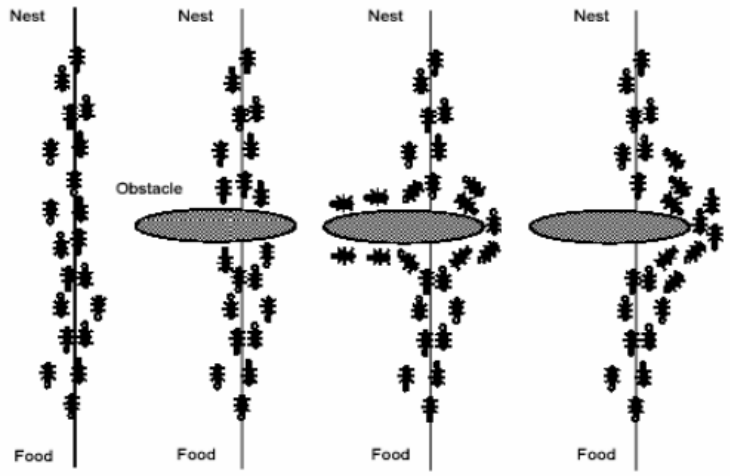

图 1-1 蚁群算法的基本思想

蚁群算法就是应用蚁群智能实现的算法，可以用它来实现在解空间上进行最优解的搜索，很多计算机问题可以转化为搜索最优解的问题。如旅行商问题（求解遍历全部城市的最短路经），就可用蚁群算法求解，在旅行商问题中，蚁穴到食物就对应遍历全部城市，蚂蚁就对应旅行商。

2、课程设计目的

了解蚁群算法的思想，并能应用蚁群算法求解具体问题。

3、基本要求

（1）应用蚁群算法求解 TSP 问题。

（2）TSP 中的城市数量不少于 30 个，组成完全图，边上的权值自定。

（3）蚂蚁数量可配置，迭代次数可配置。

（4）给出较全面的实验结果：结果路经及长度；蚁群算法执行时间；不同参数值（蚂蚁数量，迭代次数）的影响等。

（5）迭代过程需要用图形界面表示。

二、需求分析

1、使用蚁群算法解决基本 TSP 问题。TSP 的任务是求一条从起点出发周游所有城市一次又回到起点的最短路径，用传统的算法求解该问题十分繁琐。而蚁群算法作为现代智能优化算法，有鲁棒性强、全局搜索能力强、易于与其他算法相结合的优点，因此 TSP 问题能得到较好解决。

2、探究不同参数值对蚁群算法求解过程中的影响。蚁群算法也是一种启发式算法，该类算法的一个主要特征就是牺牲求解结果的精确性换取求解的效率，因而通常不可能在每次执行算法时都得到问题的精确最优解，往往只是不断的逼近问题的最优解。这个特点决定了此算法在求解实际问题的过程中有很大的调整余地，而对算法的调整主要就体现在参数的具体选择上。因而有必要讨论一下各个参数地选择，以便选取合适的参数值解决问题。

3、蚁群算法的优化。虽然蚁群算法有上述优点，但与已经发展成熟的一些启发式算法比较起来，还存在计算量比较大、搜索时间比较长、易出现停滞现象等缺点。并且，在人工蚁群算法中经常出现信息素浓度最强的路径不是所需要的最优路径的情况。因此有必要对此算法进行改进和优化。

三、算法准备

由于我们对此算法较陌生，因此在设计算法前我们对此算法做了大量的调研工作。一方面，我们首先明确了要解决的问题和算法适用范围，其次我们明确了此算法的原理和算法流程，最后我们了解了此算法的优劣，并产生对其进行优化改进的想法；另一方面，此算法在互联网上有大量的参考代码，我们尝试调试运行，研究代码思路，观察运行结果，直观的感受此算法的魅力。

