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：B 题

我们的参赛报名号为：202018001012

所属学校：国防科技大学

参赛队员（打印并签名）：1. 罗海鹏

2. 靳增源

3. 虞朝栋

指导教师或指导教师组负责人 （打印并签名）：

日期：　　年　　月　　日

评阅编号（由组委会评阅前进行编号）：

评阅编号：
（由组委会填写）

2020年湖南省高校第五届研究生数学建模竞赛

编 号 专 用 页

评阅记录：

评阅人						
备注						

（请勿改动此页内容和格式。此编号专用页仅供评阅使用。）

题　目：基于改进遗传算法的定向越野路线规划设计

摘　要

合理的定向越野路线规划能够很好地提高参赛人员比赛体验、很好地避免比赛过程中作弊等情况发生。通过规划设计各条路线及其出发时间间隔，也可以使各条路线完成时间尽量相等、容纳尽量多的参赛人员加入比赛，或者得到最小的比赛路线保障一定人员的定向越野比赛，因此可以看出比赛路线规划很有意义。设立不同目标函数进行优化，可以得到适应不同背景下的路线规划。本题中根据题意将问题分解为必经点及终点选取、多路线规划建模和遗传算法求解三部分来完成。必经点选取着重考虑各条路线更容易地经过，在问题一二三中分别选取 6 号检查点和 64 号检查点、20 号检查点和 26 号检查点、137 号检查点和 35 号检查点为必经点；终点选取先根据题目“难度较低点”划分待选解空间，再综合考虑距离起点距离、及难度，在问题一二三中分别选取 84 号检查点、131 号检查点、36 号检查点为终点。多路线规划模型部分为题目各项规则建立不同数学模型，根据题意分别以最大化参赛人数或最小化总比赛时间为目标函数建立路线规划模型。在优化求解方面设立比赛路线，使用遗传算法优化得到对应路线数下最佳路线规划，进一步比较不同比赛路线数量下各项指标得到全局最优解。问题一求解得到当比赛路线为 10 条时，最大参赛人数为 96 人；问题二求解得到当比赛路线为 14 条时，最大参赛人数为 140 人；问题三为双目标优化，要求完成时间短、比赛路线少，求解过程中考虑时间及比赛路线增加都带来举办比赛代价，将二者加权和以总代价最小为最优，还可以通过更改权值适应不同背景进行寻优，折中选取比赛路线数和比赛完成时间权值分别为 0.6、0.4，得到当比赛路线数为 10 条，总完成时间为 197.54 分钟时总代价最小。

关键词：定向越野　多路线规划　多目标优化　遗传算法

一、问题重述与分析

合理的定向越野路线规划能够很好地提高参赛人员比赛体验、很好地避免比赛过程中作弊等情况发生；也能更有效地利用各项资源，用固有的条件满足更多人员参加比赛，或者用最少的资源满足人们的比赛需求，因此定向越野比赛中路线规划很有必要。

本题题目给出了一定区域内各个检查点相关数据以及路线设计规则，问题一、二要求规划路线设计使得规定时间内参加比赛的选手人数尽可能多。问题三则是给出参加比赛总人数，要求规划设计路线使得比赛路线尽可能少以及总完成时间尽可能短。综合考虑本题要求，可以将问题分为“必经点选取、终点选取”及“路线规划”两部分来完成并简化问题。

其中，“必经点选取、终点选取”部分，根据题目要求，必经点为各条路线都通过的检查点。因此在必经点选取策略上更多考虑区域内所有检查点能较为方便的与必经点相连，用必经点到其他检查点距离之和表示代价，要求代价最小。并且考虑到要求选取两个必经点，两个必经点距离越近越不合理，用两个必经点间距离表示代价，距离越近，代价越大。最后综合考虑，以全局代价最小为目标，选取得到必经检查点。终点选取题目只简单的要求“选择难度较小的检查点”即可，但进一步考虑到为了满足路线规划其他原则，使路段重复率尽量低，因此还需要起点终点距离尽量远来满足。最后，综合考虑难度及距离起点距离进行终点检查点的选取。

多路线规划部分，考虑到路线规划往往实现的是路线的规划及参赛人员的规划，很难实现比赛路线数目的规划，因此可以考虑先人为设定比赛路线数目，然后根据题目约束及不同问题的优化目标进行优化，得到各自情况下的最佳结果。然后再比较得到全局最优解。优化算法的选取，通过参考相关文献，发现遗传算法、模拟退火算法、粒子群算法等都有使用，但遗传算法使用更多而且也很利于本题的多路线规划问题求解，因此最后采取遗传算法，针对本问题具体情况进行改进用于模型的优化求解。

问题一、二要求规划路线设计使得规定时间内参加比赛的选手人数尽可能多，这就决定了在路线规划和间隔选取时，需要折中考虑并优化；问题三则指定了参赛人员数量，希望通过对路线和人员组别、次序分配方案的双最优规划，来实现更少的路线数量和更短的比赛完成时间。与前两问相区别的是，不同参赛人员速度水平存在差异，在规划中使得“同一时间到达同一检查点人员应不超过 5 人”约束的实现更加困难，这也就意味着在对人员进行分配时，需要着重考虑速度这个差异性因素，通过达到各组速度的相对平衡来既保证不会发生各组速度差异性过大导致“水桶效应”使得最终完成时间只有速度最慢的一组决定，也保证间隔的安排可以在时间的约束上得到一定程度的简化。

二、模型假设

除了题目中的假设外， 模型分析和建立求解依赖以下假设条件：

(1) 假设比赛过程中各参赛队员匀速前进；

(2) 假设没有意外事件发生，不用考虑中途停下的情况；

(3) 假设检查点难度只给比赛选手带来完成时间上的增加，在排序过程中不需要考虑检查点难度影响；

(4) 假设连续几个检查点相距较近不会带来影响，在路径优化过程中只需要考虑不同路线的总长度、难度、爬高等方面约束；
(5) 假设不同路段上参赛人员不需要考虑距离过近问题，从而可以简化问题将避免距离过近转化为出发间隔约束；
(6) 假设检查点没有人员限制，只有“同时到达不超过 5 人”限制，前面到达某检查点的参赛人员还未完成打卡并不会给新到达人员带来影响。

三、参数符号说明

表 1 参数符号说明

参数	含义
n_i	检查点 i
K	路线总数
N	检查点总数
x_i	检查点 i 的横坐标
y_i	检查点 i 的纵坐标
z_i	检查点 i 的高程
n_{b1}	第 1 个必经检查点
n_{b2}	第 2 个必经检查点
n_s	起点
n_e	终点
d_{ij}	检查点 i 和 j 之间的直线距离
$l_{i,j}$	连接检查点 i 和 j 的线段
$h_{l_{i,j}}$	从点 i 连接到 j 的线段的爬高量
C_i	第 i 条路径检查点的无序集合
a_{C_i}	集合 C_i 的检查点数目

S_i	路径 i 的检查点的有序集合
L_i	路径 i 依次经过路段有序集合
D_i	路径 i 的总长度
δ_{dth}	路径总长度方差阈值
DF_i	路径 i 的总难度
df_i	检查点 i 的难度
δ_{DFth}	路径总难度方差阈值
$v_{L_i,j}$	路线 i 中第 j 个人的行进速度
δ_{vth}	各路线平均速度的方差阈值
p_{L_i}	路线 L_i 被分配的选手数
H_i	路径 i 产生的爬高量
η	路径重复长度占比
Δt_i	第 i 条路线的出发时间间隔
T_{endi}	第 i 条路线的选手最终完成时间
$u_{i,k}^t$	在时间区间 $[t,t+1]$ 内路线 k 是否有参赛人员到达检查点 i 的节点状态
g_i	第 i 条路线的出发组数
P	参加比赛的总人数

四、模型建立与求解

4.1 问题一模型建立与求解

问题一考虑二维平面中路线设计问题，比赛路线由一个起点、一个终点、若干个中间检查点构成，其中起点固定，有两个中间检查点所有路线必须经过。如下图所示：

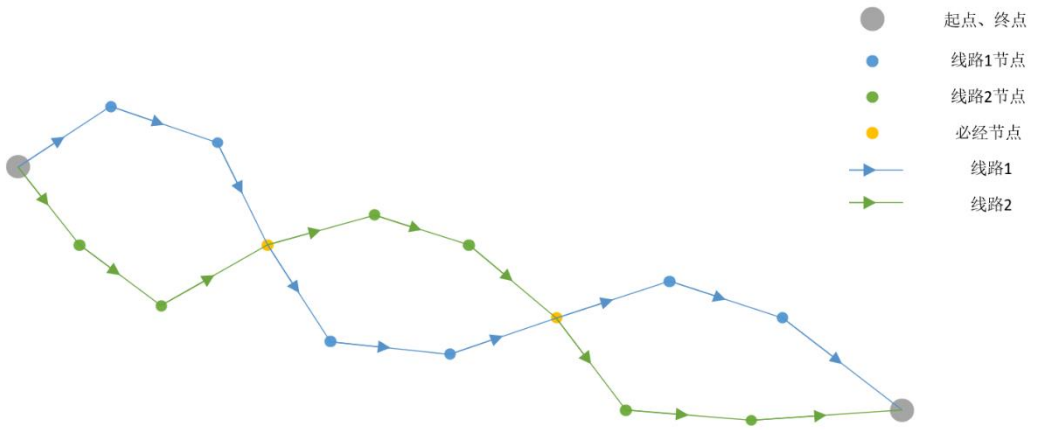


图 1　越野路线示意图

因此这个问题可以先确定这两个中间检查点和终点，然后根据约束条件完成一个路径规划问题。通过这两部分实现问题建模与求解。

图 2 模型简化图

4.1.1 必经点、终点选取策略

(一) 必经点选取策略

每条路线中检查点数量不低于 20 个，在这些检查点中设置两个是所有路线必须经过的点，常用于补给水和医疗保障，并且及时了解参赛选手情况避免出现伤病等意外情况。

这两个点是所有路线都必须经过的点，因此应该安排在整个定向越野比赛场地较为中间的部分，能够让所有路线较为容易地经过它，在数学模型体现为这两个必经点 $n_i = n_{b1}$， $n_j = n_{b2}$ 到其他检查点的距离总和应该尽可能地小，距离总和用下式表示：

$$d_{st1} = \sum_{\substack{k=1 \\ k \neq i,j}}^{N} d_{ik} + \sum_{\substack{k=1 \\ k \neq i,j}}^{N} d_{jk} \tag{1}$$

考虑实际情况，两个必经检查点距离不能太近，一般设置在比赛路线 1/3、2/3 处比较好，这里可以为必经检查点过近设置距离惩罚函数，距离越近，惩罚越大，用下式表示：

$$d_{st2} = -\alpha \times d_{ij} \tag{2}$$

其中 α 为常数，通过改变 α 的值可以改变惩罚大小产生更合理的必经点分布。

综合上面两方面的考虑，可以把两个必经点选取问题用如下模型表示：

$$\min d_{st}$$

$$s.t.\begin{cases} d_{st} = d_{st1} + d_{st2} \\ d_{st1} = \sum_{\substack{k=1 \\ k\neq i,j}}^{N} d_{ik} + \sum_{\substack{k=1 \\ k\neq i,j}}^{N} d_{jk} \\ d_{st2} = -\alpha \times d_{ij} \end{cases} \tag{3}$$

在$\alpha = 10, \alpha = 150$和$\alpha = 500$时，分别得到必经点选取图如下：

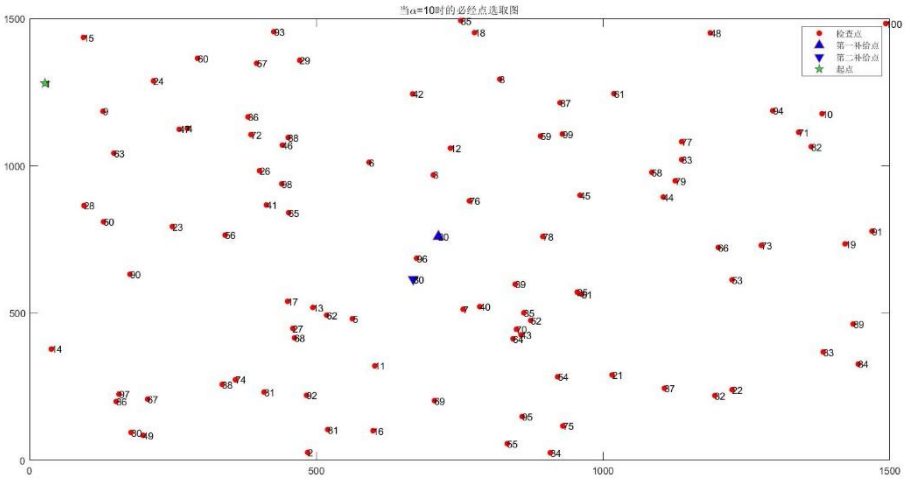


图 3 $\alpha = 10$时的必经点选取图

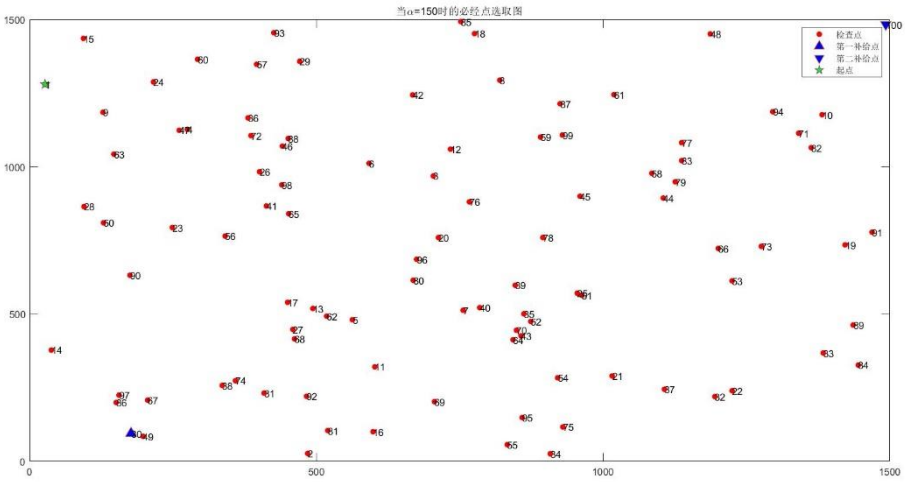


图 4 $\alpha = 150$时的必经点选取图

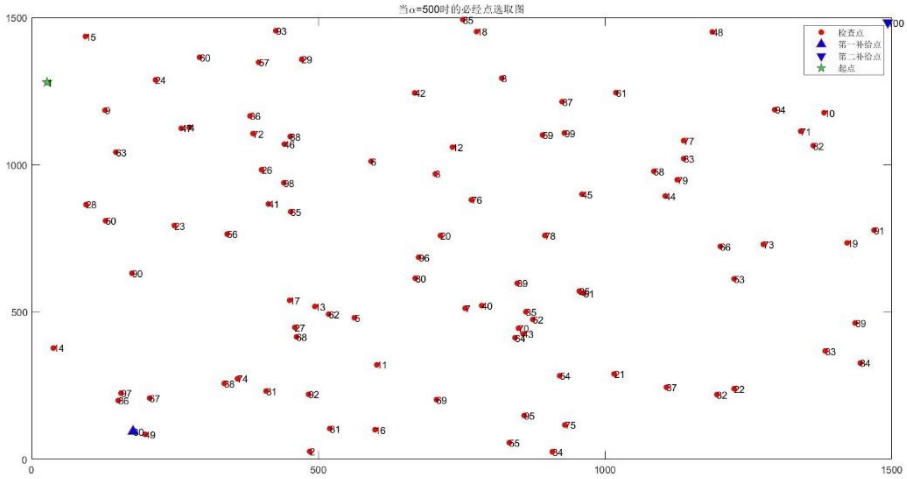


图 5 $\alpha = 500$时的必经点选取图

如图所示，改变α值可以得到区别较大的结果。改变α值改变了两个约束条件的重要性程度，当$\alpha = 0$时，两个必经点到其他点距离的和最小成为主要因素，当$\alpha = 500$时，两个必经点之间距离尽量远的条件得到满足。选取合适的α值可以得到较为合理的结果，但是考虑到人为选取的α值有包含较多主观因素，不能得到最优，因此对上式中两个距离分别归一化，去掉α值，可以得到一个较为客观的必经点选取策略，用下式表示：

$$\min d_{st}^{'}$$

$$s.t.\begin{cases} d_{st}^{'} = d_{st1}^{'} + d_{st2}^{'}; \\ d_{st1}^{'} = \dfrac{\sum\limits_{\substack{k=1 \\ k\neq i,j}}^{N} d_{ik} + \sum\limits_{\substack{k=1 \\ k\neq i,j}}^{N} d_{jk}}{2\times(N-2)}; \\ d_{st2}^{'} = -d_{ij}; \end{cases} \tag{4}$$

重新画图如下，得到较为客观的最佳必经点选取为打卡点 6 和 64。

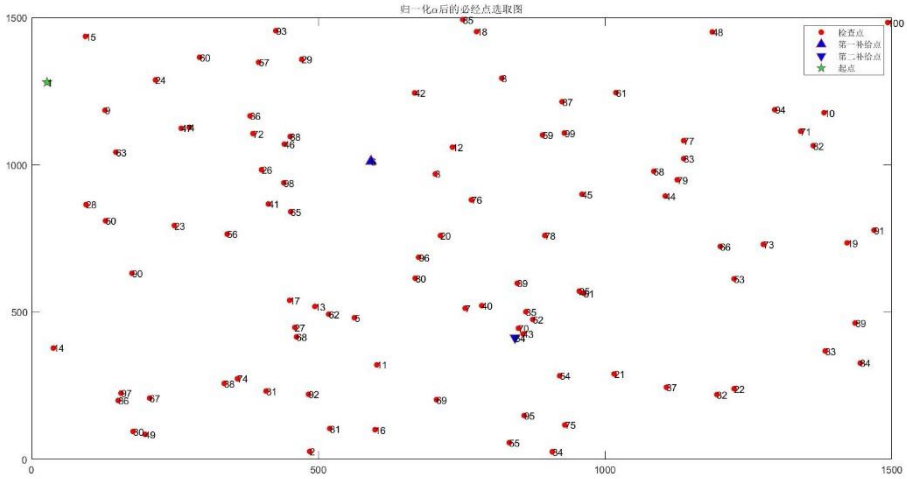


图 6 归一化α后的必经点选取图

(二) 终点选取策略

在终点 N_e 选取策略上，题目仅要求终点选取在难度较低的点，根据附件一数据可以看出各打卡点难度共有“1、2、3、4”多种情况，我们规定“1、2”两种难度的点为难度较低的点。可得待选终点解空间：

$$n_e \in C_e \quad s.t. \quad df_i \le 2, \forall n_i \in C_e; \tag{5}$$

C_e 表示包含所有检查点难度为 2 以下的点的集合。

根据附件数据画各打卡点二维分布图如下所示：

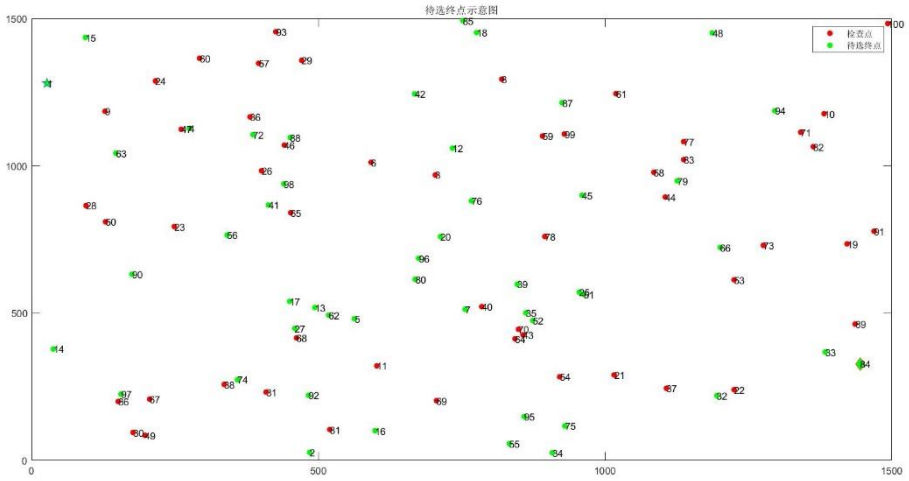


图 7 待选终点示意图

所有的检查点分布在整个区域为 1.5km*1.5km 正方形场地，问题一要求平均参赛时间不少于 2 小时，每个参赛人员速度为 6km/h。路线长、区域小，还要求重复路段尽可能少、以及同时到达同一检查点人数不超过 5 人，因此终点选取在距离起点尽可能远的地方，最终目标应该使终点距离起点最远且难度最小，即

$$\min\ d_{n_s n_e} = \begin{cases} \sqrt{\left|x_{n_s} - x_{n_e}\right|^2 + \left|y_{n_s} - y_{n_e}\right|^2} & \text{,检查点为2}D\text{坐标点时,} \\ \sqrt{\left|x_{n_s} - x_{n_e}\right|^2 + \left|y_{n_s} - y_{n_e}\right|^2 + \left|z_{n_s} - z_{n_e}\right|^2} & \text{,检查点为3}D\text{坐标点时} \end{cases} \tag{6}$$

经过综合筛选，最终得到距离最远且难度较小的终点为 84，如下图所示：

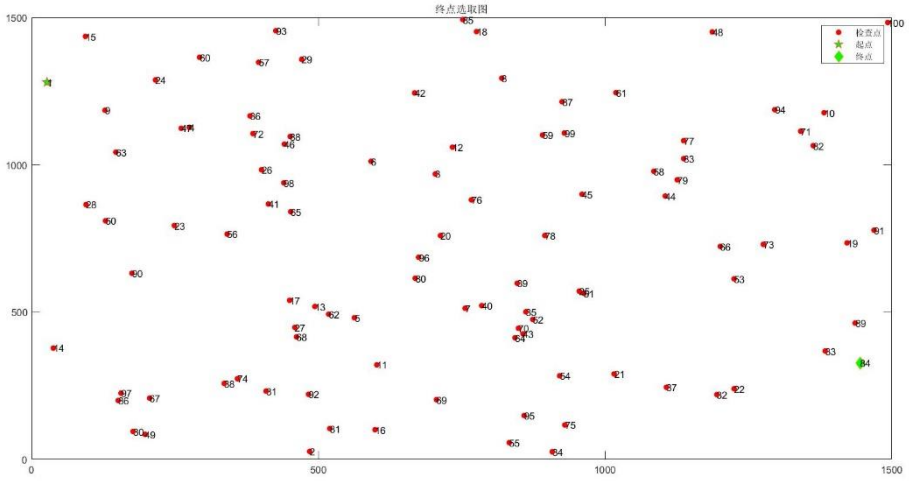


图 8 终点选取图

最终得到的总体检查点分布图如下：

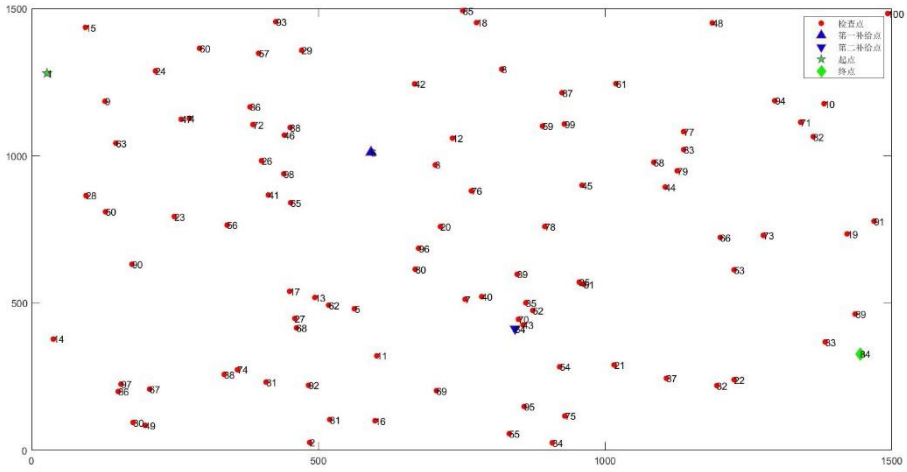


图 9 总体检查点分布图

4.1.2 路径优化模型

关于路径设计及寻优部分具体由路径优化模型及遗传算法来实现。

首先优化总目标是最大化参加比赛人数，比赛人数由比赛路线数量和同一路线出发波次决定，给定时间限制出发波次由出发时间间隔决定。要想更多人参加比赛，需要比赛路线更多以及出发间隔更短。但是因为各线路检查点数目限制、重复路段占比限制、以及在同时事件到达同一检查点人数限制，上面两个因素比赛线路数目、出发间隔本身不能很大或很小，而且相互之间还有制约，因此在建模过程中可以根据比赛路线来计算满足题目需求的最小出发时间间隔，从而得到该情况下的最大参加比赛人数。通过遗传算法等不断迭代，得到起、终点确定的多路径优化模型的全局较优解。

