2020 年湖南省研究生数学建模竞赛

竞赛承诺书

我们仔细阅读了湖南省研究生数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：

我们的参赛报名号为（如果组委会设置报名号的话）：

所属学校（请填写完整的全名）：

参赛队员（打印并签名）：1. 范会迎

2. 刘卓群

3. 乔兴涵

指导教师或指导教师组负责人 （打印并签名）：

日期： 2020 年 11 月 30 日

评阅编号（由组委会评阅前进行编号）：

定向越野比赛的路线设计优化研究

摘要

定向越野路线设计问题属于多约束条件下的路径规划问题，本文先行选取了必经检查点和终点，之后对总时间和总长度、总爬高量和路段重复度计算后进行约束，筛选得到可行路径。最后以出发间隔为变量，动态模拟并观察计算路径选手相遇情况和检查点拥挤程度，调整各路径合适的出发间隔，得到路线设计方案。

对于问题一，本文构建了**基于改进蚁群算法的路径选择模型**和**多约束规划模型**，构建思路为：首先依据********确定检查点和终点及其在路径中的位置，然后分段构建改进蚁群算法模型，组合出所有可能的路径，并依据题目添加约束条件，构建多约束规划模型，筛选出符合要求的路径。最后针对选手相遇和参赛拥挤度问题，调整出发间隔，最终确定合理的路径设计。

对于问题二，重新计算从起点到各检查点的路径长度及用时，重新构建路径选择模型，并添加爬高量均匀的限制条件，重新进行筛选。

对于问题三，在给定参赛选手速度和数量的情况下，改为求最少路径问题，为避免同一路线上选手有“赶超现象”，将行进速度相近的选手划分到同一路线上，可压缩选手出发间隔，选手数量一定时，可减少路径数量。

关键词：多约束规划；改进蚁群算法；

1 问题重述

1.1 问题背景

定向越野是一项借助地形图和指北针，按规定的顺序独立地打卡若干个检查点，并以最短的时间完成任务的一项户外运动，从竞技比赛到团建活动，已受到越来越多的人们的喜爱。在定向越野比赛中，比赛路线一般由一个起点，若干个检查点和一个终点构成。一次比赛安排若干条从起点到终点的比赛路线，所有路线的起点和终点相同。每一条比赛路线可安排一个或多个参赛选手，选择同一条路线的参赛选手按照一定的时间间隔错时出发，不同路线的首发参赛选手同时出发。所有参赛选手须按照所选路线中检查点的顺序依次打卡，不同检查点的打卡难度不一样，通常与打卡点附近的地形和打卡要求有关。

1.2 问题提出

通过建模分析，解决以下问题：

问题 1: 考虑二维平面中的路线设计问题（即不考虑不同打卡点之间的高程信息），附件 1 给出了某次定向越野赛的起点、候选检查点的平面坐标和检查点的难度，假设期望本次活动中，参赛选手完成任务的平均时间不低于 2 小时，竞赛的总完成时间最长不超过 3 个小时，请为本次定向越野比赛设计比赛路线、路线的终点、必经检查点，使得尽可能多参赛选手能够加入到此次竞赛中。请给出总参赛选手数量、各条路线的参赛选手数量、检查点、总长度及总难度。

问题 2：附件 2 给出了定向越野赛的起点、候选检查点的三维坐标和检查点的难度，进一步要求每条比赛路线的爬高量尽可能均匀，请重新考虑问题 1。请给出总参赛选手数量、各条路线的参赛选手数量、检查点、总长度、总难度及爬高量，并将你们设计的比赛路线填入附件 4 的“第 2 问结果”中。

问题 3：附件 3 给出了某次定向越野比赛的起点和候选检查点三维坐标和检查点难度数据，此次活动有 120 名参赛选手参赛，通过事先对不同参赛选手的识图能力、体能等的预估，其平均行进速度存在差异，不同参赛选手的速度数据请见附件 3。本次比赛期望参赛选手平均完成时间不低于 2 小时，比赛总完成时间不超过 3.5 个小时， 请为此次活动设计最佳比赛路线，使得使用的路线尽可能少，完成的时间尽可能短。请给出各条路线的参赛选手数量、检查点、总长度、总难度及爬高量。