1、TSP 问题简介

给定 n 个城市的集合{1,2 …n}及城市之间环游的花费C_{ij}(1 ≤ i ≤ n, 1 ≤ j ≤ n, i ≠ j)。TSP 问题是指找到一条经过每个城市一次且回到起点的最小花费的环游。若将每个顶点看成是图上的节点，花费C_{ij} 为连接顶点 V_i、V_j边上的权，则 TSP 问题就是在一个具有 n 个节点的完全图上找到一条花费最小的 Hamilton 回路。

2、算法概述

蚁群算法的基本思想来源于自然界蚂蚁觅食的最短路径原理，根据昆虫科学家的观察，发现自然界的蚂蚁虽然视觉不发达，但它们可以在没有任何提示的情况下找到从食物源到巢穴的最短路径，并在周围环境发生变化后，自适应地搜索新的最佳路径。

蚂蚁在寻找食物源的时候，能在其走过的路径上释放一种叫信息素的激素，使一定范围内的其他蚂蚁能够察觉到。当一些路径上通过的蚂蚁越来越多时，信息素也就越来越多，蚂蚁们选择这条路径的概率也就越高，结果导致这条路径上的信息素又增多，蚂蚁走这条路的概率又增加，生生不息。这种选择过程被称为蚂蚁的自催化行为。对于单个蚂蚁来说，它并没有要寻找最短路径，只是根据概率选择；对于整个蚁群系统来说，它们却达到了寻找到最优路径的客观上的效果。这就是群体智能。

3、算法目的

蚁群算法根据模拟蚂蚁寻找食物的最短路径行为来设计的仿生算法，因此一般而言，蚁群算法用来解决最短路径问题，并真的在旅行商问题（TSP，一个寻找最短路径的问题）上取得了比较好的成效。目前，也已渐渐应用到其他领域中去，在图

着色问题、车辆调度问题、集成电路设计、通讯网络、数据聚类分析等方面都有所应用。

4、算法原理

蚁群算法是对自然界蚂蚁的寻径方式进行模拟而得出的一种仿生算法。为了说明蚁群算法的原理,先简要介绍一下蚂蚁搜寻食物的具体过程。在蚂蚁群找到食物时，它们总能找到一条从食物到巢穴之间的最优路径。这是因为蚂蚁在寻找路径时会在路径上释放出一种特殊的信息素。当它们碰到一个还没有走过的路口时,就随机地挑选一条路径前行。与此同时释放出与路径长度有关的信息素。路径越长,释放的激素浓度越低。当后来的蚂蚁再次碰到这个路口的时候,选择激素浓度较高路径概率就会相对较大。这样形成了一个正反馈。最优路径上的激素浓度越来越大,而其它的路径上激素浓度却会随着时间的流逝而消减。最终整个蚁群会找出最优路径。不仅如此,蚂蚁还能够适应环境的变化,当蚁群运动路线上突然出现障碍物的,蚂蚁能够很快地重新找到最优路径。这个过程和前面所描述的方式是一致的。在整个寻径过程中,虽然单个蚂蚁的选择能力有限,但是通过激素的作用,整个蚁群之间交换着路径信息,最终找出最优路径。

5、算法特点

（1）蚁群算法是一种自组织的算法。在系统论中，自组织和它组织是组织的两个基本分类，其区别在于组织力或组织指令是来自于系统的内部还是来自于系统的外部，来自于系统内部的是自组织，来自于系统外部的是他组织。如果系统在获得空间的、时间的或者功能结构的过程中，没有外界的特定干预，我们便说系统是自组织的。在抽象意义上讲，自组织就是在没有外界作用下使得系统熵减小的过程(即是系统从无序到有序的变化过程)。蚁群算法充分体现了这个过程，以蚂蚁群体优化为例子说明。当算法开始的初期，单个的人工蚂蚁无序的寻找解，算法经过一段时间的演化，人工蚂蚁间通过信息激素的作用，自发的越来越趋向于寻找到接近最优解的一些解，这就是一个无序到有序的过程。