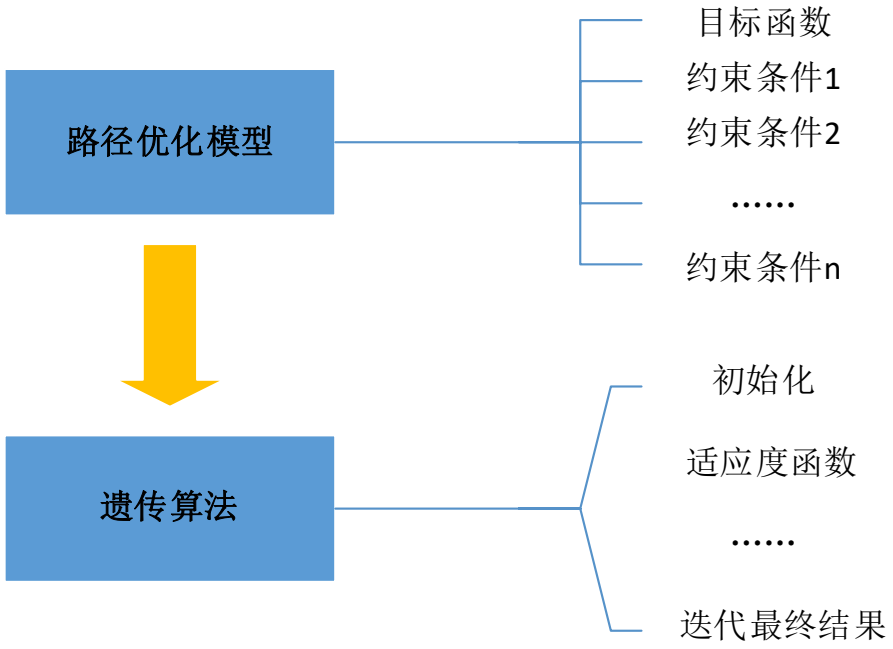


图 10 路径优化关系示意图

下面根据题目要求的路线设计原则建立优化目标函数及各约束条件。

比赛路线设计规则(1)要求不同路线总长度应尽可能平均，优化目标为“尽可能相等”难以实现，可以转化为不同路线总距离方差尽量小来简化问题。设定总距离方差阈值 δ_{dth}，要求不同路径之间的方差小于阈值。数学模型如下所示：

$$\sum_{i=1}^{k}\left(D_i-\overline{D}\right)^2<\delta_{dth}$$

$$s.t.\begin{cases}D_i=\sum\left\{d_{l_{m,n}}\right\},\forall l_{m,n}\in L_i;\\ \overline{D}=\dfrac{1}{k}\sum_{i=1}^{k}D_i\end{cases}\tag{7}$$

比赛路线设计规则(2)要求不同路线总难度尽可能均匀，与规则(1)建模方式类似，设定总难度方差阈值 δ_{DFth}，要求不同路径方差小于阈值。数学模型如下所示：

$$\sum_{i=1}^{k}\left(DF_i-\overline{DF}\right)^2<\delta_{DFth}$$

$$s.t.\begin{cases}DF_i=\sum\left\{df_{n_j}\right\},\forall n_j\in C_i;\\ \overline{DF}=\dfrac{1}{k}\sum_{i=1}^{k}DF_i;\end{cases}\tag{8}$$

比赛路线设计规则(4)要求不同路线之间应尽可能避免包含较多重复路段，这里进一步考虑到不同路段可能长度不相同，有些情况下虽然两条路线的重复路段很少甚至只有一条，但是重复长度占总长度的比例较大，也不合理。因此可以通过计算 K 条路线中重复长度占总长度的比例 η 并始终保持低于设定的重复距离阈值 η_{th} 实现约束。如下式所示：

$$\begin{cases}L_{repeat}=\bigcup_{\substack{j=1\\ j\neq i}}^{k}\left\{L_i\bigcap L_j\right\},\forall 1\le i\le K;\\ D_{repeat}=\sum\left\{d_{l'_{m,n}}\right\},\forall l'_{m,n}\in L_{repeat};\\ \eta=\dfrac{D_{repeat}}{\sum_{i=1}^{K}D_i}<\eta_{th};\end{cases}\tag{9}$$

比赛路线设计规则(4)还要求为避免出现“跟跑”现象或者相互交流，查阅资料了解到通常定向越野比赛中同一路线前后两人最好不要相遇。这里都可以建模为对各路线出发时间间隔 Δt_k 的约束：

$$\Delta t_k\ge\max\left(df_i\right),\forall 1\le k\le K,1\le i\le a_{C_i};\tag{10}$$

设算法实现过程中某次迭代时比赛路线数量为 K，其中第 i 条路径中包含的检查点由无序集合 $C_i=\left\{n_s,n_e,n_{b1},n_{b2},...\right\}$ 表示，集合中的元素表示该路线经过的检查点。a_{C_i} 表示集合 C_i 的容量，路线设计规则(5)要求每条路径包含检查点不少于 20 个，并且固定起点、终点及两个必经点：

$$\begin{cases}a_{C_i}\ge 20;\\ C_i\bigcap C_0=C_0;\end{cases}\forall 1\le i\le K\tag{11}$$

其中，$C_0=\{n_s,n_{b1},n_{b2},n_e\}$表示包含且仅包含起点、终点和必经点的点集。

比赛路线设计规则(6)要求同一路线中距离相近的检查点的打卡顺序应尽可能连续，制定检查点打卡顺序时要考虑路线的长度尽可能短。在规则(6)的约束下对表示第i条路径经过检查点的无序集合C_i内元素进行排序，第一个元素为起点，最后一个元素为终点，得到表示第i条路线依次经过检查点的有序集合S_i。设集合S_i内第j个元素用$S_i(j)$表示，满足如下式子：

$$\begin{cases} S_i(1)=n_s; \\ S_i\left(a_{C_i}\right)=n_e; \end{cases} \tag{12}$$

得到表示这K条路径依次经过检查点的有序集合$S_1\sim S_K$后也就得到K条路线各自依次经过的有序路段集合$L_1\sim L_K$。其中$L_i=\{...,l_{m,n},l_{n,p},l_{p,q},...\}$，$l_{m,n},l_{n,p},l_{p,q}$为$S_i$内连续检查点$m,n,p,q$之间的路段。

比赛路线设计规则(7)要求同时到达同一个检查点的参赛选手数量应不超过5人，“同时到达”指的是不同参赛选手到达同一检查点的时间差不超过1分钟。根据某次迭代过程中对这K条路径，设$u_{i,k}^t$表示在时间区间$[t,t+1]$内路线k是否有参赛人员到达检查点i，若有则为1，没有则为0。则有：

$$\sum_{k=1}^{K}u_{i,k}^t\le 5,\forall 1\le i\le N,t\in[0,T_{endi}]; \tag{13}$$

问题一题目规定每个运动员速度为6km/h，要求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，设3小时内第i条路线的参赛组数为g_i，则应满足如下约束：

$$\begin{cases} \dfrac{1}{K}\sum_{i=1}^{K}T_{endi}=\dfrac{1}{K}\sum_{i=1}^{K}\left[\dfrac{D_i}{v_0}+\sum\left\{df_{n_j}\right\}+(g_i-1)\Delta t_i\right]\ge 2; \\ \max T_{endi}=\max\left[\dfrac{D_i}{v_0}+\sum\left\{df_{n_j}\right\}+(g_i-1)\Delta t_i\right]\le 3, \quad \forall 1\le i\le K,\forall n_j\in C_k; \\ v_0=6\ \text{km/h} \end{cases} \tag{14}$$

4.1.3 总优化目标

综上所述，整个问题一路径优化模型流程可以用下图表示：

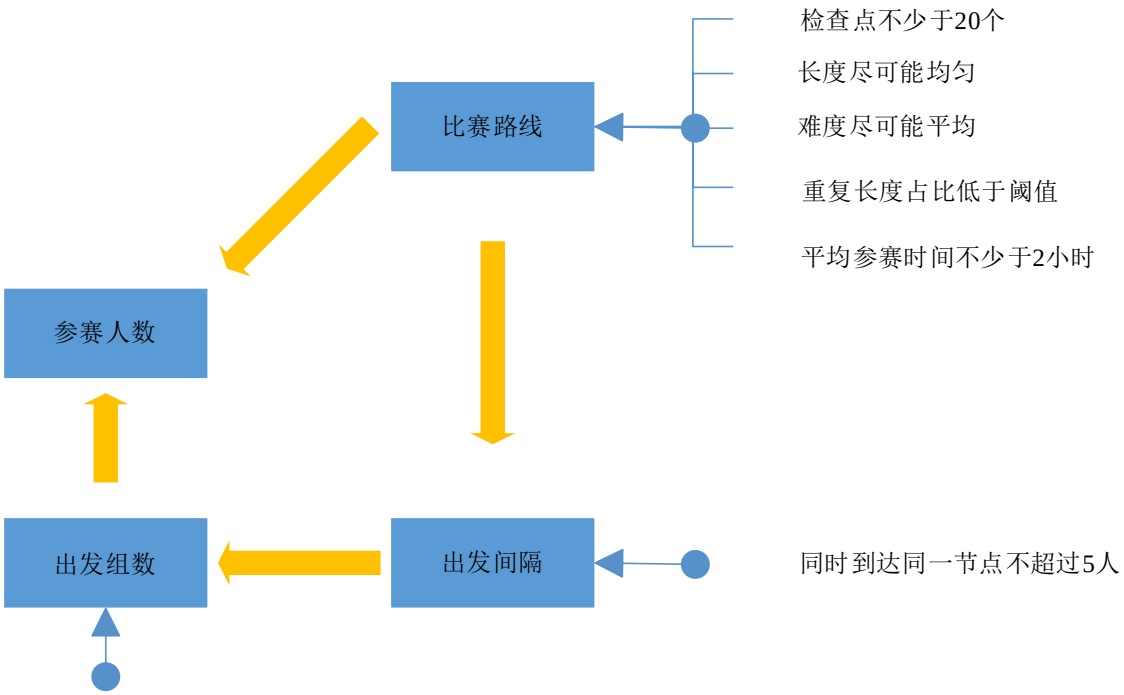


图 11 规划约束关系图

从而可以得到总的路径优化模型目标函数及其约束条件：

$$\max P=\sum_{k=1}^{K} g_k$$

$$s.t.\begin{cases}
\sum_{k=1}^{K}\left(\sum\left\{d_{l_{m,n}}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}\right)^2<\delta_{dth},\forall l_{m,n}\in L_k;① \\
\sum_{k=1}^{K}\left(\sum\left\{df_{n_j}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{df_{n_j}\right\}\right)^2<\delta_{DFth},\forall n_j\in C_k;② \\
\eta=\dfrac{\sum\left\{d_{l'_{m,n}}\right\}}{\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}}<\eta_{th},\quad \forall 1\le k\le k, l_{m,n}\in L_i, l'_{m,n}\in L_{repeat};③ \\
\Delta t_k\ge\max\left(df_{n_j}\right),\forall 1\le k\le K, n_j\in C_k;④ \\
\begin{cases}
a_{C_k}\ge 20; \\
C_k\bigcap C_0=C_0;
\end{cases}\forall 1\le k\le K;⑤ \\
\sum_{k=1}^{K}u_{i,k}^{t}\le 5,\forall 1\le i\le N, t\in[0,T_{endk}];⑥ \\
\begin{cases}
\frac{1}{K}\sum_{k=1}^{K}T_{endk}=\frac{1}{K}\sum_{k=1}^{K}\left[\frac{D_k}{v_0}+\sum\left\{df_{n_j}\right\}+(g_k-1)\Delta t_k\right]\ge 2;⑦ \\
\max T_{endk}=\max\left[\frac{D_k}{v_0}+\sum\left\{df_{n_j}\right\}+(g_k-1)\Delta t_k\right]\le 3,\quad \forall 1\le k\le K;⑧
\end{cases} \\
v_0=6\ \text{km/h}
\end{cases} \tag{15}$$

其中，式①表示不同路线距离尽可能均匀，式②表示不同路线距离尽可能均

匀，式③表示不同路线重复率尽可能少，式④表示不会发生超越交流现象，式⑤表示每条路线必须大于等于 20 个检查点，式⑥表示同时到达同一个检查点的参赛选手数量应控制在不超过 5 人，式⑦表示参赛选手完成任务的平均时间不低于 2 小时，式⑧表示竞赛的总完成时间最长不超过 3 个小时。

4.1.4 遗传算法

a) **遗传算法步骤**:

进化论的基本规则就是“自然选择，适者生存”，每个个体通过交配繁衍出后代，后代在成长过程中经过自然法则的筛选而淘汰掉部分个体，而那些拥有优良性状的个体得以存活并且通过繁衍使得优秀遗传信息得以保留和扩散。

遗传算法（Genetic Algorithm,GA）正是借鉴了这种思想。我们把生物的进化过程抽象成一个数学模型，生物进化指的是初始种群里拥有各种各样的同种生物，经过繁衍和自然选择得到的最终种群里，每个个体或者说绝大部分是最适应环境的优秀个体。那么对应的数学模型则可简化成，初始集合中包含一定数量的随机数，而这些随机数通过一定规则反复迭代计算，然后设计一种个体的评价体系，把不符合规则或者评分比较低的个体淘汰，得到的最终集合里绝大部分是最符合规则的数。

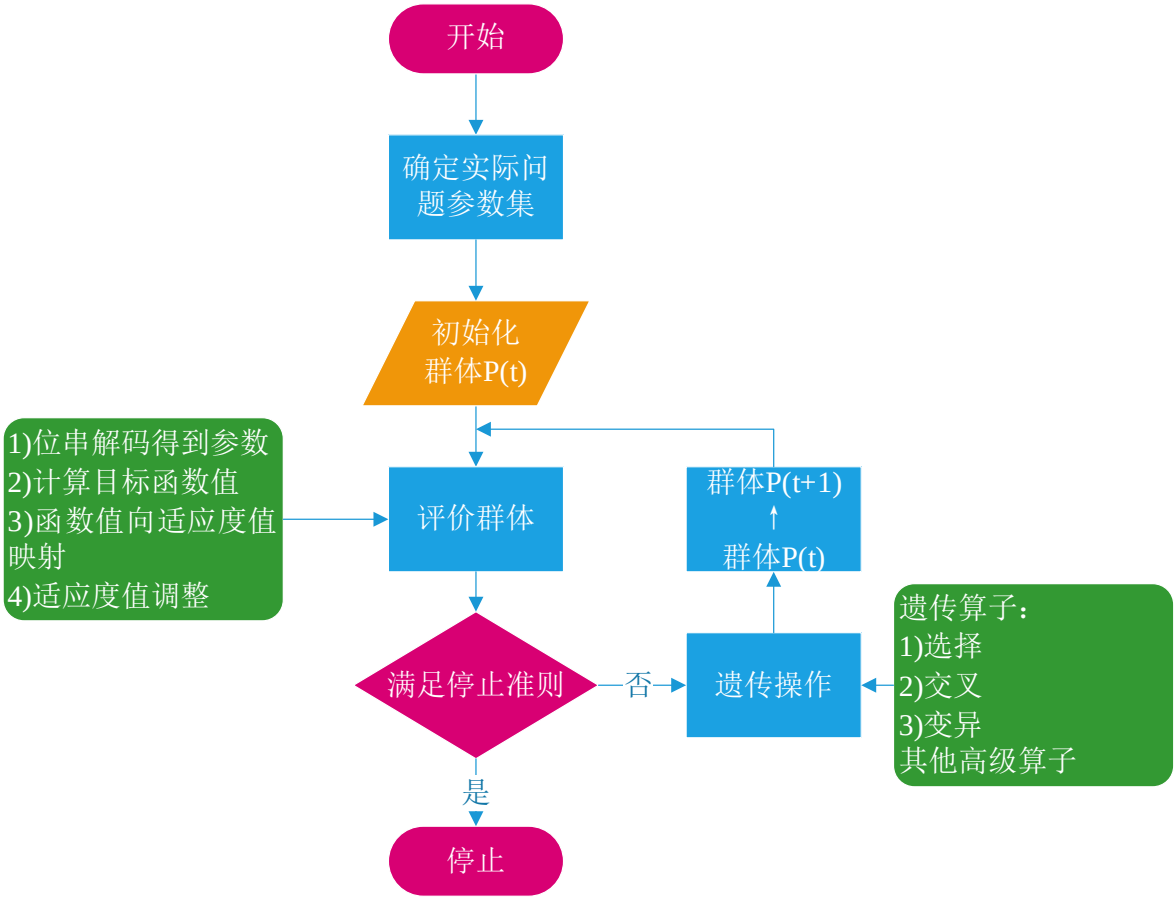


图 12 遗传算法步骤图

b) **染色体编码**

我们这里采用自然编码的方式，编码的数组需要包含检查点信息和标识符，这样就能确定一条参赛的路线方案。例如，4，15，11，22，9，M。这样的一组数字表示参赛选手从起点 4 号检查点出发，依次到达 15 号检查点，11 号检查点，22 号检查点进行打卡，最终到达终点 9 号检查点；M 值为一个负数，既是为了区别于检查点的序号，也是为了与下一条路线分隔开来，由于 M 值附带的负属性，所以很容易就能从数组中找出分割点。将上述的一串数字看成一个组，那么代表一个可行解的染色体编码就是由若干个组连接而成的，每个组表示一条比赛路线，有几个组就表示总数有几条路线。

c) 适应度函数

一个个体(解)的好坏主要由用适应度函数值来评价，在本问题中，f(x)就是适应度函数。且适应度函数值越大，解的质量越高。适应度函数是遗传算法进化的驱动力，也是进行自然选择的唯一标准，它的设计应结合求解问题本身的要求而定。在本问题情境中，考虑到问题对每条路线的约束，我们要实现路径的长度近似均匀和难度近似均匀，同时为了保证映射后的适应度值是非负的，而且目标函数的优化方向应该对应于适应度值增大的方向，所以我们建立如下适应函数与目标函数的映射关系形式

$$f\left(x\right)=c_{\max}-g(x) \tag{16}$$

其中，$c_{\max}$ 是我们引入的一个输入的理论最大值，$g\left(x\right)$ 为目标函数。

$$\begin{aligned} g=&k_1\sum_{k=1}^{K}\left(\sum_{l_{ij}\in L_k}\sqrt{\left|x_i-x_j\right|^2+\left|y_i-y_j\right|^2}-\frac{1}{K}\sum_{k=1}^{K}\sum_{l_{ij}\in L_k}\sqrt{\left|x_i-x_j\right|^2+\left|y_i-y_j\right|^2}\right)^2\\ &+k_2\sum_{k=1}^{K}\left(\sum\left\{df_{n_j}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{df_{n_j}\right\}\right)^2\\ &+k_3\sum_{k=1}^{K}\frac{T_{endk}-\dfrac{\sum\limits_{l_{ij}\in L_k}\sqrt{\left|x_i-x_j\right|^2+\left|y_i-y_j\right|^2}}{v_0}-\sum\left\{df_{n_m}\right\}}{\Delta t_k} \end{aligned} \tag{17}$$

其中，k_1,k_2,k_3 分别为与路线长度、路线难度和预计路线安排人数相关的加权系数。

d) 遗传算法初始解产生

由于起点、终点及两个必经的补给点和总路线数 K 已经确定，我们先随意生成 K 次 16 个乱序的检查点编号数组，并在每组数据的首端和末端加入起点和终点，在中间的任意 2 个位置插入补给点，从而形成 K 个 20 点的新数组，并利用标识符 M 将这 K 个数组连接起来，形成一条初始可行解染色体。

e) 更新染色体

在每次迭代更新过程中，不断更新个体极值和群体极值。

在遗传操作过程中，我们秉承着“物竞天择，适者生存”和“自然选择”的原则，进行如下操作：

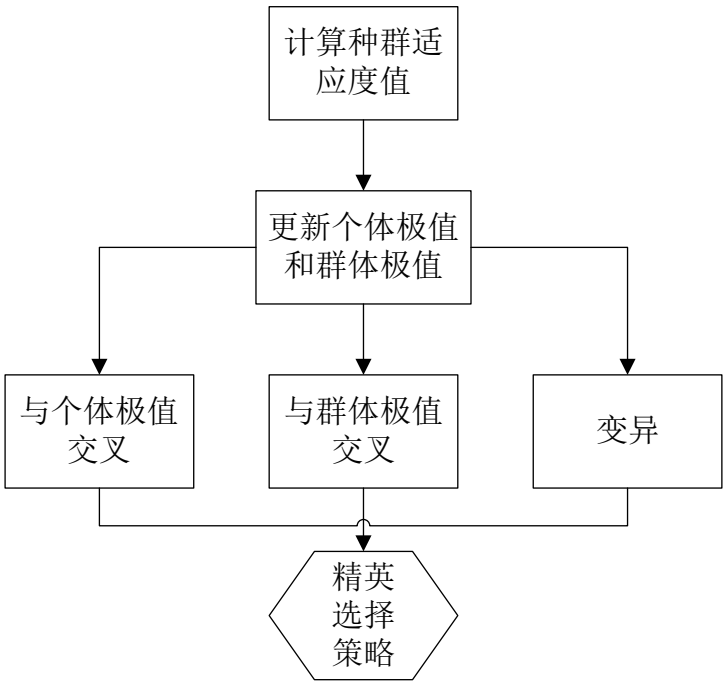


图 13 染色体处理筛选图

f) 选择运算

从群体中选择优胜个体，淘汰劣质个体的操作叫选择。选择即从当前群体中选择适应度值高的个体以生成配对池的过程。通过选择算子模拟“优胜劣汰”，适应度高的个体被遗传到下一代的概率较大，适应度低的算子被遗传到下一代的概率较小。这里我们采用轮盘赌的选择法。对于给定的规模为 N 的染色体群体 $P=\{a_1,a_2,...,a_N\}$，个体 $a_j \in P$ 的适应度值为 $f\left(a_j\right)$，其被选择的概率为

$$P\left(a_j\right)=\frac{f\left(a_j\right)}{\sum_{i=1}^{N} f\left(a_j\right)} \tag{18}$$

g) 交叉运算

交叉即对两个相互配对的染色体依据交叉概率 Pc 按某种方式相互交换其部分基因，从而形成两个新的个体。交叉运算是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起关键作用，是产生新个体的主要方法。

此问题的交叉方法我们采用整数交叉的方式。在进行交叉运算之前，首先要删除染色体中所插入的负数 M，只留下表示检查点序号的数；然后随机选择两个交叉位置和数目，把个体和个体极值或者群体极值进行交叉。以下式的 A，B 染色体进行交叉为例：

$$A=1\ 9\ 15\ 11\ [6\ 13\ 2\ 17]\ 10\ 18\ 14\ 3\ 8\ 7\ 19\ 4\ 16\ 5\ 20\ 12$$

$$B=1\ 27\ 5\ 23\ [21\ 34\ 13\ 2]\ 20\ 33\ 10\ 11\ 8\ 31\ 16\ 29\ 26\ 38\ 30\ 12$$

对 A 和 B 染色体交叉后，我们得到

$$A=1\ 9\ 15\ 11\ [21\ 34\ 13\ 2]\ 10\ 18\ 14\ 3\ 8\ 7\ 19\ 4\ 16\ 5\ 20\ 12$$

$$B=1\ 27\ 5\ 23\ [6\ 13\ 2\ 17]\ 20\ 33\ 10\ 11\ 8\ 31\ 16\ 29\ 26\ 38\ 30\ 12$$

h) 变异运算

变异即依据变异概率 Pm 将个体编码串中的某些基因值用其它基因值来替换，从而形成一个新的个体。遗传算法中的变异运算是产生新个体的辅助方法，它决定了遗传算法的局部搜索能力，同时保持了种群的多样性。具体实现步骤如下：首先在种群中随机选择 3 个基因位点 14、28 和 13，然后将 3 个基因位点之前的染色体片段位置互换，例如：

$$C = 1\ 32\ 15\ 40\ [14]\ 22\ 26\ 36\ [28]\ 30\ 18\ 34\ 8\ [13]\ 37\ 23\ 35\ 29\ 3\ 12$$

变异后得到

$$C' = 1\ 32\ 15\ 40\ [30]\ 18\ 34\ 8\ 13\ [14]\ 22\ 26\ 36\ [28]\ 37\ 23\ 35\ 29\ 3\ 12$$

i) 精英选择

从 GA 的整个选择策略来讲，精英选择时群体收敛导优化问题全局最优解的种基本保障。我们按照染色体适应度值从大至小进行排序，我们将每代中具有最大适应度值的个体定义为精英个体。在通过交叉和变异操作产生的种群中，如果下一代群体的最佳个体适应度值小于当前群体最佳个体的适应度值，则将当前群体最佳个体或者适应度值大于下一代最佳个体适应度值的多个个体直接复制到下一代，随机替代或者替代最差的下一代群体中的相应数量的个体，排在末位的个体则被舍弃。在迭代过程中。使用这种自适应方法不仅可以有效的避免早熟和近亲繁殖，还可以加快种群整体收敛速度。