2 模型假设

1. 本着“安全第一”的原则，为充分发挥必经检查点的补水和医疗保障功能，假设必经检查点在各路径长度和用时的前 1/3 或后 1/3 附近。
2. 假设各选手爬高“如履平地”，爬坡和下坡不会影响选手行进速度。
3. 假设选手并不“贪婪”，每到一个点寻找下一目标点时，并不找距离最近的点，而是找完成所有打卡点总耗时最短的点。
4. 起点和终点不纳入“同时到达”人数限制考虑范围。
5. 不考虑其他地形因素，选手爬高时按空间直线行进。
6. 除起点和终点外，同一路径上相邻两检查点间距离不大于 500，不小于 50。
7. 。

3 符号说明

表 1　符号说明表

符号	意义
$v=\{v_1,v_2,\cdots,v_n\}$	检查点集
v_p、v_{q1}、v_{q2}、v_r	起始点、必经检查点、终点
$\bar{v}$	选手平均行进速度

4 问题 1 模型建立与求解

4.1 问题 1 分析

问题 1 目标是在各约束条件下，使得路径规划后能够参赛的选手数量最多。影响参赛选手数量的因素中，最直接的有路径数量和出发间隔，其他限制条件都是通过影响这两条因素，进而影响选手数量。所以问题 1 的目标是尽量规划较多的路径、尽量缩短选手出发间隔。

由于该问题约束条件过多，可考虑第一步确定下必经检查点和终点的位置，以降低建模难度，第二步分别构建起点到第一必经检查点、第一必经检查点到第二必经检查点、第二检查点到终点的改进蚁群算法模型，组合出所有可能的路径方案，第三步通过对多约束模型的建立，筛选掉不符合规则的路径方案，然后通过研究出发间隔对选手相遇情况和检查点拥挤情况的影响，确定合适的出发间隔，最后计算方案的参赛选手数量、各条路线的参赛选手数量、总长度及总难度等完成求解。

4.2 问题 1 模型建立

4.2.1 确定必经检查点和终点位置。

4.2.2 建立改进蚁群算法模型。

蚁群算法(ACA)是意大利学者 MarcoDorigo 等提出的一种仿生寻优算法。算法主要由选择策略、信息素更新和搜索算法组成。算法原理是对蚁群协作觅食过程的模拟，觅食中，蚂蚁会选择信息素浓的路径，同时释放信息素，信息素也会随着时间的延长而挥发，最后使得最短路径上的信息素不断得到强化，从而找到最短路径。其基本方法如下：假设有 m 个城市，n 只蚂蚁，$n_{ij}=\dfrac{1}{d_{ij}}$（可见度），d_{ij} 是城市 i 到城市 j 的距离；τ_{ij} 为 t 时刻 i 和 j 间的信息量；$\Delta\tau_{ij}^{k}$ 为蚂蚁 k 访问 (i,j) 释放的信息量；$p_{ij}^{k}(t)$ 为蚂蚁 k 由 i 向 j 的转移概率。初始时刻 $\tau_{ij}(0)=C$，蚂蚁 k 按照公式(*)由 i 选择 j，当所有蚂蚁完成周游，环路上的信息素按公式(4-2)更新，最后计算每只蚂蚁走过的路径长度，保存最短路径。

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{r\notin tabu_k}[\tau_{ir}]^{\alpha}[\eta_{ir}]^{\beta}}, j\notin tabu_k\\ 0, otherwise\end{cases} \tag{4-1}$$

其中， $tabu_k$ 为蚂蚁 k 已访问的城市集和， α 和 β 为信息量和自启发的权重。

$$\begin{aligned}\tau_{ij}(t+1)&=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t)\\ \tau_{ij}(t+1)&=(1-\rho)\tau_{ij}(t)+\rho\Delta\tau_{ij}(t)\\ \Delta\tau_{ij}&=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\end{aligned} \tag{4-2}$$