（2）蚁群算法是一种本质上并行的算法。每只蚂蚁搜索的过程彼此独立，仅通过信息激素进行通信。它在问题空间的多点同时开始进行独立的解搜索，不仅增加了算法的可靠性，也使得算法具有较强的全局搜索能力。

（3）蚁群算法是一种正反馈的算法。从真实蚂蚁的觅食过程中我们不难看出，蚂蚁能够最终找到最短路径，直接依赖于最短路径上信息激素的堆积，而信息激素的堆积却是一个正反馈的过程。对蚁群算法来说，初始时刻在环境中存在完全相同的信息激素，给予系统一个微小扰动，使得各个边上的轨迹浓度不相同，蚂蚁构造的解就存在了优劣，算法采用的反馈方式是在较优的解经过的路径留下更多的信息激素，而更多的信息激素又吸引了更多的蚂蚁，这个正反馈的过程使得初始的不同得到不断的扩大，同时又引导整个系统向最优解的方向进化。因此，正反馈是蚂蚁算法的重要特征，它使得算法演化过程得以进行。

（4）蚁群算法具有较强的鲁棒性。相对于其它算法，蚁群算法对初始路线要求不高，即蚁群算法的求解结果不依赖于初始路线的选择，而且在搜索过程中不需要进行人工的调整。其次，蚁群算法的参数数目少，设置简单，易于蚁群算法应用到其它组合优化问题的求解。

四、概要设计

1、算法流程

Step 1: 对相关参数进行初始化，包括蚁群规模、信息素因子、启发函数因子、信息素挥发因子、信息素常数、最大迭代次数等，以及将数据读入程序，并进行预处理：比如将城市的坐标信息转换为城市间的距离矩阵。