其中，每一次进化都尽可能保留种群中的优秀个体，淘汰掉不理想的个体，并且在优秀个体之间进行染色体交叉，有些个体还可能出现变异。交叉运算和变异运算的相互配合，共同完成对搜索空间的全局搜索和局部搜索。

种群的每一次进化，都会产生一个最优个体。种群所有世代的最优个体，可能就是函数 $f(x)$ 最大值对应的定义域中的点。

如果种群无休止地进化，那总能找到最好的解。但实际上，我们的时间有限，通常在得到一个看上去不错的解时，便终止了进化。

4.1.5 结果与分析

在算法实现部分，利用 MATLAB 编写程序实现了模型的仿真与求解。

其中，参数设置如下：

表 2 阈值参数设置

参数	符号	参数大小
总长度均方差阈值	δ_{dth}	750
总难度均方差阈值	δ_{DFth}	6
平均重复长度占比阈值	η_{th}	10%
最小出发时间间隔(min)	t_0	3

设定不同的 K 可以优化得到不同最佳路径 $L_1 \sim L_k$ 及不同情况下每条路线出发间隔、最大参赛人数，画图如下：

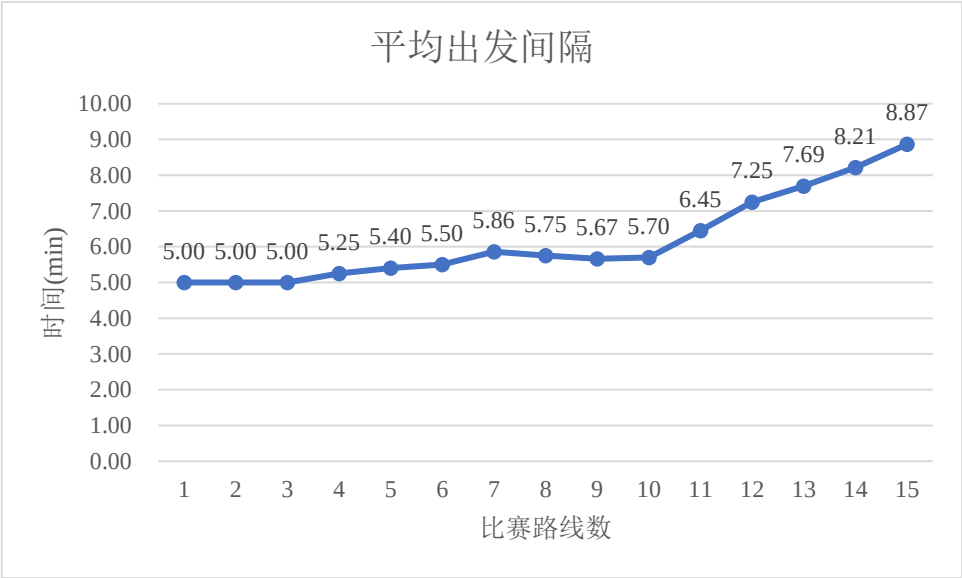


图 14 平均出发间隔与路线数关系图

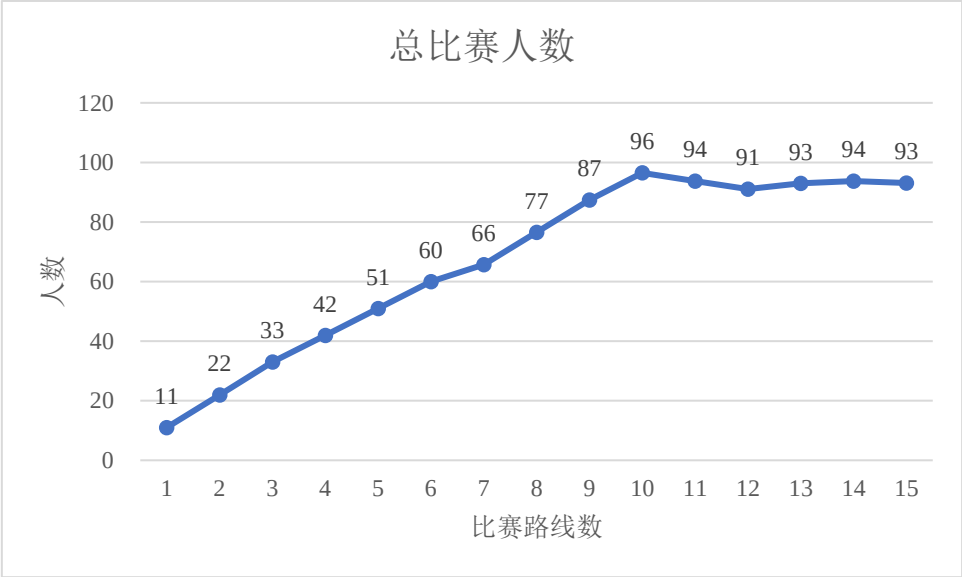


图 15 总比赛人数与路线关系图

可以看出，随着比赛路线数目增大，平均出发间隔逐渐增大，最大参赛人数先增大然后减小，与实际相符。当只有一条路径时，不存在多个人同时到达同一检查点限制，出发间隔可以设置的很小，但因为比赛路线少最终参赛人数还是很少；当路径数目很大时，在多个节点处都存在多人同时到达同一节点的现象，从而约束了出发时间间隔，使得最后出发波次较少，总参赛人数不多。最终在 $K=10$ 处取到全局最优解，此时最大参赛人数为 96，各比赛路线出发时间间隔如下表所示：

表 3 各条路线出发时间间隔表

第 i 条比赛路线	1	2	3	4	5	6	7	8	9	10	平均

出发时间间隔 Δt_i (min)	4	4	4	6	4	4	5	5	6	5	5.7

这 10 条路线各自参赛人数为：

表 4 各条路线参赛人数表

路线	1	2	3	4	5	6	7	8	9	10	总人数
参赛人数（人）	12	12	10	8	11	11	10	8	7	7	96

该方案下各路线检查点顺序见“附件 4”。

这 10 条路线各自长度为：

表 5 各条路线总长度表

路线	1	2	3	4	5	6	7	8	9	10	平均
长度 (m)	6792	7722	8148	6832	7004	6736	6687	8183	6545	7291	7194

这 10 条线路各自难度为：

表 6 各条路线总难度表

路线	1	2	3	4	5	6	7	8	9	10	平均
难度(m)	47	44	51	53	54	50	54	60	59	58	53

此时 10 条比赛路线设计如下：

迭代次数为 10：

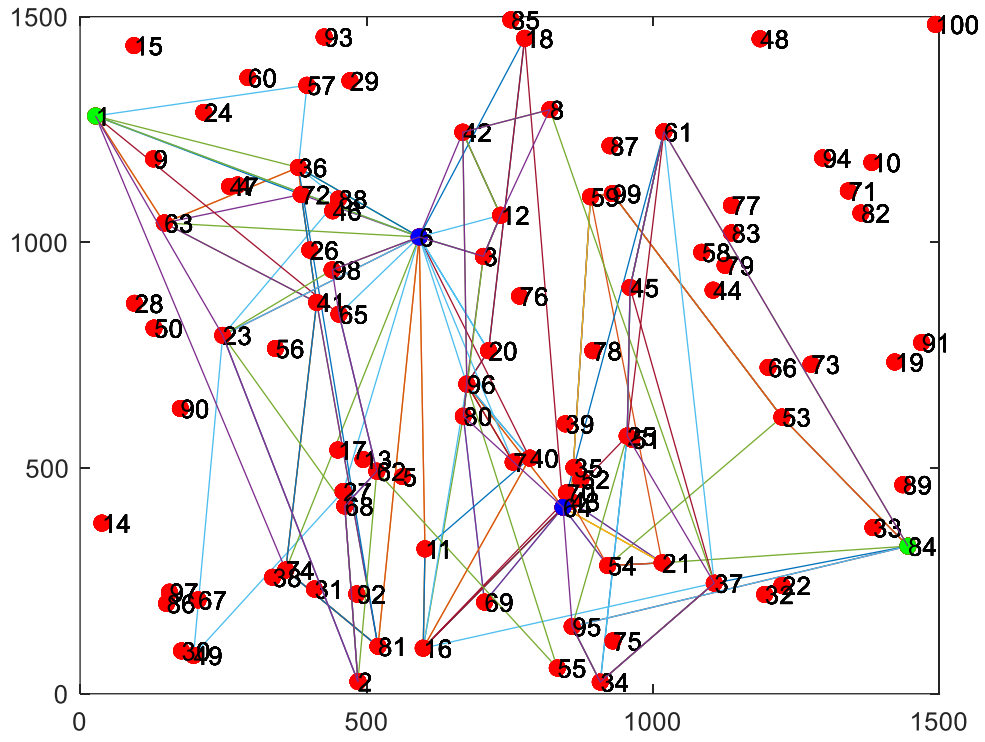


图 16 10 次迭代结果图

迭代次数为 100：

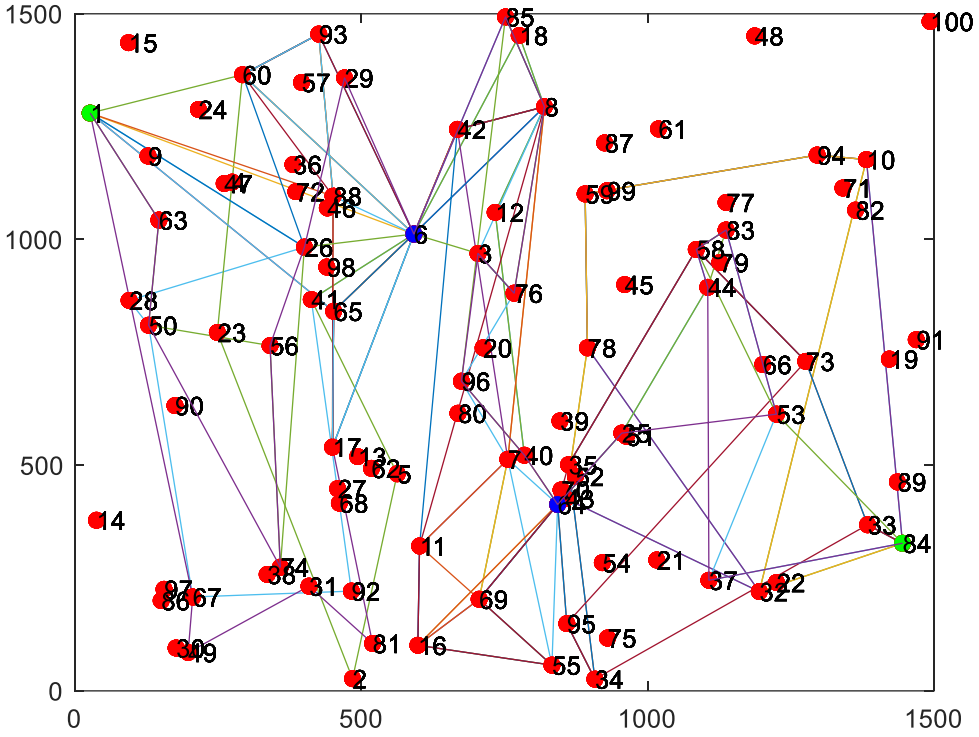


图 17 100 次迭代结果图

迭代次数为 500：

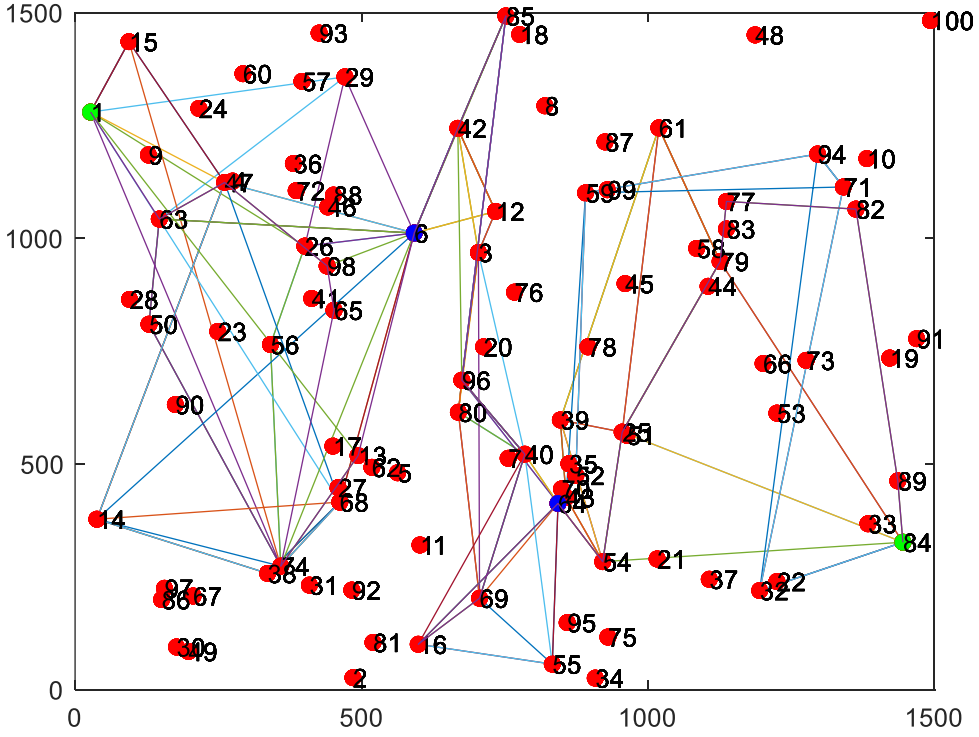


图 18 500 次迭代结果图

迭代次数为 1000：

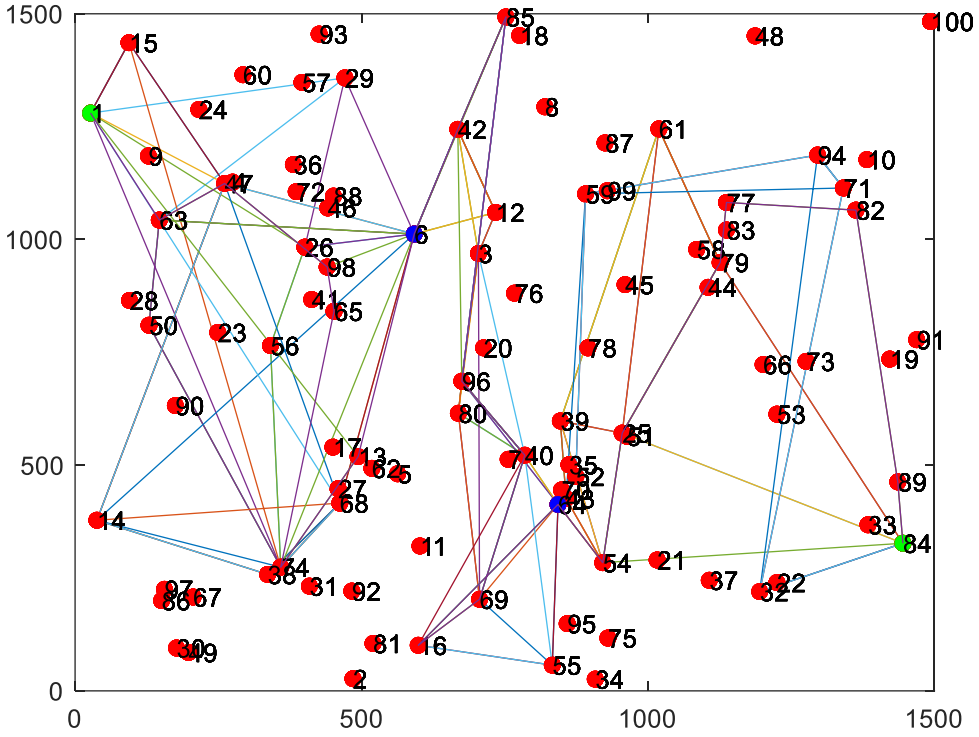


图 19 1000 次迭代结果图

迭代次数为 1500：

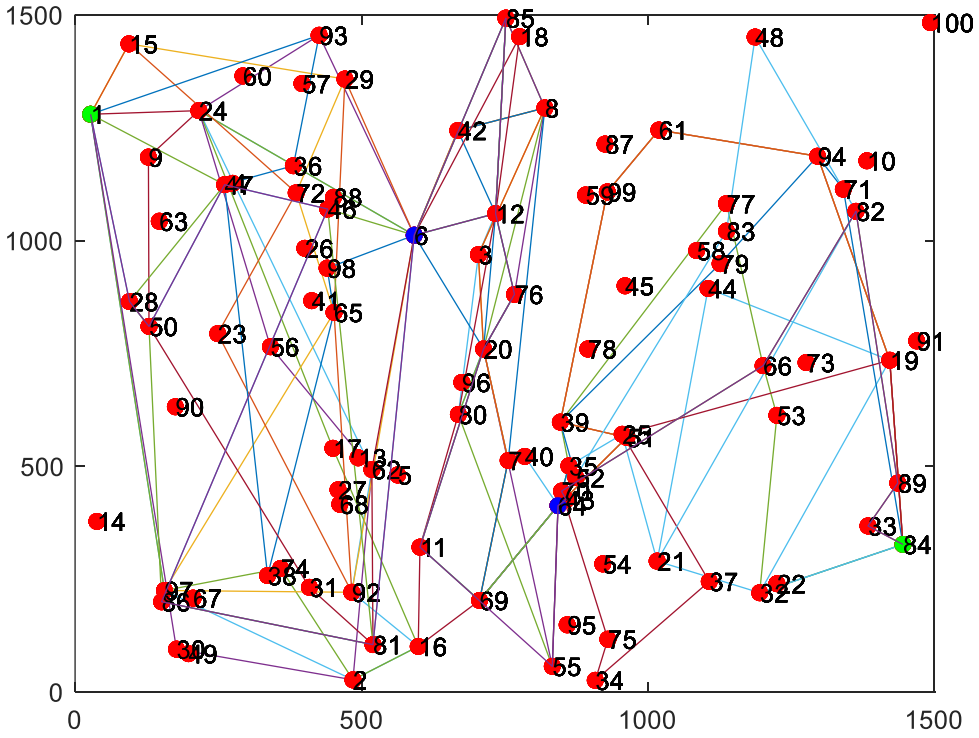


图 20 1500 次迭代结果图

上图表示在算法求解过程中设定 $K=10$，使用遗传算法不同迭代次数下得到的路

线规划结果。可以看出，当迭代次数较小时，没有找到最优解，甚至有些路线设计不符合题目约束，当迭代次数不断增加，结果趋于稳定不再变化，最终得到$K=10$时符合条件的最佳路线设计。

迭代次数为 10 时每条路线如下所示：

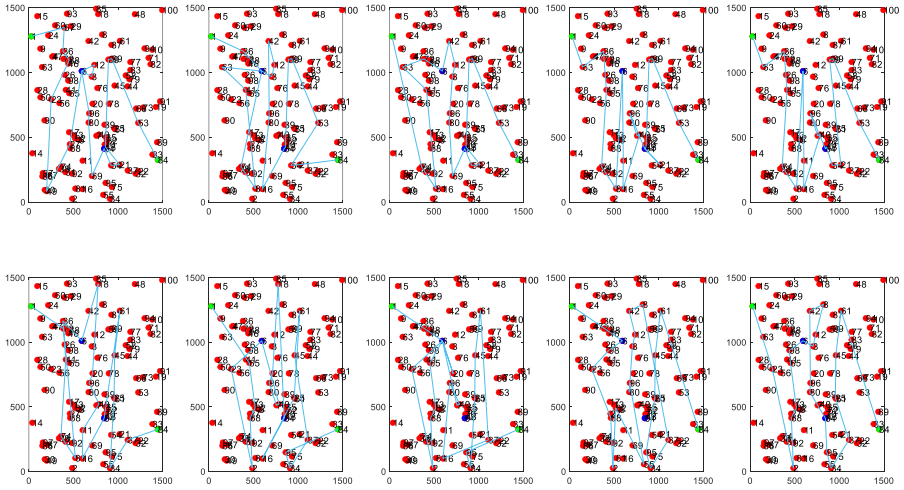


图 21 10 次迭代结果示意图

迭代次数为 1000 时每条路线如下所示：

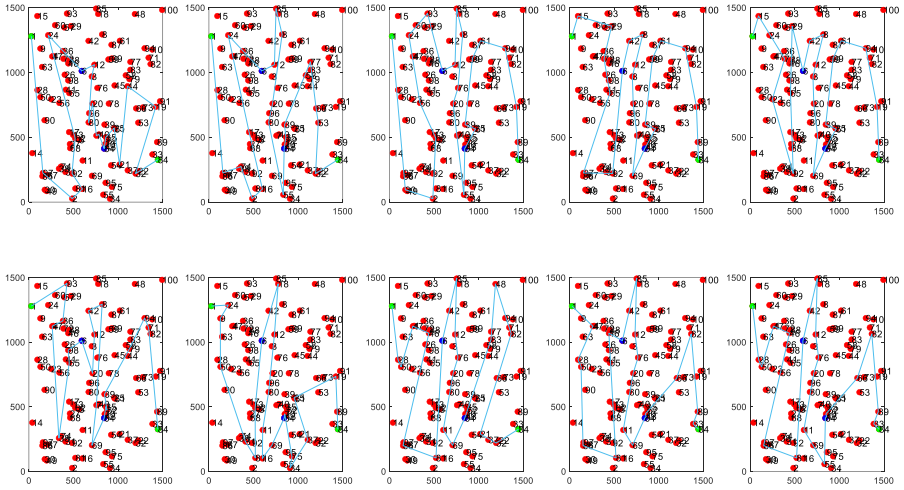


图 22 1000 次迭代结果示意图

可以看出，当迭代次数为 10 时甚至有些路线不符合题目约束，出现“绕圈跑”现象，当迭代次数到 1000 之后，则没有类似问题。说明了本题使用遗传算法实现多路径优化的可行性及良好性能。具体“细节图”如下所示：

迭代次数为 10 时“绕圈跑”现象：

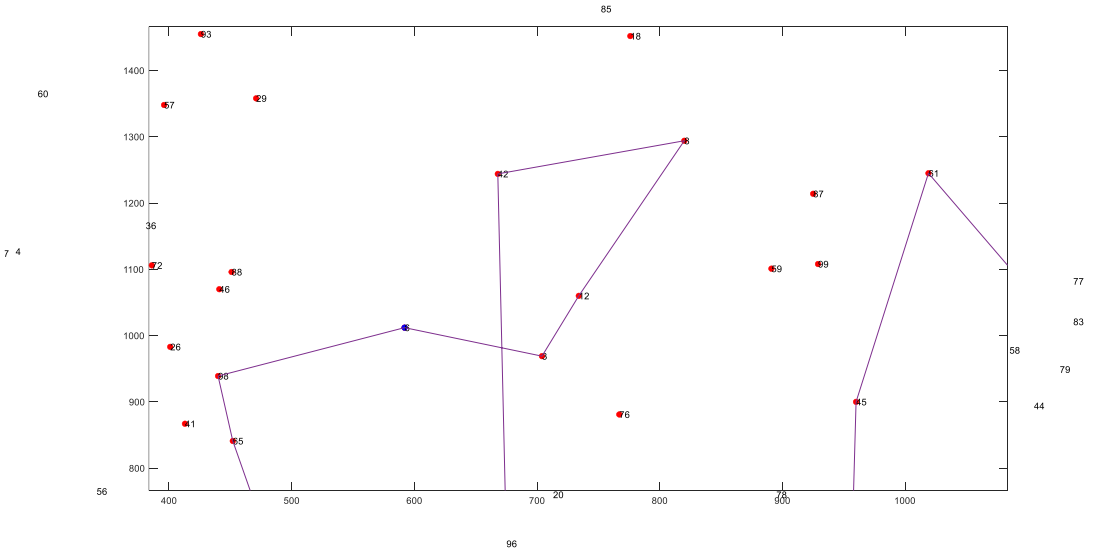


图 23 迭代 10 次结果细节展示图(a)