其中， ρ 为挥发系数， Q 为常数， L^k 为蚂蚁 k 所走路径的长度。在蚁周模型中，信息素的更新由以下公式给出：

$$\Delta\tau_{ij}^{k}(t,t+n)=\begin{cases}\dfrac{Q}{L_k} & \text{如果蚂蚁k在本次循环中经过路径 } (i,j)\\ 0 & \text{oherwise}\end{cases} \tag{4-3}$$

其中， L_k 是蚂蚁 k 在本次循环中所走的路径长度。

针对本次问题中的多约束优化问题，对蚁群算法改进如下：

(1) 设置预设的检查点数量、路线长度、难度、以及总时间消耗的期望值；根据生成路线与期望的偏差得到个评价指标 *point* ，设置阈值 *Th* 。

(2) 改进算法选择概率，使得必经点和终点被优先选择。

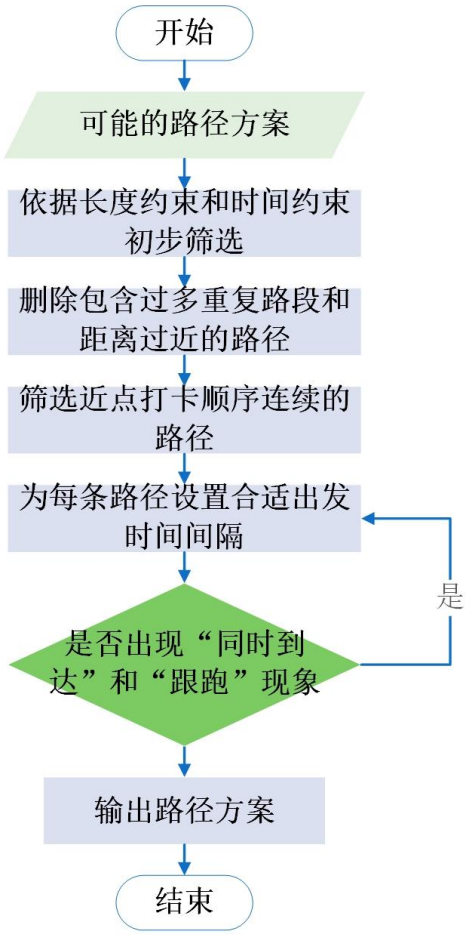
$$p_{ij}^{k}(t)=\begin{cases}\omega_j\dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{r\notin tabu_k}[\tau_{ir}]^{\alpha}[\eta_{ir}]^{\beta}}, j\notin tabu_k\\ 0, otherwise\end{cases}$$ ，其中 ω_j 为选择点

(3) 改进算法停止条件，让蚂蚁按选择概率寻找点位，如果找到终点，但是总时间小于 120min 或者检查点的总数小于 20 个，就跳出循环，算法停止；

(4) 对于某一检查点，会优先考虑选择距离最近的检查点，但是距离不能小于 100m；为了防止算法陷入局部最优，将信息素的浓度限定在[mintau,maxtau]范围之内。

4.2.3 建立多约束规划模型。

多约束规划模型建模思路如下：



图*–* 多约束规划建模流程图

4.2.2 节得到的 N 条路径规划矩阵为：

$$D_{N\times M}=\begin{bmatrix} v_p & v_i & \cdots & v_r \\ v_p & v_j & \cdots & v_r \\ \vdots & \vdots & \cdots & \vdots \\ v_p & v_k & \cdots & 0 \end{bmatrix} \tag{4-4}$$

其中每个行向量代表一个路径，行向量中的点为该路径按顺序排列的检查点。则路径难度矩阵为：

$$Dt_{N\times M}=\begin{bmatrix} t_p & t_i & \cdots & t_r \\ t_p & t_j & \cdots & t_r \\ \vdots & \vdots & \cdots & \vdots \\ t_p & t_k & \cdots & 0 \end{bmatrix} \tag{4-5}$$