Step 2: 随机将蚂蚁放于不同出发点。

Step 3: 蚂蚁按概率函数选择下一座城市，直到有蚂蚁访问完所有城市。

Step 4: 计算各蚂蚁经过的路径长度，记录当前迭代次数最优解。

Step 5: 对路径上的信息素浓度进行更新。

Step 6: 判断是否达到最大迭代次数，若否，返回步骤 2；是，结束程序。

Step 7: 输出结果。

流程图如图 4-1 所示：

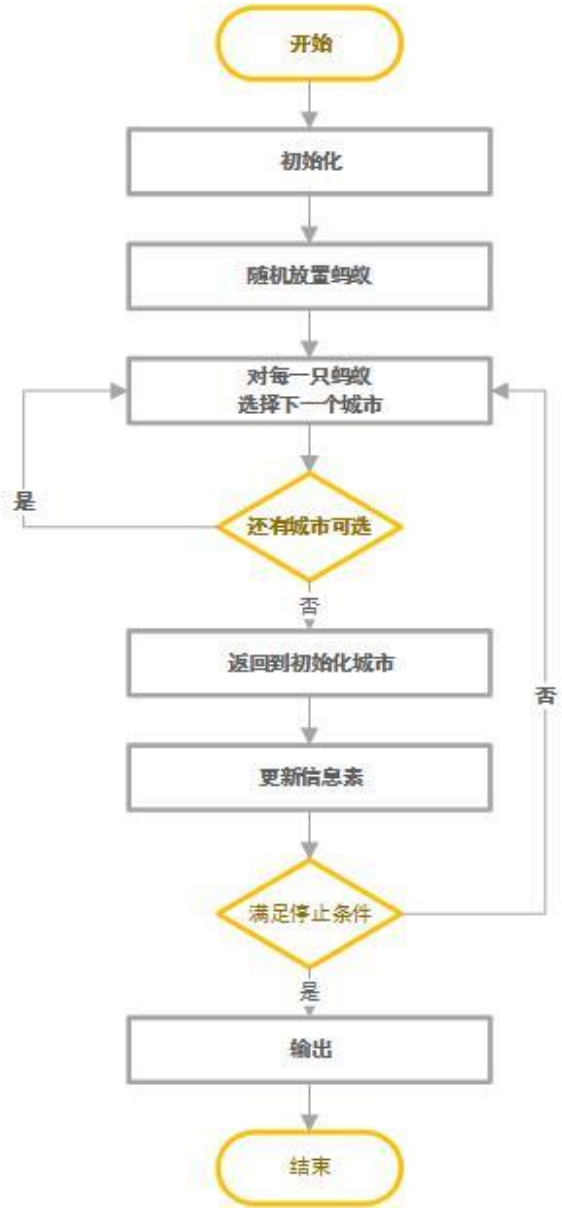

图 4-1 算法流程图

2、主要符号说明

（1）变量（如表 1 所示）

表 1 变量解释

变量名	解释
m	蚂蚁数量
n	城市数量
Nc_max	迭代次数
alpha	信息素重要程度
beta	启发式因子在蚂蚁选择路径时的重要程度
rho	路径上信息衰减速度
Q	信息素总量
Nc	当前迭代次数
Nc_max	迭代总次数

（2）矩阵（如表 2 所示）

表 2 矩阵解释

矩阵名	大小	解释
D	n*n	城市完全图复权邻接矩阵
Tau	n*n	信息素矩阵
Tabu	m*n	蚂蚁路径记录
R_best	Nc_max*n	各个迭代最佳路线

（3）参数解释

蚂蚁数量：

蚂蚁过大时，会导致搜索过的路径上信息素变化趋于平均，这样就不好找出好的路径了；蚂蚁数量过小时，易使未被搜索到的路径信息素减小到 0，这样可能会出现早熟，没找到全局最优解。根据论文，城市规模大致是蚂蚁 2.3 倍时，蚁群算法的全局收敛能力和收敛速度都较好。

信息素因子：

信息素因子反映了蚂蚁在移动过程中所积累的信息量在指导蚁群搜索中的相对重要程度，其值过大，蚂蚁选择以前走过的路径概率大，搜索随机性减弱；值过小，等同于贪婪算法，使搜索过早陷入局部最优。实验发现，信息素因子选择[1,4]区间，性能较好。

启发函数因子：

启发函数因子反映了启发式信息在指导蚁群搜索过程中的相对重要程度，其大小反映的是蚁群寻优过程中先验性和确定性因素的作用强度。过大时，虽然收敛速

度会加快，但容易陷入局部最优；过小时，容易陷入随机搜索，找不到最优解。实验研究发现，当启发函数因子为[3,4.5]时，综合求解性能较好。

信息素挥发因子：

信息素挥发因子表示信息素的消失水平，它的大小直接关系到蚁群算法的全局搜索能力和收敛速度。实验发现，当属于[0.2，0.5]时,综合性能较好。

信息素常数：

这个参数为信息素强度，表示蚂蚁循环一周时释放在路径上的信息素总量，其作用是为了充分利用有向图上的全局信息反馈量，使算法在正反馈机制作用下以合理的演化速度搜索到全局最优解。值越大,蚂蚁在已遍历路径上的信息素积累越快，有助于快速收敛。实验发现，当值属于[10,1000]时，综合性能较好。

最大迭代次数：

最大迭代次数值过小，可能导致算法还没收敛就已结束；过大则会导致资源浪费。一般最大迭代次数可以取 100 到 500 次。一般来讲，建议先取 200，然后根据执行程序查看算法收敛的轨迹来修改取值。

4、算法复杂度分析

（1）时间复杂度

下面的 7 个步骤是对蚁群算法执行步骤的简单说明，后面的数据是根据各个步骤实现过程得到的各自的渐进时间复杂度，其中 n 为旅行商问题的规模，m 是蚁群算法采用的人工蚂蚁数量。