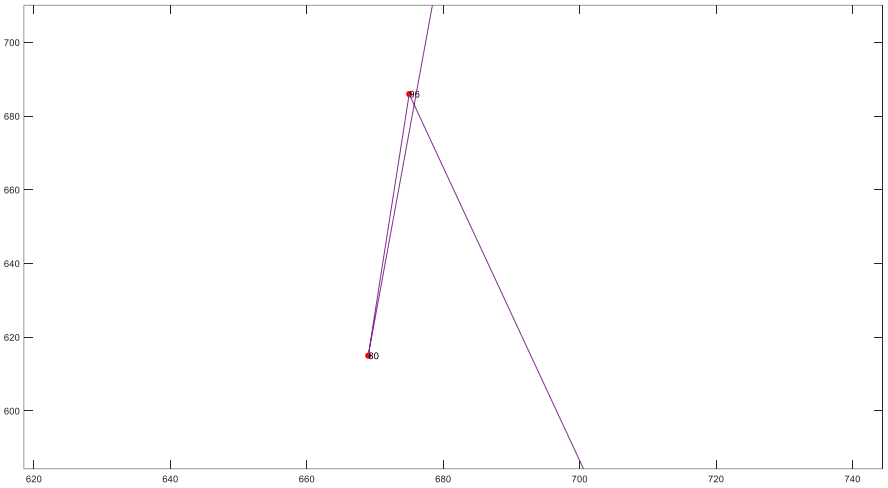


图 24 迭代 10 次结果细节展示图(b)

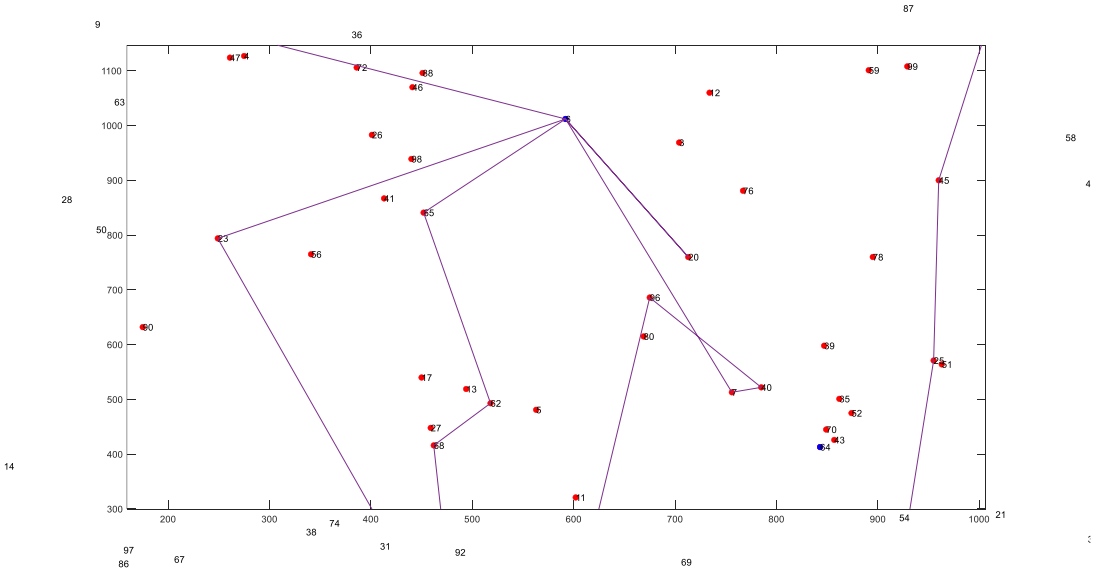


图 25 迭代 10 次结果细节展示图(c)

迭代次数为 1000 时路线细节：

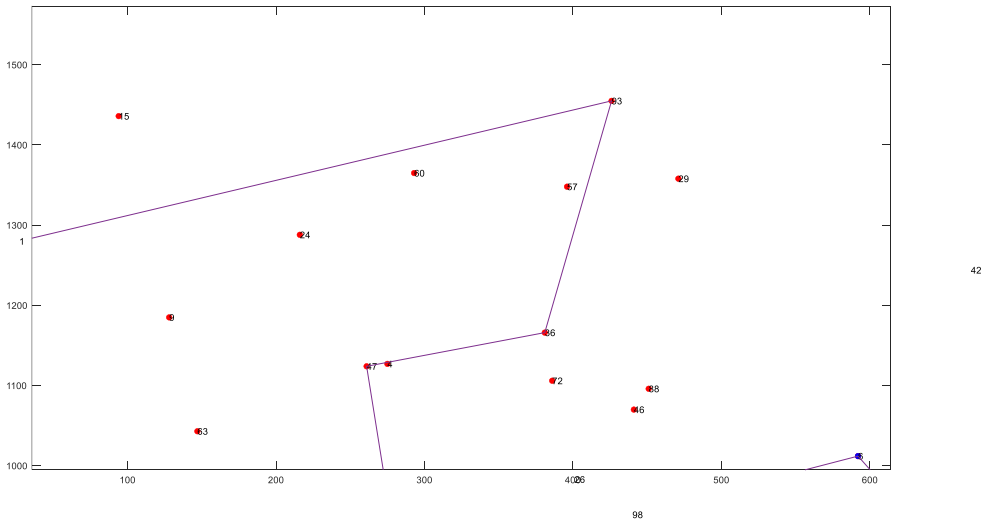


图 26 迭代 1000 次结果细节展示图

通过进一步观察细节图发现，迭代次数很小，仅仅 10 次时，很多路线设计都出现了不符合设计规则的部分，但是随着迭代次数增加，最终设计方案没有类似问题，同样验证了算法的性能。

计算最终优化结果路线的总长度方差、总难度方差、平均重复长度占比方面性能与阈值对比如下所示：

表 7 结果性能-阈值比较表

	总长度方差	总难度方差	平均重复长度(m)	平均重复长度占比
性能	613.94	5.19	482.71	6.71%
阈值	750	6	无	10%

通过设定阈值实现题目中要求的路线长度尽可能相等、难度尽可能相等、重复路段尽可能少等设计规则，最终得到的最佳方案均满足设定的性能需求，但因为目标函数为参赛人数最大与这些指标无关，因此“最佳路线规划”虽然没有体现出“路线总方差最小等”特点，但满足了题目中“长度尽可能相等”的要求，还可以在阈值选取时，结合实际考虑选取合适的阈值，提升最终优化结果在长度尽可能相等方面的性能。

4.2 问题二分析与求解

问题二进一步给出各检查点高程数据坐标 z_i，进一步要求每条比赛路线的爬高量尽可能均匀。前后两个检查点的爬高量定义为路线中后一个检查点的 z_i 坐标高于前一个检查点的坐标 z_j 时，它们之间的高程差；路线上所有前后检查点的爬高量之和称为该条路线的爬高量。问题二的其他方面与问题一相似，因此可以把爬高量尽可能相等表示成约束条件，改进问题一模型算法完成问题二的求解。

4.2.1 模型改进

(一) 路线优化模型

比赛路线设计规则(3)要求不同路线总爬高应尽可能平均，优化目标为“尽可能相等”难以实现，可以转化为不同路线总爬高均方差尽量小来简化问题。设定各路线的总距离均方差阈值δ_{hth}，要求不同路径均方差小于阈值。数学模型如下所示：

$$\frac{1}{K}\sum_{i=1}^{K}\left(H_i-\overline{H}\right)^2<\delta_{hth}$$

$$s.t.\begin{cases} h_{l_{m,n}}=\begin{cases} z_n-z_m, z_n>z_m\text{时} \\ 0, \quad z_n\le z_m\text{时} \end{cases} \\ H_i=\sum\left\{h_{l_{m,n}}\right\}, \forall l_{m,n}\in L_i; \\ \overline{H}=\frac{1}{K}\sum_{i=1}^{K}H_i; \end{cases} \tag{19}$$

其中，第一个式子表示两个检查点之间的爬高量计算，后一个节点高程坐标z_n高于前一个节点高程坐标z_m时爬高量$h_{l_{m,n}}=z_n-z_m$，否则爬高量$h_{l_{m,n}}=0$。第二个式子用于计算路线i的整条路线爬高量H_i。第三四个式子分别用来求路线爬高量的均值及均方差。

加入尽可能保证爬高量均匀的约束后，总优化目标函数及约束条件改进为：

$$\max P=\sum_{k=1}^{K} g_k;$$

$$s.t.\begin{cases}
\sum_{k=1}^{K}\left(\sum\left\{h_{l_{m,n}}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{h_{l_{m,n}}\right\}\right)^2<\delta_{hth},\forall l_{m,n}\in L_k;\\
\sum_{k=1}^{K}\left(\sum\left\{d_{l_{m,n}}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}\right)^2<\delta_{dth},\forall l_{m,n}\in L_k;\\
\sum_{k=1}^{K}\left(\sum\left\{df_{n_j}\right\}-\frac{1}{K}\sum_{k=1}^{K}\sum\left\{df_{n_j}\right\}\right)^2<\delta_{DFth},\forall n_j\in C_k;\\
\eta=\dfrac{\sum\left\{d_{l'_{m,n}}\right\}}{\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}}<\eta_{th},\quad \forall 1\le k\le k, l_{m,n}\in L_i, l'_{m,n}\in L_{repeat};\\
\Delta t_k\ge\max\left(df_{n_j}\right),\forall 1\le k\le K, n_j\in C_k;\\
\begin{cases}a_{C_k}\ge 20;\\ C_k\bigcap C_0=C_0;\end{cases}\forall 1\le k\le K;\\
\sum_{k=1}^{K}u_{i,k}^{t}\le 5,\forall 1\le i\le N, t\in\left[0,T_{endk}\right];\\
\begin{cases}\frac{1}{K}\sum_{k=1}^{K}T_{endk}=\frac{1}{K}\sum_{k=1}^{K}\left[\frac{D_k}{v_0}+\sum\left\{df_{n_j}\right\}+\left(g_k-1\right)\Delta t_k\right]\ge 2;\\ \max T_{endk}=\max\left[\frac{D_k}{v_0}+\sum\left\{df_{n_j}\right\}+\left(g_k-1\right)\Delta t_k\right]\le 3,\quad \forall 1\le k\le K;\end{cases}\\
v_0=6\ \text{km/h}
\end{cases}\tag{20}$$

(二) 算法改进

在适应函数与目标函数的映射关系形式进行一定的修正，使得目标函数$g(x)$适应第二问的情景。

$$
\begin{aligned}
g = k_1 \sum_{k=1}^{K} & \left(\sum_{l_{ij} \in L_k} \sqrt{\left|x_i - x_j\right|^2 + \left|y_i - y_j\right|^2 + \left|z_i - z_j\right|^2} \right. \\
& \left. - \frac{1}{K} \sum_{k=1}^{K} \sum_{l_{ij} \in L_k} \sqrt{\left|x_i - x_j\right|^2 + \left|y_i - y_j\right|^2 + \left|z_i - z_j\right|^2} \right)^2 \\
& + k_2 \sum_{k=1}^{K} \left(\sum \left\{ df_{n_j} \right\} - \frac{1}{K} \sum_{k=1}^{K} \sum \left\{ df_{n_j} \right\} \right)^2 \\
& + k_3 \sum_{k=1}^{K} \frac{T_{endk} - \dfrac{\sum\limits_{l_{ij} \in L_k} \sqrt{\left|x_i - x_j\right|^2 + \left|y_i - y_j\right|^2 + \left|z_i - z_j\right|^2}}{v_0} - \sum \left\{ df_{n_m} \right\}}{\Delta t_k} \\
& + k_4 \sum_{k=1}^{K} \left(\sum \left\{ h_{l_{m,n}} \right\} - \frac{1}{K} \sum_{k=1}^{K} \sum \left\{ h_{l_{m,n}} \right\} \right)^2
\end{aligned}
\tag{21}
$$

4.2.2 结果与分析

a) **必经点选取结果**

运用第一问部分必经点选取策略，带入新的检查点数据，可以得到，当α分别取 10、150、500 时必经点选取结果如下：

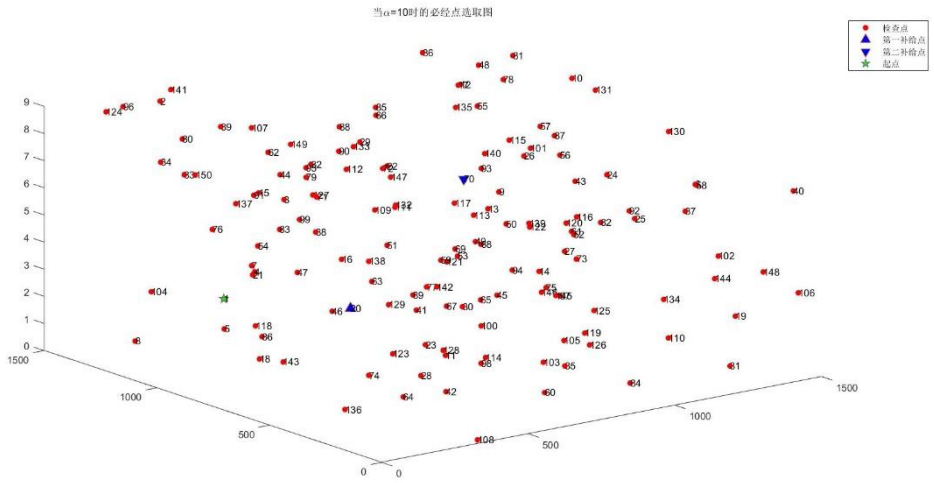


图 27 α 取 10 必经点选取结果图

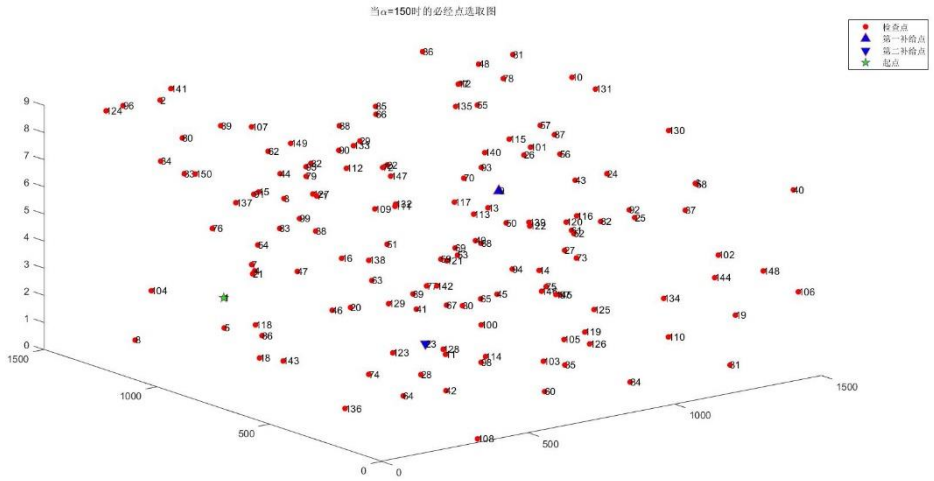


图 28 α 取 150 必经点选取结果图

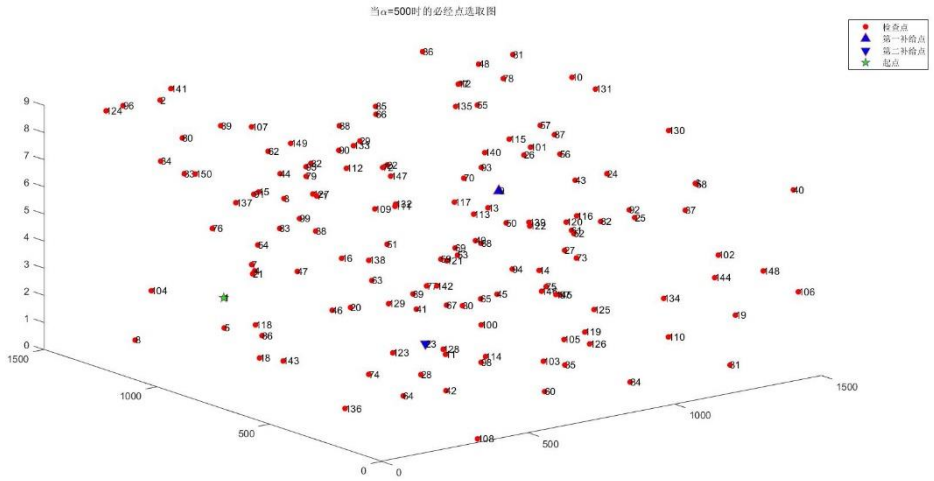


图 29 α 取 500 必经点选取结果图

事实上，这些结果并不符合实际，采用归一化策略后可以减少人为主观因素的影响，即减小了“必经点”到其他点的“代价”的差异，提高了必经点选取的公平性，最终确定问题二中了必经点选取为 N_{20} 和 N_{26}。

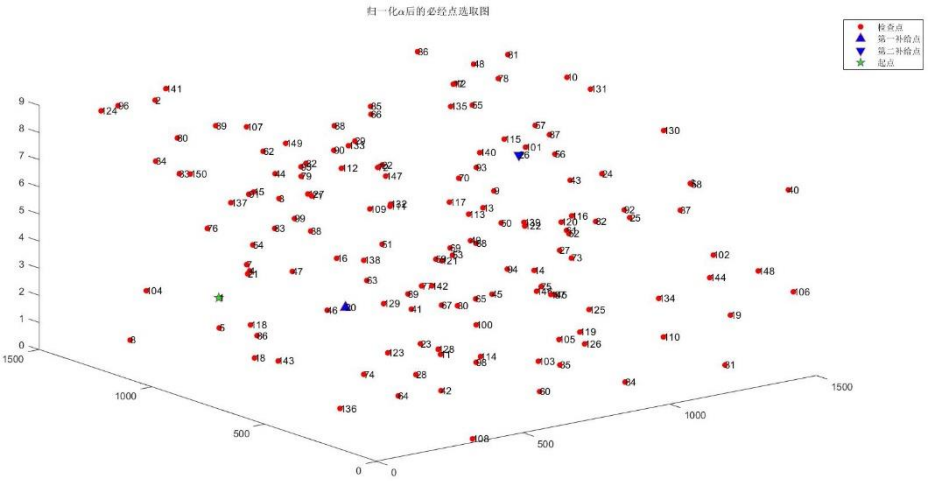


图 30 最终必经点选取结果图

b) **终点选取结果**

跟第一部分必经点选取类似，得到使选择的节点到达起点的距离较远且难度较小的终点为 N_{131}，如下图所示

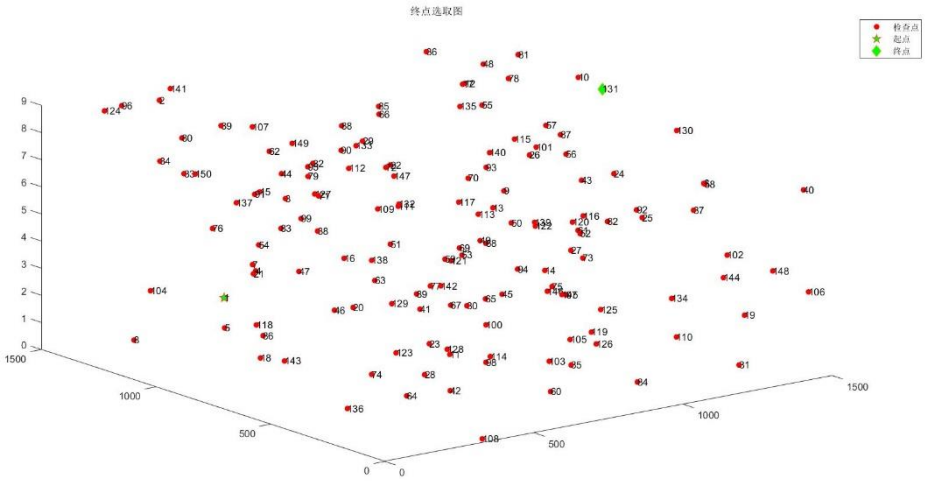


图 31 终点选取结果图

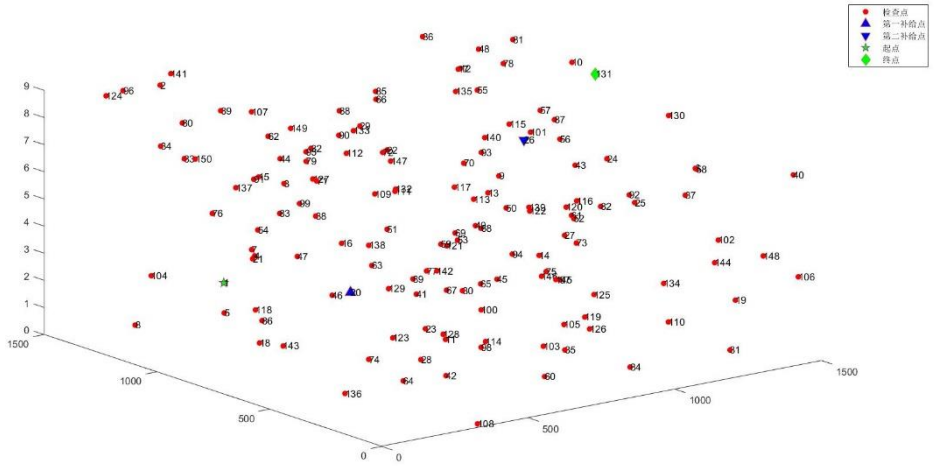


图 32 必经点、终点选取结果图

c) **路线优化算法求解**

在算法实现部分，利用 MATLAB 编写程序实现了模型的仿真与求解。其中，参数设置如下：

表 8 参数取值表

参数	符号	参数大小
总长度均方差阈值	δ_{dth}	200
总难度均方差阈值	δ_{DFth}	5
总爬高均方差阈值	δ_{hth}	3
平均重复长度占比阈值	η_{th}	10%
最小出发时间间隔(min)	t_0	3

与问题一类似，设定不同的 K 可以优化得到不同最佳路径 $L_1 \sim L_k$ 及不同情况下每条路线出发间隔、最大参赛人数如下所示：

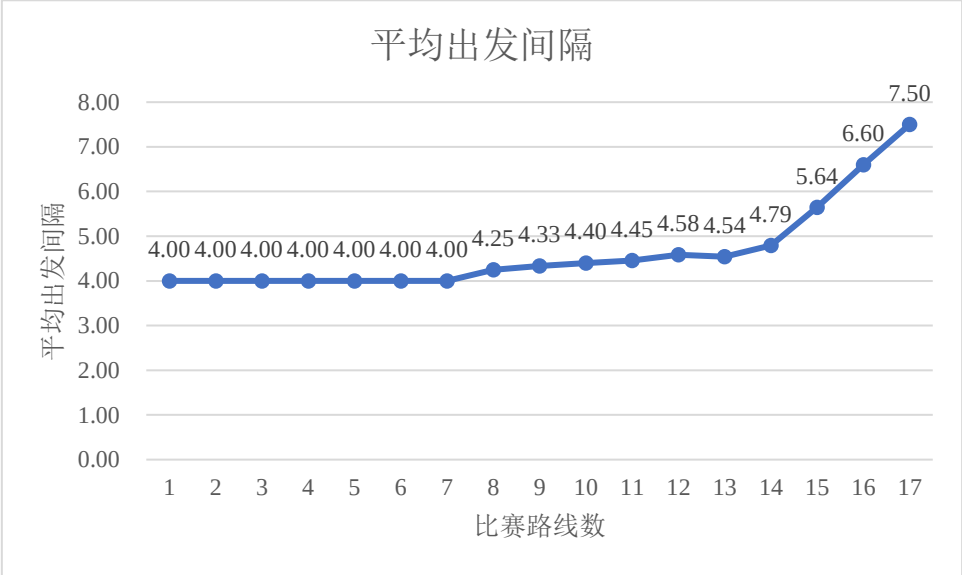


图 33 不同比赛路线数平均出发间隔图

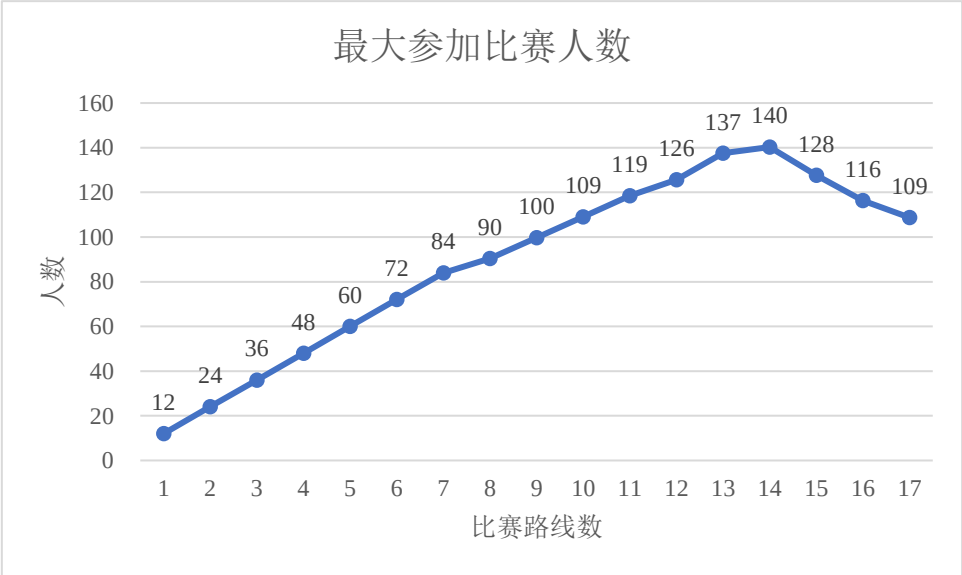


图 34 不同比赛路线最大参赛人数图

可以看出，随着比赛路线数目增大，平均出发间隔逐渐增大，最大参赛人数先增大然后减小，与实际相符。当只有一条路径时，不存在多个人同时到达同一检查点限制，出发间隔可以设置的很小，但因为比赛路线少最终参赛人数还是很少；当路径数目很大时，在多个节点处都存在多人同时到达同一节点的现象，从而约束了出发时间间隔，使得最后出发波次较少，总参赛人数不多。最终在 $K=14$ 处取到全局最优解，此时最大参赛人数为 140，各比赛路线出发时间间隔如下表所示：