每两个检查点间距离为：

$$d_{n\times n}=\begin{bmatrix}0 & d_{12} & \cdots & d_{1n}\\ d_{21} & 0 & \cdots & d_{2n}\\ \vdots & \vdots & \cdots & \vdots\\ d_{n1} & d_{n2} & \cdots & 0\end{bmatrix} \tag{4-6}$$

则路径下每点与前一点距离矩阵为：

$$Dt_{N\times M}=\begin{bmatrix}0 & d_{pi} & \cdots & d_{rx}\\ 0 & d_{pj} & \cdots & d_{ry}\\ \vdots & \vdots & \cdots & \vdots\\ 0 & d_{pk} & \cdots & 0\end{bmatrix} \tag{4-7}$$

依据题目要求建立约束条件：

$$\begin{cases}\sum_{j=1}^{M} Dd_{i\times j}\in l_{min1},l_{max2}\\ \sum_{j=1}^{M} Dt_{i\times j}\in t_{min1},t_{max2}\\ \sum_{j=1}^{M} Dt_{i\times j}+Dd_{i\times j}/\bar{\nu}\in t_{min3},t_{max4}\end{cases} \tag{4-8}$$

其中 l_{min1},l_{max2} 为约束的总路径长度区间，t_{min1},t_{max2} 为约束的路径上检查点总难度区间，t_{min3},t_{max4} 为约束的路径完成任务总时间。

继续进行如下约束：

1.某一路径下，$d\ D\ i,j\ ,D\ i,j+1\ <d\ D\ i,j\ ,D\ i,j+2$ ，即选手不会绕远打卡，保证设计路径时临近打卡点顺序连续。

2.$D_{N\times M}$ 的所有行向量的二维子向量中，为相同向量或相反向量的不超过 3 个，即各路径正向和反向重复的路段数量和不超过 3 个。

3.两路径相聚最近距离不小于 20m。

4.除起点和终点外，在某一检查点的所有首发选手数量不超过 5 人。

由以上约束层层筛选得到可行的路径，即为最终的确定路径。然后通过调整个路径上选手出发间隔，避免选手出现过近或者同一个检查点超过 5 人的情况，最后确定合适的出发间隔，拟定合理的路线设计方案。

4.3 问题 1 模型求解

通过 4.2.2 模型的求解，得到可能的路径结果如下：

5 问题 2 模型建立与求解

5.1 问题 2 分析

问题 2 相比于问题 1，所给信息增加了各检查点的 Z 坐标，要求每条比赛路线的总爬高量尽可能均匀。受到各点 Z 坐标影响而产生变化的变量有：两检查点间的距离、路径总长度、路径任务总时间。计算更新这些变量后，再用 4.2 节的方法建模，然后增加约束条件：$\sum_{j=1}^{M} Dh_{i\times j} \in h_{min1}, h_{max2}$ ，获得满足问题 1 要求且满足爬高量均匀的路径，最后调整出发间隔，避免“跟跑”和“同时到达”，确定最终的路径方案。

5.2 问题 2 模型建立

问题 2 建模过程与问题 1 大体相同，计算时更新 $d_{n\times n}$ 、$Dt_{N\times M}$ 即可，然后多约束规划模型中增加约束 $\sum_{j=1}^{M} Dh_{i\times j} \in h_{min1}, h_{max2}$ 。

5.3 问题 2 模型求解

6 问题 3 模型建立与求解

6.1 问题 3 分析

问题 3 模型建立流程如下图。

6.2 问题 3 模型建立

6.3 问题 3 模型求解

7 模型优势与不足

7.1 模型优势

通过对必经检查点和终点的先行选取，大大降低建模和仿真复杂度。并通过比较当选择不同的检查点和终点时可行的路径数量，使该选取方案更加具有说服性。

通过人为添加若干约束，使模型求解过程更贴近题目要求和实际情况，

7.2 模型不足

参考文献

[1] 冯禹．基于改进蚁群算法的定向问题研究[J]．物流工程与管理, 2013, 35(9) : 84-85．
[2] 杨理云. 基于蚁群算法的定向运动问题求解研究[J]. 计算机工程与设计，2009，30(10) : 2464-2466.
[3] 陈迎欣. 基于改进蚁群算法的车辆路径优化问题研究[J]. 计算机应用研究，2012, 29 (6) : 2032 －2034
[4] 王书勤, 黄茜.军事定向越野路径优化问题建模及混合蚁群算法求解[J]. 运筹与管理，2018, 27(4):107-111.