Step1:初始化参数$O(n^2 + m)$

loop

Step2:设置蚂蚁禁忌表$O(m)$

Step3:每个蚂蚁单独构造解$O(n^2m)$

Step4:解的评价和轨迹更新量的计算$O(n^2m)$

Step5:轨迹浓度的更新$O(n^2)$

Step6:判断是否达到预先设定的最大循环值$O(nm)$

如果没有，转 ***loop***

Step7:结果输出$O(1)$

另外，在 ***Step2-Step7*** 有一个循环，其循环变量为Nc，最大循环次数为Nc_Max，故我们容易得出，蚂蚁算法整个过程的时间复杂度为。$O(Nc \cdot n^2 \cdot m)$。如果选取$m = n$，则蚁群算法的时间复杂度为$O(Nc \cdot n^3)$。算法理论认为，这个复杂度在计算时间上是可以接受的。

（2）空间复杂度

考虑算法的空间复杂度，首先需要分析算法在实现中需要用到的数据。从第三章的描述中可以知道，蚁群算法中的数据主要实现两个方面的功能，一是问题的描述，二是实现蚁群算法功能的辅助数据。以旅行商问题为例，蚁群算法求解通常采用图来描述问题，在数据结构上，如果问题的规模为 n 维，则需要用一个 n 阶二维距离矩阵 D 描述问题本身的特征，为了表示图上的信息激素，需要用另外一个 n 阶二维矩阵来表示图上的信息激素浓度;在蚁群算法求解的过程中为了保证走过的城市不再被选择，需要为每一只蚂蚁设立一个禁忌表，它是一个 n 阶一维数组;为了保存蚂蚁寻到的解，需要为每一只蚂蚁设立一个数组;为了评价解的优劣，需定义一个变量;为了信息激素的更新，需要设置每条路经上的信息激素更新量，这是一个二维数组。以上这些就是蚁群算法中要用到的基本存储量。同样的，在算法的空间复杂度上也有着渐进空间复杂度的概念，其定义和渐进时间复杂度是相似的。对蚁群算法实现的各个步骤分别进行空间复杂度分析，综合可得整个算法的空间复杂度为。$O(n2)+O(n \cdot M)$ $O(n^2) + O(nm)$，在$m = n$。时就是$O(n^2)$。这个空间复杂度是非常小的，所以说在数据存储上蚁群算法是非常简单的，这也是算法的优点之一。

五、详细设计

1、/*ACATSPmain.m*/

```
tic;%记录开始时间
%城市的坐标矩阵
C=[1304 2312;
   3639 1315;
   4177 2244;
   3712 1399;
   3488 1535;
   3326 1556;
   3238 1229;
   4196 1004;
   4312 790;
   4386 570;
   3007 1970;
   2562 1756;
   2788 1491;
   2381 1676;
   1332 695;
   3715 1678;
   3918 2179;
   4061 2370;
   3780 2212;
   3676 2578;
   4029 2838;
   4263 2931;
   3429 1908;
   3507 2367;
   3394 2643;
   3439 3201;
   2935 3240;
   3140 3550;
   2545 2357;
   2778 2826;
   2370 2975];
%原函数为ACATSP(C,NC_max,m,Alpha,Beta,Rho,Q)
%变量推荐取值
%m=31;%蚁群中蚂蚁的数量，当m接近或等于城市个数n时，本算法可以在最少的迭代次数内找到最优解
%Nc_max=200;%最大循环次数，即算法迭代的次数，亦即蚂蚁出动的拨数（每拨蚂蚁的数量当然都是m）
%alpha=1;%蚂蚁在运动过程中所积累信息（即信息素）在蚂蚁选择路径时的相对重要程度，alpha过大时，算法迭代到一定代数后将出现停滞现象
%beta=5;%启发式因子在蚂蚁选择路径时的相对重要程度
%rho=0.5;%0<rho<1,表示路径上信息素的衰减系数（亦称挥发系数、蒸发系数），1-rho表示信息素的持久性系数
%Q=100;%蚂蚁释放的信息素量，对本算法的性能影响不大
%推荐使用ACATSP(C,31,200,1,5,0.5,100);
ACATSP(C,31,200,1,5,0.5,100);
toc;%记录结束时间
tic-toc%打印程序消耗时间
```