表 9 各条路线出发时间间隔表

第 i 条比赛路线	1	2	3	4	5	6	7	8	9	10	11	12	13	14	平均
出发时间间隔 Δt_i (min)	4	5	4	6	4	5	5	4	5	4	6	5	5	5	4.79

这 14 条路线各自参赛人数为：

表 10 各条路线参赛人数表

路线	1	2	3	4	5	6	7	8	9	10	11	12	13	14	总人数
参赛人数（人）	12	8	11	7	12	9	10	11	10	13	8	9	10	10	140

该方案下各路线检查点顺序见“附加 4”。

这 14 条路线各自长度为：

表 11 各条路线总长度表

路线	1	2	3	4	5	6	7	8	9	10	11	12	13	14	平均
长度(m)	7825	8234	7778	7826	7661	7830	7933	8029	7642	7678	7727	7862	7956	7772	7839

这 14 条线路各自爬高为：

表 12 各条路线总爬高量表

路线	1	2	3	4	5	6	7	8	9	10	11	12	13	14	平均
爬高(m)	41	33	43	38	40	33	31	33	39	41	40	42	36	40	38

这 14 条线路各自难度为：

表 13 各条路线总难度表

路线	1	2	3	4	5	6	7	8	9	10	11	12	13	14	平均
难度(min)	52	58	58	58	56	56	52	53	54	53	53	56	49	53	54

此时 14 条比赛路线迭代过程如下：

迭代次数为 100：

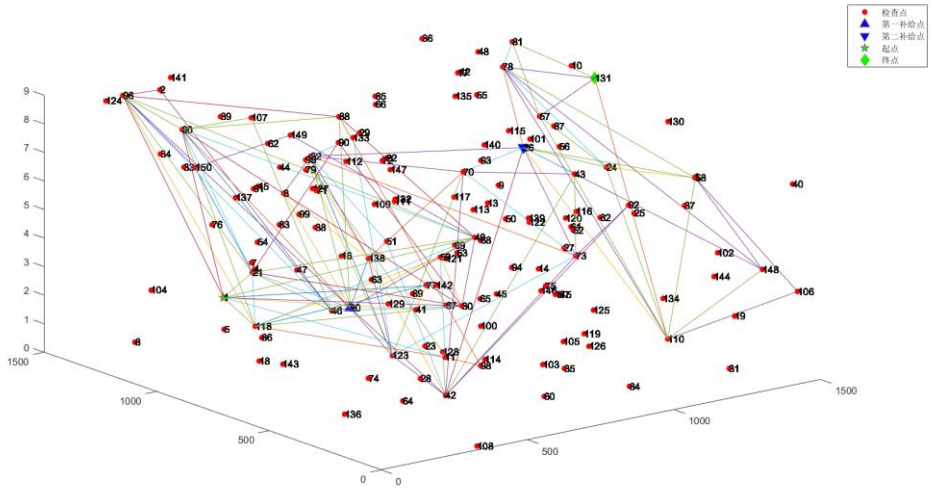


图 35 100 次迭代结果图

迭代次数为 500：

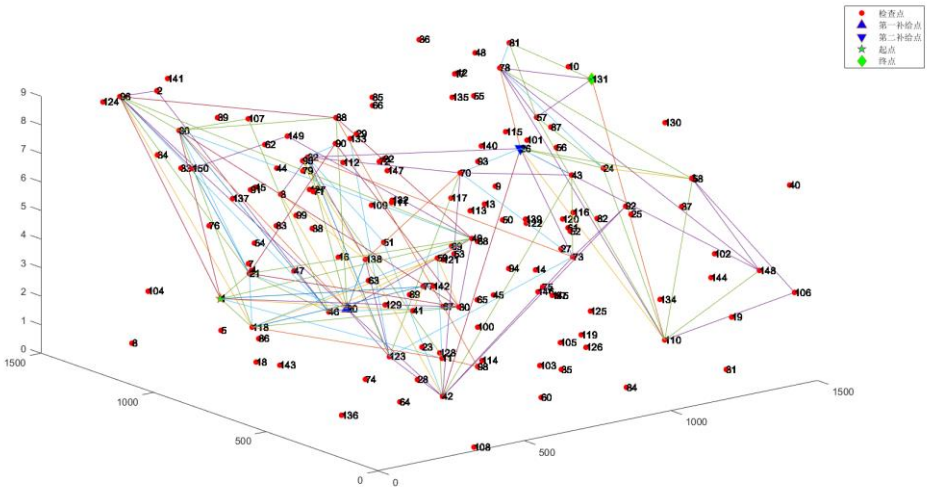


图 36 500 次迭代结果图

迭代次数为 1000：

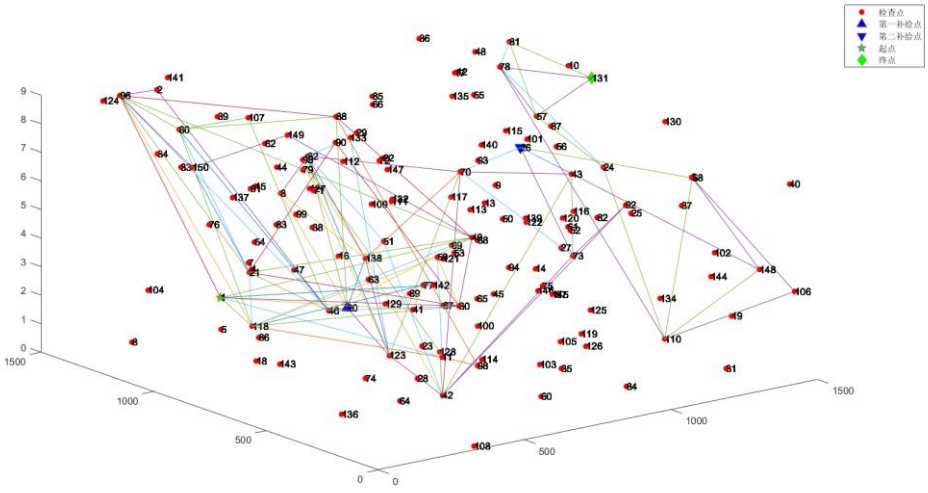


图 37 1000 次迭代结果图

迭代次数为 1500：

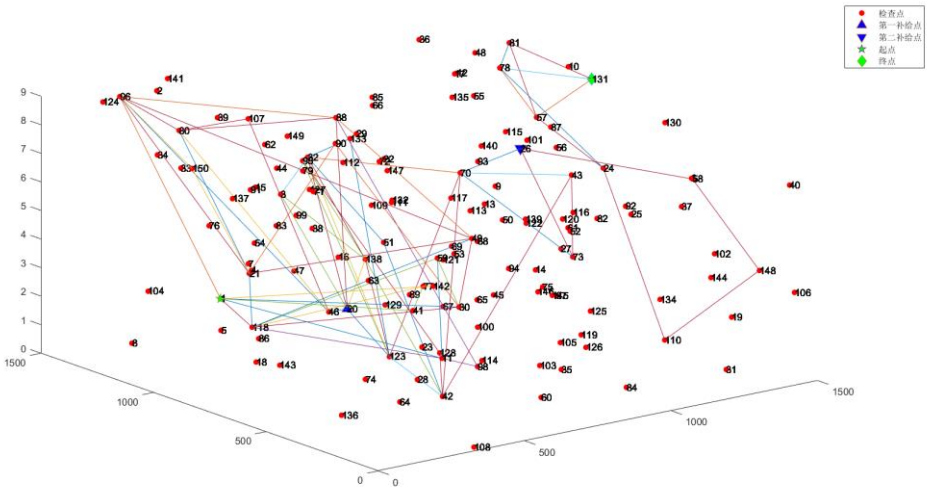


图 38 1500 次迭代结果图

迭代次数为 2000：

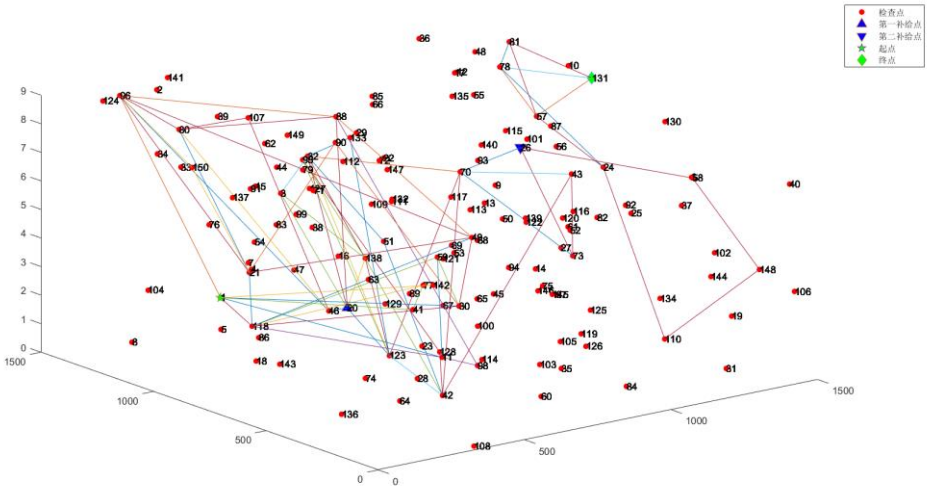
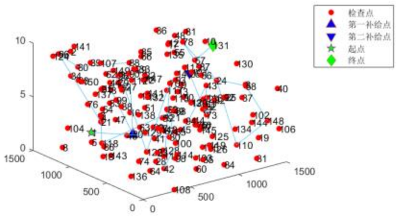
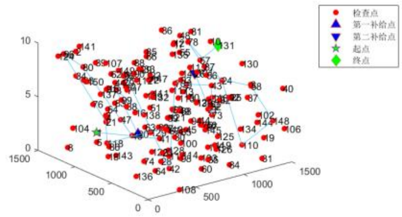


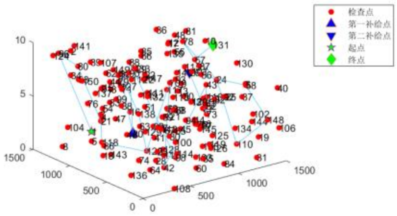
图 39 2000 次迭代结果图

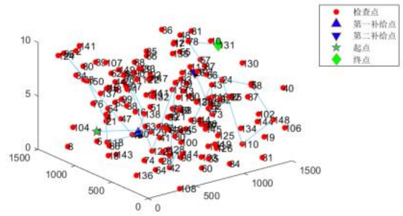
上图表示比赛路线为 10，随着迭代次数增加，优化路线规划方案变化。同样的，当迭代次数较小时，没有找到最优解，甚至有些路线设计不符合题目约束，当迭代次数不断增加，结果趋于稳定，最终得到 k=14 时符合条件的最佳路线设计。

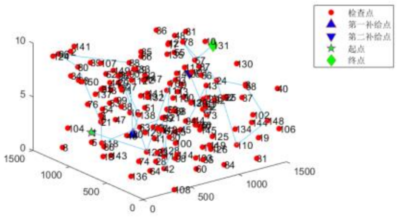
最终优化结果 14 条路线分别如下图所示：

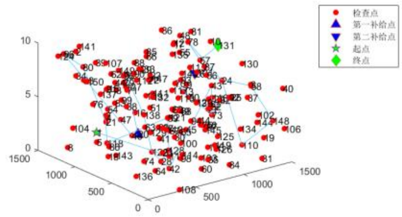

检查点
第一补给点
第二补给点
起点
终点

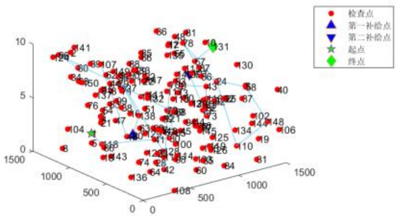

检查点
第一补给点
第二补给点
起点
终点

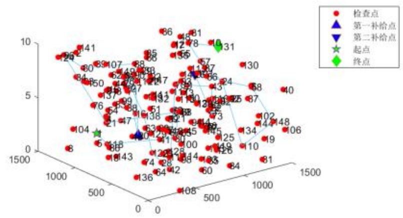

检查点
第一补给点
第二补给点
起点
终点

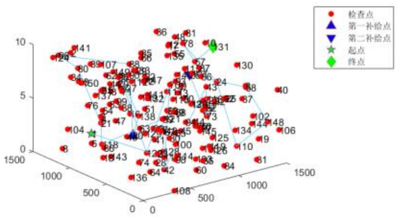

检查点
第一补给点
第二补给点
起点
终点

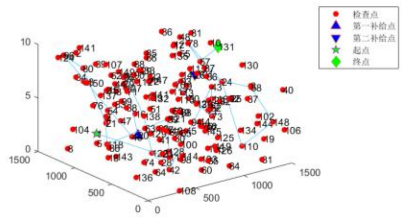

检查点
第一补给点
第二补给点
起点
终点

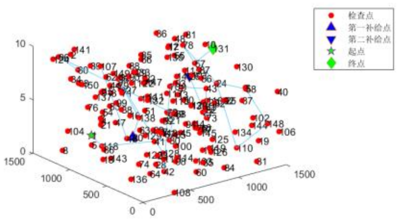

检查点
第一补给点
第二补给点
起点
终点

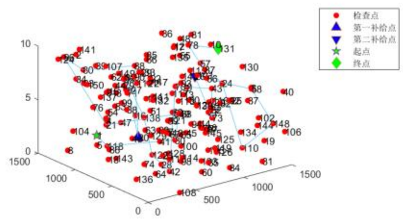

检查点
第一补给点
第二补给点
起点
终点


检查点
第一补给点
第二补给点
起点
终点


检查点
第一补给点
第二补给点
起点
终点


检查点
第一补给点
第二补给点
起点
终点


检查点
第一补给点
第二补给点
起点
终点


检查点
第一补给点
第二补给点
起点
终点

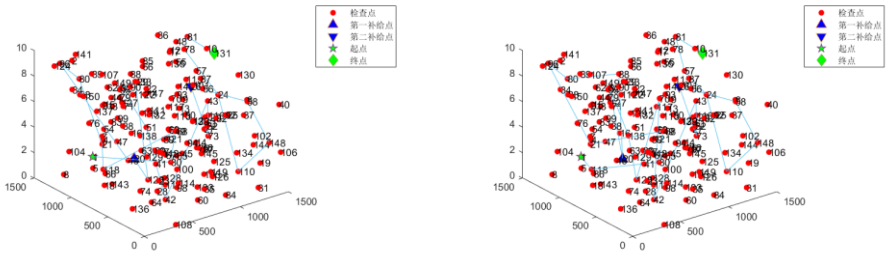


图 40 最佳 14 条路线分别展示图

可以看出，引入爬高量限制之后，算法依旧具有良好的性能，在迭代次数 1000 之后，得到满足各项约束限制的全局最优路线规划方案。

计算最终优化结果路线各方面性能与阈值对比如下所示：

表 14 结果性能-阈值比较表

	性能	阈值
总长度均方差	160.02	200
总难度均方差	3.92	5
总爬高均方差	2.71	3
平均重复长度(m)	741.60	无
平均重复长度占比	9.46%	10%

通过设定阈值实现题目中要求的路线长度尽可能相等、难度尽可能相等、重复路段尽可能少等设计规则，最终得到的最佳方案均满足设定的性能需求，但因为目标函数为参赛人数最大与这些指标无关，因此“最佳路线规划”虽然没有体现出“路线总方差最小等”特点，但满足了题目中“长度尽可能相等”的要求，还可以在阈值选取时，结合实际考虑选取合适的阈值，提升最终优化结果在长度尽可能相等方面的性能。

4.3 问题三分析与求解

与问题一二有所不同，问题三给出某次定向越野比赛的起点和检查点三维坐标及难度。确定比赛参加人数为 120 人且行进速度不同，要求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。最后希望设计此次活动的最佳比赛路线，使得使用的路线尽可能少，总完成时间尽可能短。

与问题一二相同，题目没给出必经检查点和终点，且问题三对必经点与终点约束与前面相同，因此继续使用前面问题的策略选取最佳必经点及终点。

在最佳路线设计方面，问题三为双目标优化问题，既要比赛路线少，又要完成时间短。多目标优化问题往往可以简化为单优化问题来求解，但是本题中“比赛路线数”和“总完成时间”难以结合到一起，设定一个合适的“综合优化目标”来代替。而且路线优化算法很难优化出路线数目，只能根据要求优化出满足目标的具体“路线规划方案”。

因此问题三首先还是设定路线数目 K 值，更改目标函数通过遗传算法优化得到使总完成时间最小的路线设计方案。最终结合实际情况考虑，得到最优的比赛路线数、具体比赛路线、该情况下人员分配及最小总完成时间。

此外，问题三中参赛人员速度不同，根据题目要求，我们设计的路线总长度相似、总难度相似，根据问题一二结果也可以看出每条路线出发时间间隔也相似。因此，为了避免某条路线上比赛人员速度过慢导致总完成时间过长，可以将比赛人员根据路线数目 K 平均分为人员速度均值、均方差尽可能相等的 K 组参赛人

员，进而简化问题求解难度。

4.3.1 模型改进

a) 必经点与终点选取

问题三关于必经点及终点选取策略没有较大变化，可以得到选取结果如下图：

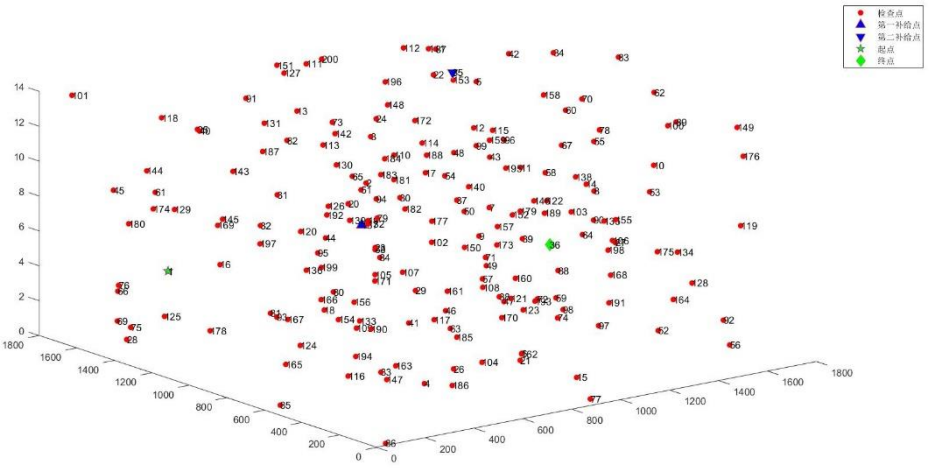


图 41 必经点选取结果图

b) 人员分配策略

为了避免某条路线上比赛人员速度过慢导致总完成时间过长，可以将比赛人员根据路线数目 K 平均分为人员速度均值尽可能相等的 K 组参赛人员。同样的，以“相等”为优化目标往往难以实现，这里转化为各组参赛人员速度均值的方差和最小来实现，数学模型如下所示：

$$\frac{1}{K}\sum_{i=1}^{K}\left(\overline{v_{L_i}}-\bar{v}\right)^2<\delta_{vth}$$

$$s.t.\begin{cases}\overline{v_{L_i}}=\dfrac{1}{p_{L_i}}\sum_{j=1}^{p_{L_i}}v_{i,j};\\ \bar{v}=\dfrac{1}{K}\sum_{i=1}^{K}\overline{v_{L_i}};\end{cases}\tag{22}$$

从而获得较均匀的人数分配方案。

c) 路线优化模型

相对于问题二，最大完成时间限制变为 3.5 小时，同时由于速度的不均匀性，我们为了保证比赛的公平性，使得同一路线上不可能发生跟跑交流的现象，就要安排出发次序为每条路线上最慢的人先出发，越快的人越后出发，保证后一个人在到达终点之前不会超过前一个人，所以可以得到增加的新的约束如下：

$$\begin{cases} T_{endi}=\dfrac{D_i}{v_{L_i,p_{L_i}}}+\sum\left\{df_{n_j}\right\}+\left(p_{L_i}-1\right)\Delta t_i; \\ \dfrac{1}{K}\sum_{i=1}^{K}T_{endi}=\dfrac{1}{K}\sum_{i=1}^{K}\left[\dfrac{D_i}{v_{L_i,p_{L_i}}}+\sum\left\{df_{n_j}\right\}+\left(p_{L_i}-1\right)\Delta t_i\right]\geq 2; \\ t_{over}=\max T_{endi} \\ =\max\left[\dfrac{D_i}{v_{L_i,p_{L_i}}}+\sum\left\{df_{n_j}\right\}+\left(p_{L_i}-1\right)\Delta t_i\right]\leq 3.5,\quad \forall 1\leq i\leq K,\forall n_j\in C_k; \end{cases} \tag{23}$$

确定比赛路线数目 K，根据策略得到 K 个比赛人员分组，根据之前模型建立的约束条件得到 K 条路线各自最小出发间隔，可以计算得到 K 条路线各自完成比赛需要时间，总完成时间即为 K 个完成时间的最大值，这样就得到了选定 K 的情况下最小比赛总时间。以完成比赛总时间 t_{over} 最小为优化目标，改进优化模型如下所示：

$$\min\left(t_{over}\right)=\min\left(\max T_{endi}\right);$$

$$s.t.\begin{cases} \sum_{k=1}^{K}\left(\sum\left\{h_{l_{m,n}}\right\}-\dfrac{1}{K}\sum_{k=1}^{K}\sum\left\{h_{l_{m,n}}\right\}\right)^2<\delta_{hth},\forall l_{m,n}\in L_k; \\ \sum_{k=1}^{K}\left(\sum\left\{d_{l_{m,n}}\right\}-\dfrac{1}{K}\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}\right)^2<\delta_{dth},\forall l_{m,n}\in L_k; \\ \sum_{k=1}^{K}\left(\sum\left\{df_{n_j}\right\}-\dfrac{1}{K}\sum_{k=1}^{K}\sum\left\{df_{n_j}\right\}\right)^2<\delta_{DFth},\forall n_j\in C_k; \\ \eta=\dfrac{\sum\left\{d_{l'_{m,n}}\right\}}{\sum_{k=1}^{K}\sum\left\{d_{l_{m,n}}\right\}}<\eta_{th},\quad \forall 1\leq k\leq k,l_{m,n}\in L_i,l'_{m,n}\in L_{repeat}; \\ \Delta t_k\geq\max\left(df_{n_j}\right),\forall 1\leq k\leq K,n_j\in C_k; \\ \begin{cases} a_{C_k}\geq 20; \\ C_k\bigcap C_0=C_0; \end{cases}\forall 1\leq k\leq K; \\ \sum_{k=1}^{K}u_{i,k}^{t}\leq 5,\forall 1\leq i\leq N,t\in\left[0,T_{endk}\right]; \\ \begin{cases} \dfrac{1}{K}\sum_{i=1}^{K}T_{endi}=\dfrac{1}{K}\sum_{i=1}^{K}\left[\dfrac{D_i}{v_{L_i,p_{L_i}}}+\sum\left\{df_{n_j}\right\}+\left(p_{L_i}-1\right)\Delta t_i\right]\geq 2; \\ \max\left[\dfrac{D_i}{v_{L_i,p_{L_i}}}+\sum\left\{df_{n_j}\right\}+\left(p_{L_i}-1\right)\Delta t_i\right]\leq 3.5,\quad \forall 1\leq i\leq K,\forall n_j\in C_k; \end{cases} \end{cases} \tag{24}$$

d) **算法改进**

在新的约束以及前提下，遗传算法中目标函数进而改进为

$$
\begin{aligned}
g = k_1 \sum_{k=1}^{K} & \left(\sum_{l_{ij} \in L_k} \sqrt{\left|x_i - x_j\right|^2 + \left|y_i - y_j\right|^2 + \left|z_i - z_j\right|^2} \right. \\
& \left. - \frac{1}{K} \sum_{k=1}^{K} \sum_{l_{ij} \in L_k} \sqrt{\left|x_i - x_j\right|^2 + \left|y_i - y_j\right|^2 + \left|z_i - z_j\right|^2} \right)^2 \\
& + k_2 \sum_{k=1}^{K} \left(\sum \left\{ df_{n_j} \right\} - \frac{1}{K} \sum_{k=1}^{K} \sum \left\{ df_{n_j} \right\} \right)^2 \\
& + k_3 \max \left[\frac{D_i}{v_{L_i, p_{L_i}}} + \sum \left\{ df_{n_j} \right\} + \left(p_{L_i} - 1 \right) \Delta t_i \right] \\
& + k_4 \sum_{k=1}^{K} \left(\sum \left\{ h_{l_{m,n}} \right\} - \frac{1}{K} \sum_{k=1}^{K} \sum \left\{ h_{l_{m,n}} \right\} \right)^2
\end{aligned}
\tag{25}
$$