附录 1

```
clear all;
A= xlsread('G:\课件\竞赛\湖南省研究生建模\program\附件 1.xlsx','候选检查点
');
load('lujing.mat')
%     save lujing.mat mayroute1
lujing=route;
x=A(:,2);y=A(:,3);
%     z=A(:,4);%高程差
jianchadian=A(:,1);
zhongdian=A(:,end);
% bjjcd1=;
% bjjcd2=;
% bjjcdshijianqujian1=[];
% bjjcdchangdunqujian1=[];
% bjjcdshijianqujian2=[];
% bjjcdchangdunqujian2=[];

lujingdianshu=size(lujing,2);
changduqujian=[6000 8000];%路径总长度区间
lujinghaoshiqujian=[120 130];%路径总耗时区间
jianchadianhaoshiqujian=[30 50];%检查点总耗时区间
pagaolianghequjian=[20 40];%爬高量和区间
%% 计算任意两点间距离
dianshu=size(A,1);
d=zeros(dianshu,dianshu);%任意两点间距离
for i=1:dianshu
    for j=1:dianshu
        d(i,j)=norm([x(i),y(i)]-[x(j),y(j)]);
%         d(i,j)=norm([x(i),y(i),z(i)]-[x(j),y(j),z(i)])
    end
end
%% 筛选后一点距离
% for i=3:lujingdianshu
%     for k=1:i-1
%         for j=size(lujing,1):-1:1
%
if(d(lujing(j,i),lujing(j,i-1))>500||d(lujing(j,i),lujing(j,k))==0||...
%
d(lujing(j,i),lujing(j,i-1))<50||d(lujing(j,i),lujing(j,i-2))<d(lujing(j,i-
1),lujing(j,i-2)))
%                 lujing(j,:)=[];
%             end
%         end
```

```
%     end
% end
%% 计算、筛选爬高量
%     [r,l]=size(lujing);
%     gedianpagaoliang=zeros(r,l);  %爬高量
%     for i=1:r
%         for j=2:l
%             if(z(lujing(i,j))>z(lujing(i,j-1)))
%             gedianpagaoliang(i,j)=z(lujing(i,j))-z(lujing(i,j-1));
%             end
%         end
%     end
%     pagaolianghe=sum(gedianpagaoliang,2);
%     for j=size(lujing,1):-1:1
%
if(pagaolianghe(j)>pagaolianghequjian(2)||pagaolianghe(j)<pagaolianghequjia
n(1))
%             lujing(j,:)=[];
%             gedianpagaoliang(j,:)=[];
%             pagaolianghe(j,:)=[];
%         end
%     end
%% 计算路程和耗时
%     [r,l]=size(lujing);
%     lujingjiange=zeros(r,l);  %路径上某点与上一点路程
%     time_lujing=zeros(r,l); %路径上某点与上一点路程耗时
%     time_jianchadian=zeros(r,l); %路径上各点耗时
%     jianchadianzongshijian=zeros(r,l);%路径上所有检查点总耗时
%     lujingzongchang=zeros(r,l); %路径上某点之前总路程
%     lujingzongshijian=zeros(r,l); %路径上到某点总耗时
% for i=1:lujingdianshu
%     for m=1:r
%         for n=1:i
%             time_jianchadian(m,n)=A((lujing(m,n)),end);
%             jianchadianzongshijian(m,n)=sum(time_jianchadian(m,1:n));
%             if n>1
%                 lujingjiange(m,n)=d(lujing(m,n),lujing(m,n-1));
%                 time_lujing(m,n)=d(lujing(m,n),lujing(m,n-1))/100;
%                 lujingzongchang(m,n)=sum(lujingjiange(m,1:n));
%
lujingzongshijian(m,n)=sum(time_lujing(m,1:n))+jianchadianzongshijian(m,n);
%             end
%         end
%     end
```

```
% end
%% 筛选路径和耗时不符合要求的点
%     for j=size(lujing,1):-1:1
%
if(lujingzongchang(j,lujingdianshu)>changduqujian(2)||lujingzongchang(j,luj
ingdianshu)<changduqujian(1)||...
%
lujingzongshijian(j,lujingdianshu)>lujinghaoshiqujian(2)||lujingzongshijian
(j,lujingdianshu)<lujinghaoshiqujian(1)||...
%
jianchadianzongshijian(j,lujingdianshu)>jianchadianhaoshiqujian(2)||jiancha
dianzongshijian(j,lujingdianshu)<jianchadianhaoshiqujian(1))
%             lujing(j,:)=[];
% %             gedianpagaoliang(j,:)=[];
% %             pagaolianghe(j,:)=[];
%             lujingzongchang(j,:)=[];
%             lujingzongshijian(j,:)=[];
%             time_jianchadian(j,:)=[];
%             time_lujing(j,:)=[];
%             lujingjiange(j,:)=[];
%             jianchadianzongshijian(j,:)=[];
%         end
%     end
%% 筛选掉必经检查点不在1/3处的路径
%         for j=size(lujing,1):-1:1
%             b1=find(lujing(j,:)==bjjcd1);
%             b2=find(lujing(j,:)==bjjcd2);
%