2、/*ACATSP.m*/

（1）函数定义

```
function [R_best,L_best,L_ave,Shortest_Route,Shortest_Length]=ACATSP(C,NC_max,m,Alpha,Beta,Rho,Q)
```

（2）变量初始化

```
%%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

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);       %各迭代最佳路线
L_best=inf.*ones(NC_max,1);   %各迭代最佳路线的长度
%计算城市间距离
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
```

（3）将 m 只蚂蚁放到 n 个城市上

```
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)) %ceil将m/n的结果向正无穷方向取整
        Randpos=[Randpos,randperm(n)];%随机分布在n个城市上，即在1-n的范围内，取m个随机整数，并且保证1-n范围内的整数至少出现一次
    end
    Tabu(:,1)=(Randpos(1,1:m))';   %将蚂蚁放在城市上之后的禁忌表，第i行表示第i只蚂蚁，第i行第一列元素表示第i只蚂蚁所在的初始城市
```

（4）蚂蚁按概率函数选择下一座城市，直到有蚂蚁访问完所有城市。

```
%%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));  %已访问的城市矩阵
        J=zeros(1,(n-j+1));       %定义待访问的城市矩阵
        P=J;                      %定义待访问城市的访问概率矩阵
        Jc=1;
        for k=1:n                 %for循环用于求待访问的城市。
            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
        %Pcum(find(isnan(Pcum)==1))=1;
        to_visit=J(Select(1));
        Tabu(i,j)=to_visit;
    end
```

```
end
if NC>=2
Tabu(1,:)=R_best(NC-1,:);
end
```

（5）计算各蚂蚁经过的路径长度，记录当前迭代次数最优解。

```
%%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
```

（6）对路径上的信息素浓度进行更新。

```
%%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
```

（7）输出结果。

```
%%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
title('最短距离')      %标题
```

3、画路线函数

```
%画路线图的子函数
function DrawRoute(C,R)
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 ')
```

六、测试数据及其结果分析

1、测试数据

31 个城市的坐标矩阵为：

```
C=[1304 2312;
   3639 1315;
   4177 2244;
   3712 1399;
   3488 1535;
   3326 1556;
   3238 1229;
   4196 1004;
   4312 790;
   4386 570;
   3007 1970;
   2562 1756;
   2788 1491;
   2381 1676;
   1332 695;
   3715 1678;
   3918 2179;
   4061 2370;
   3780 2212;
   3676 2578;
   4029 2838;
   4263 2931;
   3429 1908;
   3507 2367;
   3394 2643;
   3439 3201;
   2935 3240;
   3140 3550;
   2545 2357;
   2778 2826;
   2370 2975];
```

2、结果分析

我们从网络上选取 TSP 问题，用 Matlab 语言编程进行测试，对基本的蚁群算法进行了大量的实验。在基本蚁群算法中所选取的算法参数为:蚂蚁数量 m=31，循环次数 Nc_max=200，信息素重要程度 alpha=1，启发式因子 beta=5，信息衰减速度 rho=0.5，信息素总量 Q=100。所得实验结果如图 6-1 和图 6-2 所示：