4.3.2 结果与分析

综合考虑前两问人数、间隔与路线数的关系，在具有 120 人的情况下，我们取得以 10 为中心，上下波动的区间。

下面为当 K 分别取 8、9、10、11、12 时，具体路线设计方案：

在 $K=8$ 时，路线设计方案如下所示：

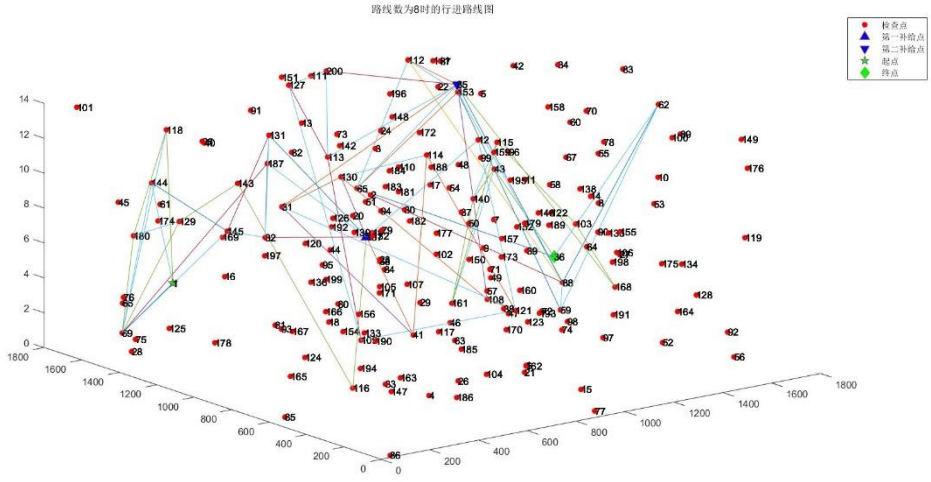


图 42 $K=8$ 路线设计图

该方案下各路线参赛选手分配如下所示：

表 15 各路线选手分组表

路线	选手数量	选手编号
1	15	101,32,91,56,48,37,4,67,79,73,80,62,69,103,86
2	15	104,45,107,64,84,68,21,71,108,85,97,99,72,105,98
3	15	33,54,18,116,94,75,22,88,3,89,112,102,81,120,110
4	15	106,19,47,118,96,78,27,90,7,26,5,115,100,39,60
5	15	15,63,117,9,113,83,30,114,36,38,6,35,111,55,17

6	15	93,2,1,10,14,87,44,119,46,57,13,49,82,76,42
7	15	20,41,8,11,23,95,58,31,50,61,16,52,92,77,53
8	15	28,51,40,34,29,109,65,43,70,74,24,66,59,12,25

该方案下各路线总长度及平均长度如下表所示：

表 16 各路线总长度表

路线	1	2	3	4	5	6	7	8	平均
长度(m)	6234	6061	7314	7272	7258	7779	7651	7678	7156

该方案下各路线总难度及平均难度如下表所示：

表 17 各路线总难度表

路线	1	2	3	4	5	6	7	8	平均
难度(min)	82	82	91	91	91	93	93	93	90

该方案下各路线总爬高量及平均爬高量如下表所示：

表 18 各路线总爬高量表

路线	1	2	3	4	5	6	7	8	平均
爬高量(m)	61	64	59	60	59	54	47	58	58

在 $K=9$ 时，路线设计方案如下所示：

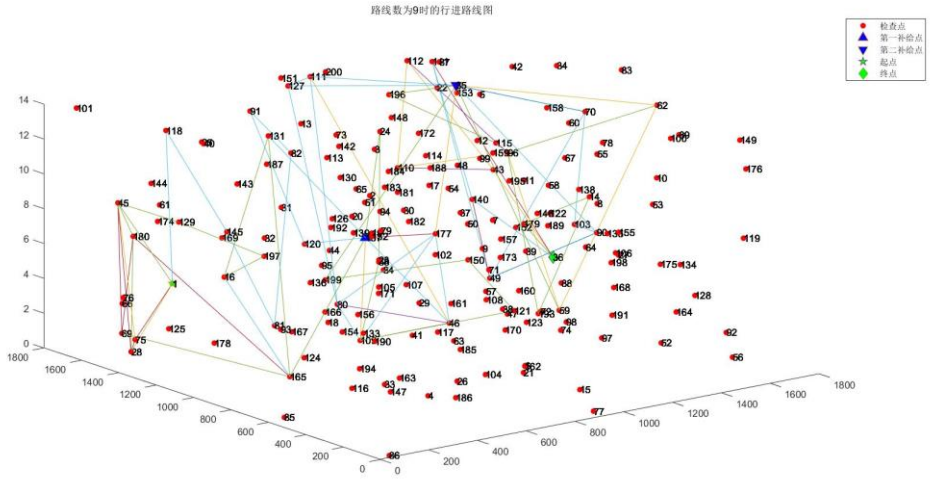


图 43 $K=9$ 路线设计图

该方案下各路线参赛选手分配如下所示：

表 19 各路线选手分组表

路线	选手数量	选手编号
1	14	101,106,63,1,11,29,4,71,3,26,6,49,92,12
2	14	104,15,2,8,34,37,21,88,7,38,13,52,59,86
3	14	33,93,41,40,48,68,22,90,36,57,16,66,103,98
4	13	20,51,56,84,75,27,114,46,61,24,69,105,110

5	13	28,91,64,94,78,30,119,50,74,62,72,120,60
6	13	32,107,116,96,83,44,31,70,80,99,81,39,17
7	13	45,18,118,113,87,58,43,73,97,102,100,55,42
8	13	54,47,9,14,95,65,79,85,112,115,111,76,53
9	13	19,117,10,23,109,67,108,89,5,35,82,77,25

该方案下各路线总长度及平均长度如下表所示：

表 20 各路线总长度表

路线	1	2	3	4	5	6	7	8	9	平均
长度(m)	6675	6728	6809	7071	5895	6077	5927	6315	6026	6391

该方案下各路线总难度及平均难度如下表所示：

表 21 各路线总难度表

路线	1	2	3	4	5	6	7	8	9	平均
难度(min)	92	90	90	90	89	89	89	87	88	89

该方案下各路线总爬高量及平均爬高量如下表所示：

表 22 各路线总爬高量表

路线	1	2	3	4	5	6	7	8	9	平均
爬高量(m)	53	42	50	58	56	51	51	51	46	51

在 $K=10$ 时，路线设计方案如下所示：

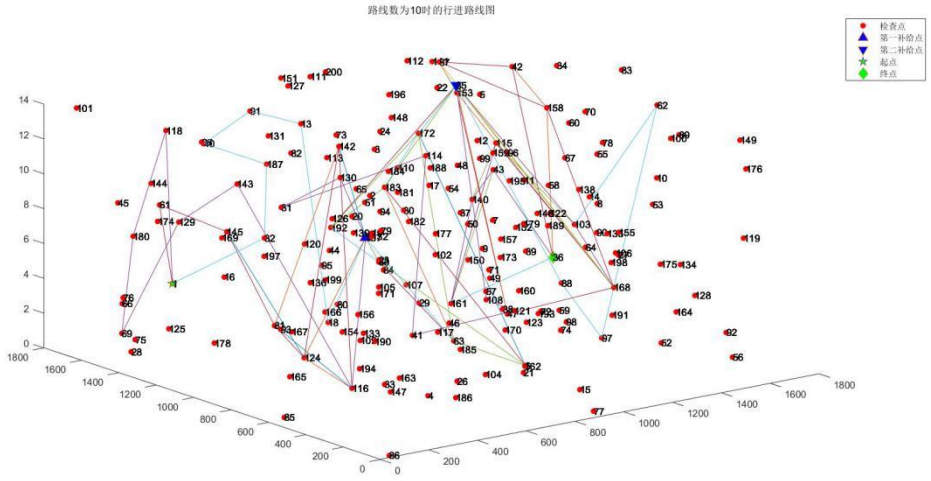


图 44 $K=10$ 路线设计图

该方案下各路线参赛选手分配如下所示：

表 23 各路线选手分组表

路线	选手数量	选手编号
1	12	101,54,117,11,37,22,114,50,80,102,111,77
2	12	104,19,1,34,68,27,119,70,97,115,82,12

3	12	33,63,8,48,75,30,31,73,112,35,92,86
4	12	106,2,40,84,78,44,43,85,5,49,59,98
5	12	15,41,56,94,83,58,79,89,6,52,103,110
6	12	93,51,64,96,87,65,108,26,13,66,105,60
7	12	20,91,116,113,95,67,3,38,16,69,120,17
8	12	28,107,118,14,109,71,7,57,24,72,39,42
9	12	32,18,9,23,4,88,36,61,62,81,55,53
10	12	45,47,10,29,21,90,46,74,99,100,76,25

该方案下各路线总长度及平均长度如下表所示：

表 24 各路线总长度表

路线	1	2	3	4	5	6	7	8	9	10	平均
长度(m)	5805	5946	6002	6249	5944	6255	6139	6218	6346	6228	6113

该方案下各路线总难度及平均难度如下表所示：

表 25 各路线总难度表

路线	1	2	3	4	5	6	7	8	9	10	平均
难度(min)	102	96	96	96	96	96	96	90	88	82	94

该方案下各路线总爬高量及平均爬高量如下表所示：

表 26 各路线总爬高量表

路线	1	2	3	4	5	6	7	8	9	10	平均
爬高量(m)	61	51	47	56	51	53	57	42	61	59	54

在 $K=11$ 时，路线设计方案如下所示：

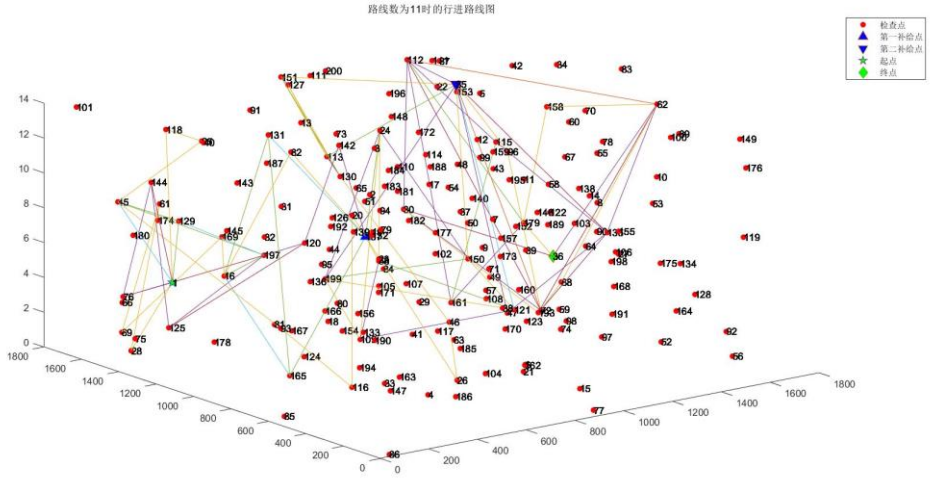


图 45 $K=11$ 路线设计图

该方案下各路线参赛选手分配如下所示：

表 27 各路线选手分组表

路线	选手数量	选手编号
1	11	101,54,1,48,78,58,108,38,24,81,76
2	11	104,19,8,84,83,65,3,57,62,100,77
3	11	33,63,40,94,87,67,7,61,99,111,12
4	11	106,2,56,96,95,71,36,74,102,82,86
5	11	15,41,64,113,109,88,46,80,115,92,98
6	11	93,51,116,14,4,90,50,97,35,59,110
7	11	20,91,118,23,21,114,70,112,49,103,60
8	11	28,107,9,29,22,119,73,5,52,105,17
9	11	32,18,10,37,27,31,85,6,66,120,42
10	11	45,47,11,68,30,43,89,13,69,39,53
11	10	117,34,75,44,79,26,16,72,55,25

该方案下各路线总长度及平均长度如下表所示：

表 28 各路线总长度表

路线	1	2	3	4	5	6	7	8	9	10	11	平均
长度(m)	6347	6427	6697	6925	6381	6293	7029	6293	6293	6461	6617	6524

该方案下各路线总难度及平均难度如下表所示：

表 29 各路线总难度表

路线	1	2	3	4	5	6	7	8	9	10	11	平均
难度(min)	88	88	84	86	88	88	88	88	88	88	85	87

该方案下各路线总爬高量及平均爬高量如下表所示：

表 30 各路线总爬高量表

路线	1	2	3	4	5	6	7	8	9	10	11	平均
爬高量(m)	55	66	64	55	51	50	56	50	50	46	47	54

在$K=12$时，路线设计方案如下所示：

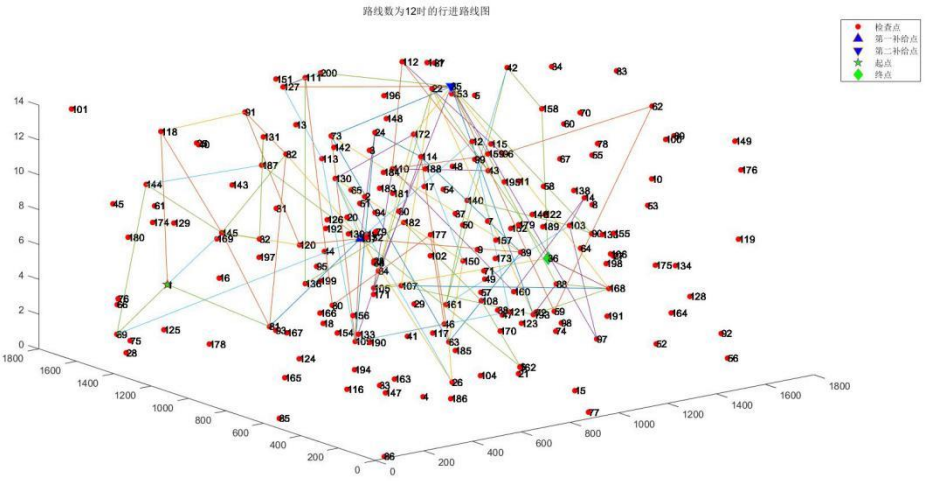


图 46 $K=12$ 路线设计图

该方案下各路线参赛选手分配如下所示：

表 31 各路线选手分组表

路线	选手数量	选手编号
1	10	101,63,56,113,4,114,73,6,69,55
2	10	104,2,64,14,21,119,85,13,72,76
3	10	33,41,116,23,22,31,89,16,81,77
4	10	106,51,118,29,27,43,26,24,100,12
5	10	15,91,9,37,30,79,38,62,111,86
6	10	93,107,10,68,44,108,57,99,82,98
7	10	20,18,11,75,58,3,61,102,92,110
8	10	28,47,34,78,65,7,74,115,59,60
9	10	32,117,48,83,67,36,80,35,103,17
10	10	45,1,84,87,71,46,97,49,105,42
11	10	54,8,94,95,88,50,112,52,120,53
12	10	19,40,96,109,90,70,5,66,39,25

该方案下各路线总长度及平均长度如下表所示：

表 32 各路线总长度表

路线	1	2	3	4	5	6	7	8	9	10	11	12	平均
长度(m)	6742	6819	6200	6130	6177	6285	5870	5905	5760	6559	6374	6675	6291

该方案下各路线总难度及平均难度如下表所示：

表 33 各路线总难度表

路线	1	2	3	4	5	6	7	8	9	10	11	12	平均

难度(min)	90	88	94	92	92	92	94	94	94	95	95	92	93

该方案下各路线总爬高量及平均爬高量如下表所示：

表 34 各路线总爬高量表

路线	1	2	3	4	5	6	7	8	9	10	11	12	平均
爬高量(m)	47	45	64	57	58	58	43	55	49	56	57	53	54

并可以优化得到不同比赛路线数目 K 取值下最小总完成时间如下图所示：

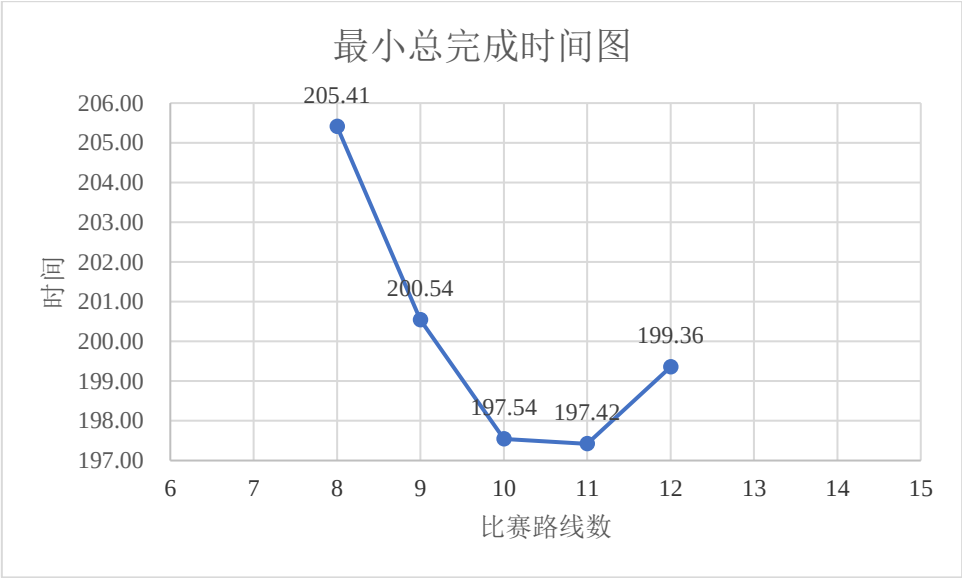


图 47 不同比赛路线数下最小完成时间图

根据对问题三分析，题目共有两个优化目标且难以结合为单优化目标，此时可以对两个优化目标“比赛路线数目”和“最小完成时间”求加权和为如下所示：

$$G=\lambda_1 \bullet K+\lambda_2 \bullet t_{over} \tag{26}$$

其中权值 λ_1 和 λ_2 由人为选定，且满足 $\lambda_1=1-\lambda_2$。

此时权值选定引入了人为主观因素带来的影响，但是结合实际情况，有时这种主观影响是有必要的。比如当天天气预报有恶劣天气出现，为了避免带来过大影响以及造成比赛选手受伤等，“比赛完成时间最短”更加重要，此时比赛完成时间前面权值应该更大一些；当物资资金紧张，难以支撑多条比赛路线同时开展时，“比赛路线最少”成为了更重要因素，此时比赛路线数目权值应该更大一些；在有些情况下，比赛完成时间及比赛路线数目给举办方带来额外的经济压力，这时要“比赛路线尽可能少”且“总完成时间尽量短”就可以根据实际关系转化成花费资金尽可能少来确定最佳方案。

因为比赛路线数目与最小完成时间数据相差较大，因此当权值选择为 $\lambda_1=0.6$、$\lambda_2=0.4$ 时，体现了一种较为折中的考虑，得到总代价 G 与比赛路线数目 K 关系曲线如下所示：

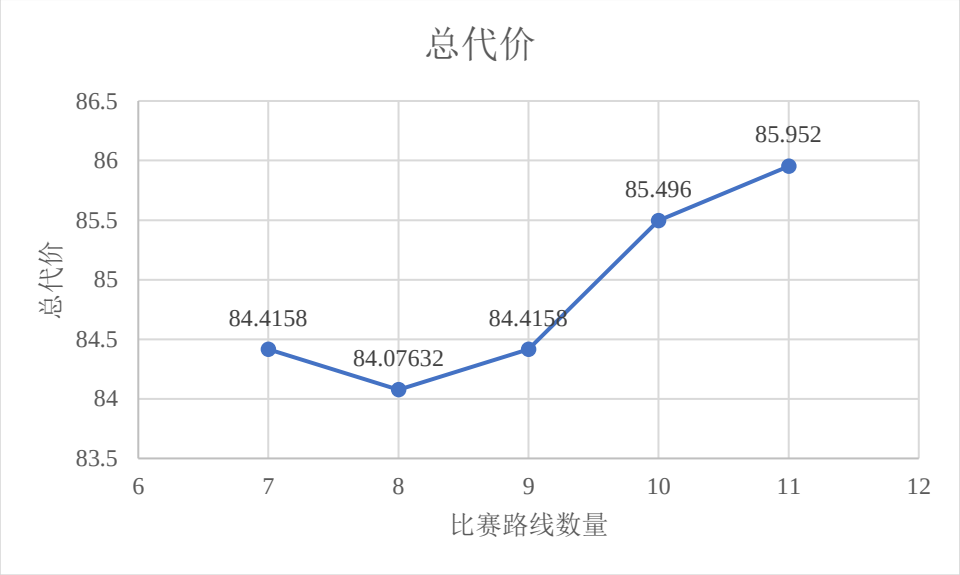


图 48 不同比赛路线数下总代价图

可以看到，在 $K=10$ 时总代价最小，为最佳方案，具体路线规划数据见“附件 4”。

五、总结与展望

本文以定向越野比赛路线设计为背景完成一个多路径规划问题，根据题目中各项要求将问题分为“必经点、终点选取”和“路径规划”两部分内容来求解。在必经点选取策略上更多考虑区域内所有检查点能较为方便的与必经点相连，用必经点到其他检查点距离之和表示代价，要求代价最小。并且考虑到要求选取两个必经点，两个必经点距离越近越不合理，用两个必经点间距离表示代价，距离越近，代价越大。最后综合考虑，以全局代价最小为目标，选取得到必经点。终点要求选取难度较小的点，选取过程中还考虑了题目要求路段重复率尽可能低，且同时到达同一节点参赛人员数目不超过 5 人的要求。因此，首先筛选出难度较小的检查点，得到终点待选解空间，然后以起点终点距离最大为目标选取终点。

在路径规划部分则是通过路径规划模型及遗传算法求解来实现，其中路径规划模型主要是为题目中各项约束条件建立数学模型，然后分别针对不同问题，以最大总参赛人数或者最小总比赛时间为总目标函数建立模型。在算法求解部分，先人为设定比赛路线数目，通过遗传算法优化得到最佳路线规划方案。并比较不同比赛路线数目下总参赛人数或总比赛时间，得到全局最优解及该情况下具体路线规划方案。

本题给出的硬性约束不多，因此解空间极其复杂。在求解过程中，我们把问题分为“点的选取”和“路径规划”两部分来完成，将“比赛长度尽可能相等”等约束条件转化为“不同路线长度方差小于阈值”等数学模型来实现，都极大地简化了问题难度，使问题变得可解。在算法求解部分使用“遗传算法”，通过设定合适的适应度函数、遗传变异方案等，利用了其求解速度快的特点，利于在及其复杂的解空间中快速找出局部最优或者全局最优解，也提高了算法的实用性及对不同情况的适应性。

但是还是有不足的地方，通过求解我们发现，有些路线存在连续通过多个距离较近的检查点，这虽然在题目中没有要求，但结合实际考虑，一条路线中相邻

两个检查点之间距离应尽可能相等，但在实际建模求解过程中引入一个新的约束可能导致算法性能下降，最优解难以得到，因此没有额外添加约束，在之后实际应用中可以更多添加一些符合客观实际的约束，通过足够多次的迭代，得到全局最优解。

参考文献

[1] 屈援,汪波,钟石泉.单车场多送货点车辆路径问题的改进遗传算法[J].计算机工程与应用,2007(25):237-239+243.
[2] 胡春磊,章飞,曾庆军.基于多目标蚁群策略的 AUV 全局路径规划算法[J].传感与微系统,2020,39(11):107-109+113.
[3] 李丹莲,曹倩,徐菲.基于改进遗传算法的连锁便利店配送路径优化[J].计算机工程与科学,2020,42(11):2096-2102.
[4] 张惠彬,黄顺祥,刘峰,关彩虹.基于遗传算法的人员疏散路径优化控制[J].工业安全与环保,2020,46(10):24-28.
[5] 刘建胜,谭文越.应用混合遗传算法的多集货中心多车型整车路径规划研究[J].机械设计与制造,2020(11):236-240.
[6] 丁承君,王鑫,冯玉伯,张家梁.基于粒子群优化算法的 AGV 路径规划[J].传感器与微系统,2020,39(08):123-126.
[7] 白秋产.基于改进 PSO 算法的 WSN 移动 Sink 路径规划算法[J].传感技术学报,2020,33(05):733-737.
[8] 涂铮铮. 基于进化和强化学习算法的动态路径规划研究[D].电子科技大学,2020.
[9] 鲍伟,贾江鸣,李湘生,周庆红.考虑软时间窗的多车型车辆配送路径优化[J].物流科技,2020,43(10):76-82.