if(lujingzongchang(j,b1)>bjjcdchangdunqujian2(2)||lujingzongchang(j,b1)<bjj
cdchangdunqujian1(1)||...
%
(lujingzongchang(j,b1)>bjjcdchangdunqujian1(2)&&lujingzongchang(j,b1)<bjjcd
changdunqujian2(1))||...
%
lujingzongshijian(j,b1)>bjjcdshijianqujian2(2)||lujingzongshijian(j,b1)<bjj
cdshijianqujian1(1)||...
%
(lujingzongshijian(j,b1)>bjjcdshijianqujian1(2)&&lujingzongshijian(j,b1)<bj
jcdshijianqujian2(1))||...
%
lujingzongchang(j,b2)>bjjcdchangdunqujian2(2)||lujingzongchang(j,b2)<bjjcdc
hangdunqujian1(1)||...
%
(lujingzongchang(j,b2)>bjjcdchangdunqujian1(2)&&lujingzongchang(j,b2)<bjjcd
```

```
changdunqujian2(1))||...
%
lujingzongshijian(j,b2)>bjjcdshijianqujian2(2)||lujingzongshijian(j,b2)<bjj
cdshijianqujian1(1)||...
%
(lujingzongshijian(j,b2)>bjjcdshijianqujian1(2)&&lujingzongshijian(j,b2)<bj
jcdshijianqujian2(1)))
%              lujing(j,:)=[];
%              %              gedianpagaoliang(j,:)=[];
%              %              pagaolianghe(j,:)=[];
%              lujingzongchang(j,:)=[];
%              lujingzongshijian(j,:)=[];
%              time_jianchadian(j,:)=[];
%              time_lujing(j,:)=[];
%              lujingjiange(j,:)=[];
%              jianchadianzongshijian(j,:)=[];
%              end
%          end
%% 去掉重复路段（不允许有三段重复路段）
%          nn=0;
%          temp3=[];
%          for j=size(lujing,1):-1:1
%
%              for i=1:lujingdianshu-2
%                  temp=lujing(j,i:i+1);
%                  for jj=j-1:-1:1
%                      for ii=1:lujingdianshu-2
%                          tteemmpp=lujing(jj,ii:ii+1);
%                          ppmmeett=lujing(jj,ii+1:-1:ii);
%                          if (temp==tteemmpp)
%                              nn=nn+1;
%                          end
%                          if (temp==ppmmeett)
%                              nn=nn+1;
%                          end
%                          if nn>=3
%                              temp3=[temp3 j];
%                              nn=0;
%                          end
%                      end
%                  end
%              end
%          end
%          lujing(temp3,:)=[];
```

```
% %          gedianpagaoliang(temp3,:)=[];
% %          pagaolianghe(temp3,:)=[];
%          lujingzongchang(temp3,:)=[];
%          lujingzongshijian(temp3,:)=[];
%          time_jianchadian(temp3,:)=[];
%          time_lujing(temp3,:)=[];
%          lujingjiange(temp3,:)=[];
%          jianchadianzongshijian(temp3,:)=[];
%%  同一点人数<=5 判断
%          temp3=[];
%          for k=jianchadian'%(jianchadian~=zhongdian)'
%              [tempx,tempy]=find(lujing==k);
%              if length(tempx)>5
%                  for i=1:length(tempx)
%                                                          temp1(i,:)=[tempx(i)
lujingzongshijian(tempx(i),tempy(i))];
%                  end
%                  temp2=sortrows(temp1,2);
%                  for kk=length(temp2):-1:6
%                      if (temp2(kk,2)-temp2(kk-5,2)<1)
%                          temp3=[temp3 temp2(kk-5,1)];
%                          temp2(kk-5,:)=[];
%                      end
%                  end
%              end
%          end
%          lujing(temp3,:)=[];
% %          gedianpagaoliang(temp3,:)=[];
% %          pagaolianghe(temp3,:)=[];
%          lujingzongchang(temp3,:)=[];
%          lujingzongshijian(temp3,:)=[];
%          time_jianchadian(temp3,:)=[];
%          time_lujing(temp3,:)=[];
%          lujingjiange(temp3,:)=[];
%          jianchadianzongshijian(temp3,:)=[];
```