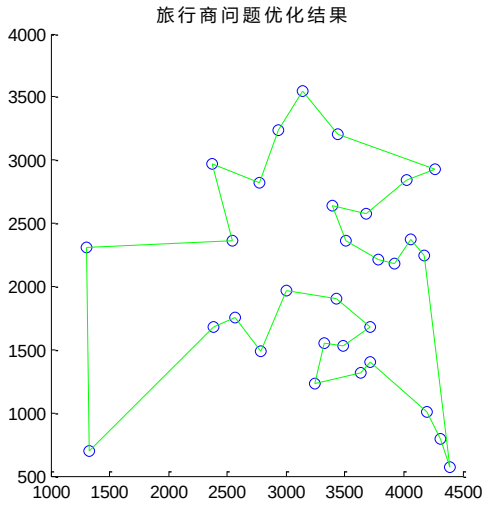

图 6-1 最优路线

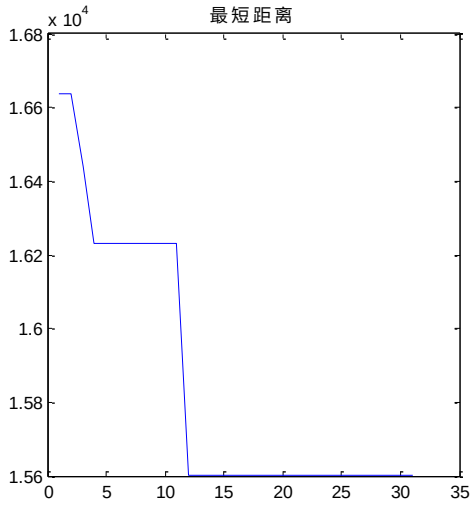

图 6-2 最短距离迭代图

根据数学推导，从图 6-1 可以看出，所得答案十分靠近最优解，说明此算法良好的解决了 TSP 问题；图 6-2 则给出了问题的收敛曲线，显然，对于基本蚁群算法，随着迭代的进行，算法的确进入了早熟停滞状态，说明迭代次数较少，可以令人接受。

运行结果如图 6-3 所示，最后给出了最优路径以及最优路径长度，最优路径所经过的城市编号分别为 14-12-13-11-23-16-5-6-7-2-4-8-9-10-19-24-25-20-21-22-26-27-30-31-29-1-15-14，最优路径长度为1.5602×10^4，运行时间大约为 50 秒。

```
>> ACATSPmain

Shortest_Route =

  Columns 1 through 17

    14    12    13    11    23    16     5     6     7     2     4     8     9    10

  Columns 18 through 31

    19    24    25    20    21    22    26    28    27    30    31    29     1    15


Shortest_Length =

   1.5602e+04

Elapsed time is 50.794261 seconds.
```

图 6-3 运行结果展示

七、调试过程中的问题

1、出现明显不是最优解的情况。一开始程序陷入局部最优解，因为路线出现交叉的情况，根据数学推理此解一定不是最优解。根据多次排查，我们发现此问题是由于参数设置不当导致的，由此我们调研了大量文献，找到了适合的参数，并解决了问题。

2、蚂蚁分布的问题。虽然蚂蚁可以随机分布在各个城市上，但是可能会出现某些城市有多只蚂蚁，有些城市没有蚂蚁的情况，这样便造成可能没有找到最优解的情况，考虑到这个问题，我们用 Randpos=[Randpos,randperm(n)]此条语句保证蚂蚁随机分布在 n 个城市上，并且蚂蚁分布相对均匀。

3、作图问题。一开始尝试用 c 语言编写代码，因为 c 语言相对于其他高级语言运行速度较快，但是 c 语言画图需要调用包，而且使用封装好的函数想做出比较好的图也比较麻烦，因此我们转而使用 Matlab，作图方便美观。

八、课程设计总结

在为期一周的网络软件设计实验周中，我们小组完成了所选项目，不仅实现了课题的基本要求，还做了相关延伸探究。在完成任务的过程中，我们查阅的大量资料，学习了算法的原理、性质、流程等等，并根据 c 语言的参考代码编写了 matlab 的代码，调试中遇到了很多问题，这只能一一排查解决。此次实验提高了我对资料的收集、整合、学习能力，也提高了我的编程能力、实践能力，但是依然还有待提高。同时对于大量代码的逻辑调用处理，锻炼了我的逻辑思维和耐心。

团队中我们分工合作，发挥了团队的协作能力。每个人都了解我们组整体的目的和计划，但是对于具体细节每个人都有专攻。由于个人能力限制，疏漏之处在所难免，敬请老师批评指正。