附　录

问题一代码：

```
clc;
clear all;

a=1000;
b=0;
c=0;

%while a>1000 || b>50 || c<12000

delete 'lujin'.xls
close all



[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')
```

```
xy1=xlsread('D:\桌面\省建模\代码\第一问\xy1.xls')


  mtspof_galll(xy1,'sheet1');



xy2=xlsread('D:\桌面\省建模\代码\第一问\xy2.xls')
  mtspof_galll7(xy2,'sheet2');



xy3=xlsread('D:\桌面\省建模\代码\第一问\xy3.xls')
  mtspof_galll(xy3,'sheet3');


  function varargout=mtspof_galll(xy2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')

xy=xy2;


n=length(xy);
for i=1:n-2
       xyy(i,:)=xy(i+1,:)            %掐头去尾
end
xyt(1,1)=xy(1,3);


    nn=6;                      %基站点数

    t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

    for j=1:length(t)
          xyt(1,j+1)=xyy(t(j),3);
    end

xyt(1,length(t)+2)=xy(n,3);

for i=1:length(xyt)
       xytt(i,1)=data(xyt(1,i),2);            %(X,Y)形式
       xytt(i,2)=data(xyt(1,i),3);
       xytt(i,3)=xyt(i);
end
GA(xytt,sheet);
```

```
 end


 function varargout=mtspof_galll7(xy2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')
%xy2=xlsread('D:\桌面\省建模\代码\xy2.xls')
xy=xy2;


n=length(xy);
for i=1:n-2
      xyy(i,:)=xy(i+1,:)            %掐头去尾
end
xyt(1,1)=xy(1,3);


   nn=7;                    %基站点数

   t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

   for j=1:length(t)
        xyt(1,j+1)=xyy(t(j),3);
   end

xyt(1,length(t)+2)=xy(n,3);

for i=1:length(xyt)
      xytt(i,1)=data(xyt(1,i),2);          %(X,Y)形式
      xytt(i,2)=data(xyt(1,i),3);
      xytt(i,3)=xyt(i);
end
GA(xytt,sheet);




 end


 function []=GA(xy2,sheet)
for ss=1:3
xy=xy2;
n=length(xy);
for i=1:n
      xx(i,1)=xy(i,1);
      xx(i,2)=xy(i,2);
      x(i,1)=xy(i,1);
      y(i,1)=xy(i,2);
```

```
end


NIND = 100;              %种群大小
MAXGEN = 10;             %最大迭代次数
Pc = 0.9;                %交叉概率，相当于基因遗传的时候染色体交叉
Pm = 0.05;               %染色体变异
GGAP = 0.9;              %这个是代沟，通过遗传方式得到的子代数为父代数
*GGAP
D = Distance(xx);        %通过这个函数可以计算 i,j 两点之间的距离
N = size(D,1);           %计算有多少个坐标点
%%初始化种群
Chrom = InitPop(NIND,N);       %Chrome 代表的种群



%%%
%DrawPath(Chrom(1,:),xx)
%pause(0.0001)
%输出随机解的路线和总距离
  disp('初始种群中的一个随机值')
OutputPath(Chrom(1,:));%其中一个个体
Rlength = PathLength(D,Chrom(1,:));
disp(['总距离:',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%优化
gen = 0;
figure;
hold on;
box on;
xlim([0,MAXGEN])
title('优化过程')
xlabel('迭代次数')
ylabel('当前最优解')
ObjV = PathLength(D,Chrom);%计算当前路线长度，即上面随机产生的那些个体
路径
preObjV = min(ObjV);%当前最优解
while gen<MAXGEN
    %%计算适应度
    ObjV = PathLength(D,Chrom);          %计算路线长度
    line([gen - 1,gen],[preObjV,min(ObjV)]);
    pause(0.0001);
    preObjV = min(ObjV);
    FitnV = Fitness(ObjV);
    %选择
    SelCh = Select(Chrom,FitnV,GGAP);
```

```
        %交叉操作
        SelCH = Recombin(SelCh,Pc);
        %变异
        SelCh = Mutate(SelCh,Pm);
        %逆转操作
        SelCh = Reverse(SelCh,D);
        %重插入子代的新种群
        Chrom = Reins(Chrom,SelCh,ObjV);
        %更新迭代次数
        gen = gen + 1;
end

for i=1:NIND
th1=Chrom(i,1);
th2=Chrom(i,n);
label1=find(Chrom(i,:)==1);
label2=find(Chrom(i,:)==n);

Chrom(i,1)=1; Chrom(i,label1)=th1;
Chrom(i,n)=n; Chrom(i,label2)=th2;
end



%画出最优解的路线图
ObjV = PathLength(D,Chrom);          %计算路线长度
[minObjV,minInd] = min(ObjV);
%DrawPath(Chrom(minInd(1),:),xx)
%hold on



for i=1:n
lujin(1,i)=xy(Chrom(minInd(1),i),3);
end
str=['A1';'A2';'A3';'A4';'A5'];
xlswrite('lujin.xls', lujin,sheet,str(ss,:))
%%输出最优解的路线和距离
disp('最优解:')
p = OutputPath(Chrom(minInd(1),:));
disp(['：',num2str(ObjV(minInd(1)))]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
end


  end
```

```
 function [D]=DIS(x1,x2,y1,y2)
D=((x1-x2)^2+(y1-y2)^2)^0.5;
 end

 function D = Distance(a)
%

row = size(a,1);
D = zeros(row,row);
for i = 1:row
    for j = i+1:row
        D(i,j) = ((a(i,1) - a(j,1))^2 + (a(i,2)-a(j,2))^2)^0.5;
        D(j,i) = D(i,j);
    end
end
 end

function DrawPath(Chrom,X)
%输入：
%待画路线
%的坐标位置
%输出：
%
n=length(Chrom);
R =  [Chrom(1,:) Chrom(1,n)]; %第一个随机解
figure;
hold on
plot(X(:,1),X(:,2),'bo')%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐标
%plot(X(:,1),X(:,2),'o','color',[1,1,1])%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐标，
plot(X(Chrom(1,1),1),X(Chrom(1,1),2),'rv','MarkerSize',20)%标记起点
A = X(R,:);                                    %A 是将之前的坐标顺序用 R 打乱后重新存入 A 中
row = size(A,1);                               %row 为坐标数+1
for i = 2:row
    [arrowx,arrowy] = dsxy2figxy(gca,A(i-1:i,1),A(i-1:i,2));      %dsxy2figxy 坐标转换函数，记录两个点
    annotation('textarrow',arrowx,arrowy,'HeadWidth',8,'color',[0,0,1]);% 将这两个点连接起来
end
hold off
xlabel('横坐标 x')
ylabel('纵坐标 y')
title('图')
box on

end
```

```
function varargout = dsxy2figxy(varargin) %%坐标转换函数
if length(varargin{1}) == 1 && ishandle(varargin{1}) ...
                                    && strcmp(get(varargin{1},'type'),'axes')
    hAx = varargin{1};
    varargin = varargin(2:end);
else
    hAx = gca;
end
if length(varargin) == 1
    pos = varargin{1};
else
    [x,y] = deal(varargin{:});
end
axun = get(hAx,'Units');
set(hAx,'Units','normalized');
axpos = get(hAx,'Position');
axlim = axis(hAx);
axwidth = diff(axlim(1:2));
axheight = diff(axlim(3:4));
if exist('x','var')
    varargout{1} = (x - axlim(1)) * axpos(3) / axwidth + axpos(1);
    varargout{2} = (y - axlim(3)) * axpos(4) / axheight + axpos(2);
else
    pos(1) = (pos(1) - axlim(1)) / axwidth * axpos(3) + axpos(1);
    pos(2) = (pos(2) - axlim(3)) / axheight * axpos(4) + axpos(2);
    pos(3) = pos(3) * axpos(3) / axwidth;
    pos(4) = pos(4) * axpos(4 )/ axheight;
    varargout{1} = pos;
end
set(hAx,'Units',axun)
end

function FitnV = Fitness(len) %适应度函数
%输入：
%len  个体的长度（TSP 的距离）
%输出：
%FitnV  个体的适应度值
FitnV = 1./len;
end


function [D,path,min1,path1]=floyd(a,start,terminal)
D=a;n=size(D,1);path=zeros(n,n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;
        end,
```

```
    end,
end
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end,
        end,
    end,
end
if nargin==3
    min1=D(start,terminal);
    m(1)=start;
    i=1;
    path1=[ ];
    while     path(m(i),terminal)~=terminal
        k=i+1;
        m(k)=path(m(i),terminal);
        i=i+1;
    end
    m(i+1)=terminal;
    path1=m;
end
end


function Chrom = InitPop(NIND,N)%初始化种群，
%输入：
%NIND：种群大小
%N：个体染色体长度（城市个数）
%输出：
%初始种群
Chrom = zeros(NIND,N);   %用于存储种群

for i = 1:NIND

      Chrom(i,:) = randperm(N);%随机生成初始种群,randperm 函数的用法是返回一行 1~N 的整数，这 N 个数是不同的

end
end

function [a,b] = intercross(a,b)
%输入：
%a 和 b 为两个待交叉的个体
%输出：
```

```
%a 和 b 为交叉后得到的两个个体
L = length(a);
r1 = randsrc(1,1,[1:L]);
r2 = randsrc(1,1,[1:L]);
if r1~=r2
    a0 = a;
    b0 = b;
    s = min([r1,r2]);
    e = max([r1,r2]);
    for i =s:e
        a1 = a;
        b1 = b;
        a(i) = b0(i);
        b(i) = a0(i);
        x = find(a==a(i));
        y = find(b==b(i));
        i1 = x(x~=i)
        i2 = y(y~=i);
        if ~isempty(i1)
            a(i1)=a1(i);
        end
        if ~isempty(i2)
            b(i1)=b1(i);
        end
    end
end
end

function [luu]=LUJIN(lu1,lu2,lu3,ts)
jj1=length(lu1);
jj2=length(lu2);
jj3=length(lu3);
kk=1;
    for j=1:jj1
        if isnan(lu1(ts,j))== 0
            luu(1,kk)=lu1(ts,j);
            kk=kk+1;
        end
    end
    for j=1:jj2
        if isnan(lu2(ts,j))== 0
            luu(1,kk)=lu2(ts,j);
            kk=kk+1;
        end
    end
     for j=1:jj3
        if isnan(lu3(ts,j))== 0
            luu(1,kk)=lu3(ts,j);
            kk=kk+1;
        end
```

```
        end
end


function SelCh = Mutate(SelCh,Pm)
%变异操作
%输入：
%SelCh   被选择的个体
%Pm   变异概率
%输出：
%SelCh   变异后的个体

[NSel,L] = size(SelCh);
for i = 1:NSel
      if Pm >= rand
            R = randperm(L);
            SelCh(i,R(1:2)) = SelCh(i,R(2:-1:1));
      end
end
end


function p=OutputPath(R)
%打印路线函数
%以 1->2->3 的形式在命令行打印路线

R = [R];
N = length(R);
p = num2str(R(1));
for i = 2:N
      p = [p,'->',num2str(R(i))];
end
disp(p)
end


function len = PathLength(D,Chrom)
%计算所有个体的路线长度
%输入
%D
%Chrom 个体的轨迹

[~,col] = size(D); %返回 D 的列数
NIND = size(Chrom,1);%NIND 等于初始种群
len = zeros(NIND,1);%初始化一个大小等于 NOND 的 len 来记录长度
for i = 1:NIND
      p = [Chrom(i,:) Chrom(i,1)];%构造 p 矩阵保存路线图 将第一行路线提出 再
加上第一个构成回路
```

```
        i1 = p(1:end-1);%i1 从第一个开始遍历到倒数第二个
        i2 = p(2:end);%i2 从第二个开始遍历到倒数第一个
        len(i,1) = sum(D((i1-1)*col+i2));%计算出每种路线（初始种群的个体）的长度
end
end

function SelCh = Recombin(SelCh,Pc)
%交叉操作
%输入：
%SelCh  被选择的个体
%Pc     交叉概率
%输出：
%SelCh  交叉后的个体

NSel = size(SelCh,1);
for i = 1:2:NSel - mod(NSel,2)
        if Pc>=rand %交叉概率 PC
                [SelCh(i,:),SelCh(i+1,:)] = intercross(SelCh(i,:),SelCh(i+1,:));
        end
end
end

function Chrom = Reins(Chrom,SelCh,ObjV)
%%重插入子代的种群
%输入：
%Chrom          父代的种群
%SelCh          子代的种群
%ObjV           父代适应度
%输出：
%Chrom          组合父代与子代后得到的新种群
NIND = size(Chrom,1);
NSel = size(SelCh,1);
[TobjV,index] = sort(ObjV);
Chrom =    [Chrom(index(1:NIND-NSel),:);SelCh];
end

function SelCh = Reverse(SelCh,D)
%进化逆转函数
%输入：
%SelCh    被选择的个体
%D
%输出：
%SelCh    进化逆转后的个体

[row,col] = size(SelCh);
ObjV = PathLength(D,SelCh);
SelCh1 = SelCh;
```

```
for i = 1:row
    r1 = randsrc(1,1,[1:col]);
    r2 = randsrc(1,1,[1:col]);
    mininverse = min([r1 r2]);
    maxinverse = max([r1 r2]);
    SelCh1(i,mininverse:maxinverse) = SelCh1(i,maxinverse:-1:mininverse);
end
ObjV1 = PathLength(D,SelCh1);%计算路线长度
index = ObjV1<ObjV;
SelCh(index,:)=SelCh1(index,:);
end

function SelCh = Select(Chrom,FitnV,GGAP)
%输入：
%Chrom 种群
%FitnV 适应度值
%GGAP 选择概率
%输出：
%SelCh 被选择的个体
NIND = size(Chrom,1);%种群个体总数
NSel = max(floor(NIND * GGAP+.5),2);%确定下一代种群的个体数，如果不足二个就计为二个
ChrIx = Sus(FitnV,NSel);
SelCh = Chrom(ChrIx,:);
end

function NewChrIx = Sus(FitnV,Nsel)
%输入：
%FitnV 个体的是适应度值
%Nsel 被选个体的数目
%输出：
%NewChrIx 被选择个体的索引号
[Nind,ans] = size(FitnV);%Nind 为 FitnV 的行数也就是个体数 ans 为列数 1
cumfit = cumsum(FitnV);%对适应度累加 例如 1 2 3 累加后 1 3 6 用来计算累积概率
trials = cumfit(Nind)/Nsel * (rand + (0:Nsel-1)');%cumfit(Nind)表示的是矩阵 cumfit 的第 Nind 个元素 A.'是一般转置，A'是共轭转置 rand 返回一个在区间 (0,1) 内均匀分布的随机数
%cumfit(Nind)/Nsel 平均适应度
Mf = cumfit(:,ones(1,Nsel));%将生成的累积概率 复制 90 份 生成一个 100*90 的矩阵
Mt = trials(:,ones(1,Nind))';
[NewChrIx,ans] = find(Mt<Mf & [zeros(1,Nsel);Mf(1:Nind-1,:)]<= Mt);%寻找非零元素
[ans,shuf] = sort(rand(Nsel,1));%生成 Nsel*1 的随机数矩阵 按升序对 A 的元素进行排序 返回选择的 shuf
```

```
NewChrIx = NewChrIx(shuf);%将 shuf 返回
end

function [dis]=ZDIS(luu,data)
n=length(luu);
dis=0;
for i=1:n-1
      x1=data(luu(i),2); y1=data(luu(i),3);
      x2=data(luu(i+1),2);y2=data(luu(i+1),3);
      dis=dis+DIS(x1,x2,y1,y2);

end
end

function [znan]=ZN(luu,data)
n=length(luu);
znan=0;
for i=1:n

      znan=znan+data(luu(i),4);

end
end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')


% %%%画基底图
% n=length(data);
% x=[];y=[];
% label={};
% for i=1:n
%       x(i,1)=data(i,2);
%       y(i,1)=data(i,3);
%       label(1,i)=raw(i+1,1);
% end
%
% %for k=1:100
% k=41;
% ss=[];
% v=1;
% for i=2:100
%       for j=i+1:100
%             s=0;
%             iix=x(i,1);iiy=y(i,1);
%             jjx=x(j,1);jjy=y(j,1);
%             dij=DIS(iix,jjx,iiy,jjy);
%             for u=1:100
%                   uux=x(u,1);uuy=y(u,1);
%                   diu=DIS(iix,uux,iiy,uuy);
```

```
%                     dju=DIS(jjx,uux,jjy,uuy);
%                     s=s+diu+dju;
%                 end
%                 ss(v,1)=s-(2+k)*dij;
%                 ss(v,2)=i;ss(v,3)=j;
%                 v=v+1;
%         end
% end
% [w,ws]=min(ss(:,1));                        %ws 为 max 的索引值
% iz=ss(ws,2); jz=ss(ws,3);
%
%    figure(1)
%    line(x,y);
% plot(x,y,'r.','MarkerSize',20);
% hold on
% line(x(iz),y(iz));
% plot(x(iz),y(iz),'b.','MarkerSize',20);         %第一个比打卡点
% hold on
% line(x(jz),y(jz));
% plot(x(jz),y(jz),'b.','MarkerSize',20);         %第二个比打卡点
% text(x+0.02,y+0.02,label);
% hold on



lu1=xlsread('D:\桌面\省建模\解答 1.xls','sheet1')
lu2=xlsread('D:\桌面\省建模\解答 1.xls','sheet2')
lu3=xlsread('D:\桌面\省建模\解答 1.xls','sheet3')


n=size(lu1);

lu=[lu1(:,1:n(1,2)-1) lu2(:,1:n(1,2)-1) lu3];
for i=1:n(1,1)
      diss=ZDIS(lu(i,:),data);
      nann=ZN(lu(i,:),data);
      luu(i,:)=[lu(i,:) diss nann];
end


xlswrite('最终解答.xls', luu,'sheet1')

% %%%画三合一图
% for i=1:n(1,1)
%         for j=1:length(lu)
%         xxx(j,1)=data(lu(i,j),2);
%         yyy(j,1)=data(lu(i,j),3);
%
%         end
```

```
%
%        plot(xxx,yyy);
%        hold on
% end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')

lu=xlsread('D:\桌面\省建模\最终终解答.xls')
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 1.xlsx')
n=length(data);
%x=[];y=[];
label={};
for i=1:n
      x(i,1)=data(i,2);
      y(i,1)=data(i,3);
      label(1,i)=raw(i+1,1);
end

%for k=1:100
k=41;
ss=[];
v=1;
for i=2:100
      for j=i+1:100
            s=0;
            iix=x(i,1);iiy=y(i,1);
            jjx=x(j,1);jjy=y(j,1);
            dij=DIS(iix,jjx,iiy,jjy);
            for u=1:100
                  uux=x(u,1);uuy=y(u,1);
                  diu=DIS(iix,uux,iiy,uuy);
                  dju=DIS(jjx,uux,jjy,uuy);
                  s=s+diu+dju;
            end
            ss(v,1)=s-(2+k)*dij;
            ss(v,2)=i;ss(v,3)=j;
            v=v+1;
      end
end
[w,ws]=min(ss(:,1));                        %ws 为 max 的索引值
iz=ss(ws,2); jz=ss(ws,3);

 %figure(k)
%    line(x,y);
% plot(x,y,'r.','MarkerSize',20);
% hold on
% line(x(iz),y(iz));
% plot(x(iz),y(iz),'b.','MarkerSize',20);        %第一个比打卡点
% hold on
```

```
% line(x(jz),y(jz));
% plot(x(jz),y(jz),'b.','MarkerSize',20);        %第二个比打卡点
% text(x+0.02,y+0.02,label);
n=22;
nn=size(lu);
for i=1:nn(1,1)
        figure(i)
        for j=1:n
        xxxx(j,1)=data(lu(i,j),2);
        yyyy(j,1)=data(lu(i,j),3);
        %z(j,1)=data(lu(i,j),4);
        end
        line(x,y);
plot(x,y,'r.','MarkerSize',20);
hold on
line(x(iz),y(iz));
plot(x(iz),y(iz),'b.','MarkerSize',20);        %第一个比打卡点
hold on
line(x(jz),y(jz));
plot(x(jz),y(jz),'b.','MarkerSize',20);        %第二个比打卡点
text(x+0.02,y+0.02,label);
hold on
        plot(xxxx,yyyy);
        hold on
end
```