附录 2

```
function bestroute = antroute(locations)

[M,N] = size(locations);
% M 为问题的规模  M 个城市
distant = zeros(M,M);      % 用来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
for m=1:M
    for n=1:M
        distant(m,n) = sqrt(sum((locations(m,:)-locations(n,:)).^2));
    end
end
m = 30;   % 蚂蚁的个数    一般取 10-50
alpha = 1;   %  信息素的重要程度
beta = 5;    %  启发式英子的重要程度
rho = 0.25;    % 信息素蒸发系数
G = 30;
Q = 100; % 信息素增加系数
Eta = 1./distant;  % 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
gen = 1;
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
%   开始迭代计算
orignal_state = 1;
while gen<G
%  将 m 只蚂蚁放到 n 个城市上
%    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)];
%           %  random_pos=[1~31 + 1~31]    将每只蚂蚁放到随机的城市   在
random_pos 中随机选择 m 个数，代表蚂蚁的初始城市
%
%     end
    random_pos = orignal_state*ones(1,m);
    Tabu(:,1) = (random_pos(1,1:m))';   % 第一次迭代每只蚂蚁的禁忌表

    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
            % visited=visited(1,:);
```

```
            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
            %   这里运用轮盘赌选择方法       这里也可以选择选择概率最大的路径去走， 这里采用轮盘赌选择法。
            Pcum = cumsum(visit_P);
            selected = find(Pcum>=rand);
            to_visited = unvisited(1,selected(1,1));
            Tabu(j,i) = to_visited;  % 添加到禁忌表
        end

    end
    if gen>=2
        Tabu(1,:) = R_best(gen-1,:);
    end
    %  记录 m 只蚂蚁迭代的最佳路线
    L = zeros(1,m);
    for i=1:m
        R = Tabu(i,:);
        L(i) = distant(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)+distant(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
```

```
    %  更新信息素的值
    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);    %  m 只蚂蚁的信息素累加 这里采
用的是论文中 ant-cycle 模型
        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
    %  禁忌表清零
    Tabu = zeros(m,M);
    gen = gen+1;
end
bestroute = R_best(G-1,:);
end
```