问题二代码：

```
clc;
clear all;

a=1000;
b=0;
c=0;

%while a>1000 || b>50 || c<12000

delete 'lujin1'.xls
close all



[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')
xyz1=xlsread('D:\桌面\省建模\代码\第二问\xyz1.xls')


  mtspof_galll7(xyz1,'sheet1');
```

```
xyz2=xlsread('D:\桌面\省建模\代码\第二问\xyz2.xls')
 mtspof_galll7(xyz2,'sheet2');



xyz3=xlsread('D:\桌面\省建模\代码\第二问\xyz3.xls')
 mtspof_galll(xyz3,'sheet3');

 function varargout=mtspof_galll(xyz2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')
%xy2=xlsread('D:\桌面\省建模\代码\xy2.xls')
xyz=xyz2;


n=length(xyz);
for i=1:n-2
      xyy(i,:)=xyz(i+1,:)            %掐头去尾
end
xyt(1,1)=xyz(1,4);


   nn=6;                    %基站点数

   t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

   for j=1:length(t)
        xyt(1,j+1)=xyy(t(j),4);
   end

xyt(1,length(t)+2)=xyz(n,4);

for i=1:length(xyt)
      xytt(i,1)=data(xyt(1,i),2);            %(X,Y)形式
      xytt(i,2)=data(xyt(1,i),3);
      xytt(i,3)=data(xyt(1,i),4);
      xytt(i,4)=xyt(i);
end
GA(xytt,sheet);



 end

 function varargout=mtspof_galll(xyz2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')
%xy2=xlsread('D:\桌面\省建模\代码\xy2.xls')
```

```
xyz=xyz2;


n=length(xyz);
for i=1:n-2
      xyy(i,:)=xyz(i+1,:)            %掐头去尾
end
xyt(1,1)=xyz(1,4);


   nn=7;                    %基站点数

   t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

   for j=1:length(t)
        xyt(1,j+1)=xyy(t(j),4);
   end

xyt(1,length(t)+2)=xyz(n,4);

for i=1:length(xyt)
      xytt(i,1)=data(xyt(1,i),2);            %(X,Y)形式
      xytt(i,2)=data(xyt(1,i),3);
      xytt(i,3)=data(xyt(1,i),4);
      xytt(i,4)=xyt(i);
end
GA(xytt,sheet);




 end

 function []=GA(xyz2,sheet)
for ss=1:10
xyz=xyz2;
n=length(xyz);
for i=1:n
      xx(i,1)=xyz(i,1);
      xx(i,2)=xyz(i,2);
      xx(i,3)=xyz(i,3);
      x(i,1)=xyz(i,1);
      y(i,1)=xyz(i,2);
      z(i,1)=xyz(i,3);
end

NIND = 100;              %种群大小
MAXGEN = 10;             %最大迭代次数
Pc = 0.9;                %交叉概率，相当于基因遗传的时候染色体交叉
```

```
Pm = 0.05;                  %染色体变异
GGAP = 0.9;                 %这个是代沟，通过遗传方式得到的子代数为父代数
*GGAP
D = Distance(xx);       %通过这个函数可以计算 i,j 两点之间的距离
N = size(D,1);          %计算有多少个坐标点
%%初始化种群
Chrom = InitPop(NIND,N);        %Chrome 代表的种群



%%画出随机解得路线图
%DrawPath(Chrom(1,:),xx)
%pause(0.0001)
%输出随机解的路线和总距离
  disp('初始种群中的一个随机值')
OutputPath(Chrom(1,:));%其中一个个体
Rlength = PathLength(D,Chrom(1,:));
disp(['总距离;',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%优化
gen = 0;
figure;
hold on;
box on;
xlim([0,MAXGEN])
title('优化过程')
xlabel('迭代次数')
ylabel('当前最优解')
ObjV = PathLength(D,Chrom);%计算当前路线长度，即上面随机产生的那些个体
路径
preObjV = min(ObjV);%当前最优解
while gen<MAXGEN
        %%计算适应度
        ObjV = PathLength(D,Chrom);         %计算路线长度
        line([gen - 1,gen],[preObjV,min(ObjV)]);
        pause(0.0001);
        preObjV = min(ObjV);
        FitnV = Fitness(ObjV);
        %选择
        SelCh = Select(Chrom,FitnV,GGAP);
        %交叉操作
        SelCH = Recombin(SelCh,Pc);
        %变异
        SelCh = Mutate(SelCh,Pm);
        %逆转操作
        SelCh = Reverse(SelCh,D);
```

```
    %重插入子代的新种群
    Chrom = Reins(Chrom,SelCh,ObjV);
    %更新迭代次数
    gen = gen + 1;
end

for i=1:NIND
th1=Chrom(i,1);
th2=Chrom(i,n);
label1=find(Chrom(i,:)==1);
label2=find(Chrom(i,:)==n);

Chrom(i,1)=1; Chrom(i,label1)=th1;
Chrom(i,n)=n; Chrom(i,label2)=th2;
end




%画出最优解的路线图
ObjV = PathLength(D,Chrom);         %计算路线长度
[minObjV,minInd] = min(ObjV);
%DrawPath(Chrom(minInd(1),:),xx)
%hold on



for i=1:n
lujin(1,i)=xyz(Chrom(minInd(1),i),4);
end
str=['A'];
str=[str num2str(ss)];
%str=['A1','A2','A3','A4','A5','A6','A7','A8','A9','A10','A11A12A13A14A15A16A17A
18A19A20A21A22A23A24A25A26A27A28A29A30'];
%str=['A1';'A2';'A3';'A4';'A5';'A6';'A7';'A8';'A9';'A10';'A11';'A12';'A13';'A14';'A15';'A
16';'A17';'A18';'A19';'A20';'A21';'A22';'A23';'A24';'A25';'A26';'A27';'A28';'A29';'A30'];

xlswrite('lujin1.xls',
lujin,sheet,str)                                %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

%%输出最优解的路线和距离
disp('最优解:')
p = OutputPath(Chrom(minInd(1),:));
disp(['xxxxx',num2str(ObjV(minInd(1)))]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
end
```

```
 end

 function [D]=DIS(x1,x2,y1,y2,z1,z2)
D=((x1-x2)^2+(y1-y2)^2+(z1-z2)^2)^0.5;
 end

 function D = Distance(a)


row = size(a,1);
D = zeros(row,row);
for i = 1:row
    for j = i+1:row
        D(i,j) = ((a(i,1) - a(j,1))^2 + (a(i,2)-a(j,2))^2+(a(i,3)-a(j,3))^2)^0.5;
        D(j,i) = D(i,j);
    end
end
 end

 function DrawPath(Chrom,X)
%输入：
%待画路线
%坐标位置
%输出：
%
n=length(Chrom);
R =  [Chrom(1,:) Chrom(1,n)]; %第再回到起点”
figure;
hold on
plot3(X(:,1),X(:,2),X(:,3),'bo')%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐标
%plot(X(:,1),X(:,2),'o','color',[1,1,1])%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐标，
plot3(X(Chrom(1,1),1),X(Chrom(1,1),2),X(Chrom(1,1),3),'rv','MarkerSize',20)%标记起点
A = X(R,:);                                    %A 是将之前的坐标顺序用 R 打乱后重新存入 A 中
row = size(A,1);                               %row 为坐标数+1
for i = 2:row
    [arrowx,arrowy,arrowz]          =          dsxy2figxy(gca,A(i-1:i,1),A(i-1:i,2),A(i-1:i,3));      %dsxy2figxy 坐标转换函数，记录两个点
    annotation('textarrow',arrowx,arrowy,arrowz,'HeadWidth',8,'color',[0,0,1]);%  将这两个点连接起来
end
hold off
xlabel('横坐标 x')
ylabel('纵坐标 y')
zlabel('纵坐标 z')
```

```
title('图')
box on
 end

 function varargout = dsxy2figxy(varargin) %%坐标转换函数
if length(varargin{1}) == 1 && ishandle(varargin{1}) ...
                                        && strcmp(get(varargin{1},'type'),'axes')
      hAx = varargin{1};
      varargin = varargin(2:end);
else
      hAx = gca;
end
if length(varargin) == 1
      pos = varargin{1};
else
      [x,y,z] = deal(varargin{:});
end
axun = get(hAx,'Units');
set(hAx,'Units','normalized');
axpos = get(hAx,'Position');
axlim = axis(hAx);
axwidth = diff(axlim(1:2));
axheight = diff(axlim(3:4));
if exist('x','var')
      varargout{1} = (x - axlim(1)) * axpos(3) / axwidth + axpos(1);
      varargout{2} = (y - axlim(3)) * axpos(4) / axheight + axpos(2);
else
      pos(1) = (pos(1) - axlim(1)) / axwidth * axpos(3) + axpos(1);
      pos(2) = (pos(2) - axlim(3)) / axheight * axpos(4) + axpos(2);
      pos(3) = pos(3) * axpos(3) / axwidth;
      pos(4) = pos(4) * axpos(4 )/ axheight;
      varargout{1} = pos;
end
set(hAx,'Units',axun)
 end

 function FitnV = Fitness(len) %适应度函数
%输入：
%len  个体的长度（TSP 的距离）
%输出：
%FitnV  个体的适应度值
FitnV = 1./len;
 end

 function [D,path,min1,path1]=floyd(a,start,terminal)
D=a;n=size(D,1);path=zeros(n,n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
```

```
            path(i,j)=j;
        end,
    end,
end
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end,
        end,
    end,
end
if nargin==3
    min1=D(start,terminal);
    m(1)=start;
    i=1;
    path1=[ ];
    while     path(m(i),terminal)~=terminal
        k=i+1;
        m(k)=path(m(i),terminal);
        i=i+1;
    end
    m(i+1)=terminal;
    path1=m;
end
 end

 function []=GA(xyz2,sheet)
for ss=1:10
xyz=xyz2;
n=length(xyz);
for i=1:n
      xx(i,1)=xyz(i,1);
      xx(i,2)=xyz(i,2);
      xx(i,3)=xyz(i,3);
      x(i,1)=xyz(i,1);
      y(i,1)=xyz(i,2);
      z(i,1)=xyz(i,3);
end

NIND = 100;              %种群大小
MAXGEN = 10;             %最大迭代次数
Pc = 0.9;               %交叉概率，相当于基因遗传的时候染色体交叉
Pm = 0.05;               %染色体变异
GGAP =  0.9;             %这个是代沟，通过遗传方式得到的子代数为父代数
*GGAP
D = Distance(xx);       %通过这个函数可以计算 i,j 两点之间的距离
```

```
N = size(D,1);          %计算有多少个坐标点
%%初始化种群
Chrom = InitPop(NIND,N);       %Chrome 代表的种群




%%画出随机解得路线图
%DrawPath(Chrom(1,:),xx)
%pause(0.0001)
%输出随机解的路线和总距离
 disp('初始种群中的一个随机值')
OutputPath(Chrom(1,:));%其中一个个体
Rlength = PathLength(D,Chrom(1,:));
disp(['总距离;',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%优化
gen = 0;
figure;
hold on;
box on;
xlim([0,MAXGEN])
title('优化过程')
xlabel('迭代次数')
ylabel('当前最优解')
ObjV = PathLength(D,Chrom);%计算当前路线长度，即上面随机产生的那些个体路径
preObjV = min(ObjV);%当前最优解
while gen<MAXGEN
    %%计算适应度
    ObjV = PathLength(D,Chrom);         %计算路线长度
    line([gen - 1,gen],[preObjV,min(ObjV)]);
    pause(0.0001);
    preObjV = min(ObjV);
    FitnV = Fitness(ObjV);
    %选择
    SelCh = Select(Chrom,FitnV,GGAP);
    %交叉操作
    SelCH = Recombin(SelCh,Pc);
    %变异
    SelCh = Mutate(SelCh,Pm);
    %逆转操作
    SelCh = Reverse(SelCh,D);
    %重插入子代的新种群
    Chrom = Reins(Chrom,SelCh,ObjV);
    %更新迭代次数
    gen = gen + 1;
```

```
end

for i=1:NIND
th1=Chrom(i,1);
th2=Chrom(i,n);
label1=find(Chrom(i,:)==1);
label2=find(Chrom(i,:)==n);

Chrom(i,1)=1; Chrom(i,label1)=th1;
Chrom(i,n)=n; Chrom(i,label2)=th2;
end




%画出最优解的路线图
ObjV = PathLength(D,Chrom);          %计算路线长度
[minObjV,minInd] = min(ObjV);
%DrawPath(Chrom(minInd(1),:),xx)
%hold on



for i=1:n
lujin(1,i)=xyz(Chrom(minInd(1),i),4);
end
str=['A'];
str=[str num2str(ss)];
%str=['A1','A2','A3','A4','A5','A6','A7','A8','A9','A10','A11A12A13A14A15A16A17A
18A19A20A21A22A23A24A25A26A27A28A29A30'];
%str=['A1';'A2';'A3';'A4';'A5';'A6';'A7';'A8';'A9';'A10';'A11';'A12';'A13';'A14';'A15';'A
16';'A17';'A18';'A19';'A20';'A21';'A22';'A23';'A24';'A25';'A26';'A27';'A28';'A29';'A30'];

xlswrite('lujin1.xls',
lujin,sheet,str)                                %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

%%输出最优解的路线和距离
disp('最优解:')
p = OutputPath(Chrom(minInd(1),:));
disp(['xxxxx',num2str(ObjV(minInd(1)))]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
end


  end

  function [gaocha]=GC(luu,data)
n=length(luu);
```

```
gaocha=0;
for i=1:n-1
    if data(luu(i+1),4)>data(luu(i),4)
        gaocha=gaocha+data(luu(i+1),4)-data(luu(i),4);
    end

end
 end

 function Chrom = InitPop(NIND,N)%初始化种群，
%输入：
%NIND：种群大小
%N：个体染色体长
%输出：
%初始种群
Chrom = zeros(NIND,N);  %用于存储种群

for i = 1:NIND

    Chrom(i,:) = randperm(N);%随机生成初始种群,randperm 函数的用法是返回一行 1~N 的整数，这 N 个数是不同的

end
 end

 function [a,b] = intercross(a,b)
%输入：
%a 和 b 为两个待交叉的个体
%输出：
%a 和 b 为交叉后得到的两个个体
L = length(a);
r1 = randsrc(1,1,[1:L]);
r2 = randsrc(1,1,[1:L]);
if r1~=r2
    a0 = a;
    b0 = b;
    s = min([r1,r2]);
    e = max([r1,r2]);
    for i =s:e
        a1 = a;
        b1 = b;
        a(i) = b0(i);
        b(i) = a0(i);
        x = find(a==a(i));
        y = find(b==b(i));
        i1 = x(x~=i)
        i2 = y(y~=i);
        if ~isempty(i1)
```

```
                a(i1)=a1(i);
            end
            if ~isempty(i2)
                b(i1)=b1(i);
            end
        end
end
  end

  function [luu]=LUJIN(lu1,lu2,lu3,ts)
jj1=length(lu1);
jj2=length(lu2);
jj3=length(lu3);
kk=1;
        for j=1:jj1
            if isnan(lu1(ts,j))== 0
                luu(1,kk)=lu1(ts,j);
                kk=kk+1;
            end
        end
        for j=1:jj2
            if isnan(lu2(ts,j))== 0
                luu(1,kk)=lu2(ts,j);
                kk=kk+1;
            end
        end
          for j=1:jj3
            if isnan(lu3(ts,j))== 0
                luu(1,kk)=lu3(ts,j);
                kk=kk+1;
            end
          end
  end

function SelCh = Mutate(SelCh,Pm)
%变异操作
%输入：
%SelCh    被选择的个体
%Pm    变异概率
%输出：
%SelCh    变异后的个体

[NSel,L] = size(SelCh);
for i = 1:NSel
        if Pm >= rand
            R = randperm(L);
            SelCh(i,R(1:2)) = SelCh(i,R(2:-1:1));
        end
end
```

```
end

function p=OutputPath(R)
%打印路线函数
%以 1->2->3 的形式在命令行打印路线

R = [R];
N = length(R);
p = num2str(R(1));
for i = 2:N
    p = [p,'->',num2str(R(i))];
end
disp(p)
end

function len = PathLength(D,Chrom)
%计算所有个体的路线长度
%输入
%D
%Chrom 个体的轨迹

[~,col] = size(D); %返回 D 的列数
NIND = size(Chrom,1);%NIND 等于初始种群
len = zeros(NIND,1);%初始化一个大小等于 NOND 的 len 来记录长度
for i = 1:NIND
    p = [Chrom(i,:) Chrom(i,1)];%构造 p 矩阵保存路线图 将第一行路线提出 再
加上第一个构成回路
    i1 = p(1:end-1);%i1 从第一个开始遍历到倒数第二个
    i2 = p(2:end);%i2 从第二个开始遍历到倒数第一个
    len(i,1) = sum(D((i1-1)*col+i2));%计算出每种路线（初始种群的个体）的长度
end
end

function SelCh = Recombin(SelCh,Pc)
%交叉操作
%输入：
%SelCh  被选择的个体
%Pc     交叉概率
%输出：
%SelCh  交叉后的个体

NSel = size(SelCh,1);
for i = 1:2:NSel - mod(NSel,2)
    if Pc>=rand %交叉概率 PC
        [SelCh(i,:),SelCh(i+1,:)] = intercross(SelCh(i,:),SelCh(i+1,:));
    end
end
end
```

```
function Chrom = Reins(Chrom,SelCh,ObjV)
%%重插入子代的种群
%输入：
%Chrom          父代的种群
%SelCh          子代的种群
%ObjV           父代适应度
%输出：
%Chrom          组合父代与子代后得到的新种群
NIND = size(Chrom,1);
NSel = size(SelCh,1);
[TobjV,index] = sort(ObjV);
Chrom =    [Chrom(index(1:NIND-NSel),:);SelCh];
end

function SelCh = Reverse(SelCh,D)
%进化逆转函数
%输入：
%SelCh    被选择的个体
%D
%输出：
%SelCh    进化逆转后的个体

[row,col] = size(SelCh);
ObjV = PathLength(D,SelCh);
SelCh1 = SelCh;
for i = 1:row
      r1 = randsrc(1,1,[1:col]);
      r2 = randsrc(1,1,[1:col]);
      mininverse = min([r1 r2]);
      maxinverse = max([r1 r2]);
      SelCh1(i,mininverse:maxinverse) = SelCh1(i,maxinverse:-1:mininverse);
end
ObjV1 = PathLength(D,SelCh1);%计算路线长度
index = ObjV1<ObjV;
SelCh(index,:)=SelCh1(index,:);
end

function SelCh = Select(Chrom,FitnV,GGAP)
%输入：
%Chrom  种群
%FitnV  适应度值
%GGAP  选择概率
%输出：
%SelCh  被选择的个体
NIND = size(Chrom,1);%种群个体总数
NSel = max(floor(NIND * GGAP+.5),2);%确定下一代种群的个体数，如果不足二
```

```
个就计为二个
ChrIx = Sus(FitnV,NSel);
SelCh = Chrom(ChrIx,:);
end

function NewChrIx = Sus(FitnV,Nsel)
%输入：
%FitnV 个体的是适应度值
%Nsel 被选个体的数目
%输出：
%NewChrIx 被选择个体的索引号
[Nind,ans] = size(FitnV);%Nind 为 FitnV 的行数也就是个体数 ans 为列数 1
cumfit = cumsum(FitnV);%对适应度累加 例如 1 2 3 累加后 1 3 6 用来计算累积概率
trials = cumfit(Nind)/Nsel * (rand + (0:Nsel-1)');%cumfit(Nind)表示的是矩阵 cumfit 的第 Nind 个元素 A.'是一般转置，A'是共轭转置 rand 返回一个在区间 (0,1) 内均匀分布的随机数
%cumfit(Nind)/Nsel 平均适应度
Mf = cumfit(:,ones(1,Nsel));%将生成的累积概率 复制 90 份 生成一个 100*90 的矩阵
Mt = trials(:,ones(1,Nind))';
[NewChrIx,ans] = find(Mt<Mf & [zeros(1,Nsel);Mf(1:Nind-1,:)]<= Mt);%寻找非零元素
[ans,shuf] = sort(rand(Nsel,1));%生成 Nsel*1 的随机数矩阵 按升序对 A 的元素进行排序 返回选择的 shuf
NewChrIx = NewChrIx(shuf);%将 shuf 返回
end

function [dis]=ZDIS(luu,data)
n=length(luu);
dis=0;
for i=1:n-1
    x1=data(luu(i),2); y1=data(luu(i),3); z1=data(luu(i),4);
    x2=data(luu(i+1),2);y2=data(luu(i+1),3);z2=data(luu(i+1),4);
    dis=dis+DIS(x1,x2,y1,y2,z1,z2);

end
end

function [znan]=ZN(luu,data)
n=length(luu);
znan=0;
for i=1:n

    znan=znan+data(luu(i),5);

end
end
```

```
lu1=xlsread('D:\桌面\省建模\解答 1.xls','sheet1')
lu2=xlsread('D:\桌面\省建模\解答 1.xls','sheet2')
lu3=xlsread('D:\桌面\省建模\解答 1.xls','sheet3')

delete '第二问解答 1'.xls
k=1;

%lu1 筛选
k=1;
lu=lu1;
for i=1:length(lu)

    if lu(i,1) ==1
        n=size(lu);
      luu=zeros(1,n(1,2)-1);
      for j=1:n(1,2)-1
        luu(1,j)=lu(i,j+1);
      end
      y00=isempty(find(luu(1,:)==1));
        if y00==1
            luu=[1 luu];
            str=['A'];
            str=[str num2str(k)];
            xlswrite('第二问解答 1.xls', luu,'sheet1',str)
            k=k+1;
        end
    end
end

%lu2 筛选
k=1;
lu=lu2;
for i=1:length(lu)

    if lu(i,1) ==20
        n=size(lu);
      luu=zeros(1,n(1,2)-1);
      for j=1:n(1,2)-1
        luu(1,j)=lu(i,j+1);
      end
      y00=isempty(find(luu(1,:)==20));
        if y00==1
            luu=[20 luu];
            str=['A'];
            str=[str num2str(k)];
            xlswrite('第二问解答 1.xls', luu,'sheet2',str)
            k=k+1;
        end
```

```
    end
end


%lu3 筛选
k=1;
lu=lu3;
for i=1:length(lu)

    if lu(i,1) ==26
        n=size(lu);
      luu=zeros(1,n(1,2)-1);
      for j=1:n(1,2)-1
        luu(1,j)=lu(i,j+1);
      end
      y00=isempty(find(luu(1,:)==26));
        if y00==1
            luu=[26 luu];
            str=['A'];
            str=[str num2str(k)];
            xlswrite('第二问解答 1.xls', luu,'sheet3',str)
            k=k+1;
        end
    end
end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')

delete '第二问最终解答.xls'

% %%%画基底图
% n=length(data);
% x=[];y=[];z=[];
% label={};
% for i=1:n
%         x(i,1)=data(i,2);
%         y(i,1)=data(i,3);
%         z(i,1)=data(i,4);
%         label(1,i)=raw(i+1,1);
% end
% iz=20; jz=26;
%   figure(1)
%   line(x,y,z);
% plot3(x,y,z,'r.','MarkerSize',20);
% hold on
% line(x(iz),y(iz),z(iz));
% plot3(x(iz),y(iz),z(iz),'b.','MarkerSize',20);          %第一个比打卡点
% hold on
% line(x(jz),y(jz),z(jz));
```

```
% plot3(x(jz),y(jz),z(jz),'b.','MarkerSize',20);          %第二个比打卡点
% text(x+0.02,y+0.02,z+0.02,label);
% hold on



% lu1=xlsread('D:\桌面\省建模\第二问解答 2.xls','sheet1')
% lu2=xlsread('D:\桌面\省建模\第二问解答 2.xls','sheet2')
% lu3=xlsread('D:\桌面\省建模\第二问解答 2.xls','sheet3')
lu=xlsread('D:\桌面\省建模\代码\第二问\第二问优秀结果.xlsx','sheet1')
n=size(lu);
% n=size(lu1);
%
% lu=[lu1(:,1:n(1,2)-1) lu2(:,1:n(1,2)-1) lu3];
for i=1:n(1,1)
    diss=ZDIS(lu(i,:),data);
    nann=ZN(lu(i,:),data);
    gaocha=GC(lu(i,:),data);
    luu(i,:)=[lu(i,:) diss gaocha nann];
end


xlswrite('第二问最终解答.xls', lu,'sheet1')
xlswrite('第二问优秀结果长宽高.xls', luu,'sheet1')
% %%画三合一图
% for i=1:n(1,1)
%     for j=1:length(lu)
%     xxx(j,1)=data(lu(i,j),2);
%     yyy(j,1)=data(lu(i,j),3);
%     zzz(j,1)=data(lu(i,j),4);
%     end
%
%     plot3(xxx,yyy,zzz);
%     hold on
% end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')
n=length(data);
x=[];y=[];z=[];
label={};
for i=1:n
    x(i,1)=data(i,2);
    y(i,1)=data(i,3);
    z(i,1)=data(i,4);
    label(1,i)=raw(i+1,1);
end

lu=xlsread('D:\桌面\省建模\代码\第二问\第二问最终解答.xls')
n=24;
```

```
nn=size(lu);



iz=20; jz=26;
for i=1:nn(1,1)
       %figure(i)
       for j=1:n
       xxx(j,1)=data(lu(i,j),2);
       yyy(j,1)=data(lu(i,j),3);
       zzz(j,1)=data(lu(i,j),4);
       end
     % line(x,y,z);
plot3(x,y,z,'r.','MarkerSize',20);
hold on
%line(x(iz),y(iz),z(iz));
plot3(x(iz),y(iz),z(iz),'b.','MarkerSize',20);          %第一个比打卡点
hold on
%line(x(jz),y(jz),z(jz));
plot3(x(jz),y(jz),z(jz),'b.','MarkerSize',20);          %第二个比打卡点
text(x+0.02,y+0.02,z+0.02,label);
       plot3(xxx,yyy,zzz);
       hold on
end
```

问题三代码：

```
clc;
clear all;

a=1000;
b=0;
c=0;
delete 'D:\桌面\省建模\解答 1.xls'
%while a>1000 || b>50 || c<12000
  for j=1:2
delete 'lujin1'.xls
close all



[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 3.xlsx')
xyz1=xlsread('D:\桌面\省建模\代码\第三问\xyz1.xls')


  mtspof_galll(xyz1,'sheet1');
```

```
xyz2=xlsread('D:\桌面\省建模\代码\第三问\xyz2.xls')
 mtspof_galll(xyz2,'sheet2');



xyz3=xlsread('D:\桌面\省建模\代码\第三问\xyz3.xls')
 mtspof_galll(xyz3,'sheet3');



 for i=1:3
       str1=['A'];
       str=['sheet'];
       str=[str num2str(i)];

       aaa=xlsread('D:\桌面\省建模\代码\第三问\lujin1.xls',str)
       k=size(aaa);
       k1=1+(j-1)*k(1,1);
       str1=[str1 num2str(k1)];
       xlswrite('D:\桌面\省建模\解答 1.xls', aaa,str,str1);
 end
 end


 function varargout=mtspof_galll(xyz2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 3.xlsx')

xyz=xyz2;


n=length(xyz);
for i=1:n-2
      xyy(i,:)=xyz(i+1,:)            %掐头去尾
end
xyt(1,1)=xyz(1,4);


   nn=6;                    %基站点数

   t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

   for j=1:length(t)
         xyt(1,j+1)=xyy(t(j),4);
   end

xyt(1,length(t)+2)=xyz(n,4);

for i=1:length(xyt)
```

```
        xytt(i,1)=data(xyt(1,i),2);            %(X,Y)形式
        xytt(i,2)=data(xyt(1,i),3);
        xytt(i,3)=data(xyt(1,i),4);
        xytt(i,4)=xyt(i);
end
GA(xytt,sheet);




  end

  function varargout=mtspof_galll(xyz2,sheet)
[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 2.xlsx')
%xy2=xlsread('D:\桌面\省建模\代码\xy2.xls')
xyz=xyz2;


n=length(xyz);
for i=1:n-2
        xyy(i,:)=xyz(i+1,:)            %掐头去尾
end
xyt(1,1)=xyz(1,4);


    nn=7;                      %基站点数

    t=randperm(n-2,nn);%从 1-n 个元素中随即取出 m 个元素，m 的值由你指定

    for j=1:length(t)
          xyt(1,j+1)=xyy(t(j),4);
    end

xyt(1,length(t)+2)=xyz(n,4);

for i=1:length(xyt)
        xytt(i,1)=data(xyt(1,i),2);            %(X,Y)形式
        xytt(i,2)=data(xyt(1,i),3);
        xytt(i,3)=data(xyt(1,i),4);
        xytt(i,4)=xyt(i);
end
GA(xytt,sheet);




  end

  function []=GA(xyz2,sheet)
```

```
for ss=1:10
xyz=xyz2;
n=length(xyz);
for i=1:n
      xx(i,1)=xyz(i,1);
      xx(i,2)=xyz(i,2);
      xx(i,3)=xyz(i,3);
      x(i,1)=xyz(i,1);
      y(i,1)=xyz(i,2);
      z(i,1)=xyz(i,3);
end

NIND = 100;              %种群大小
MAXGEN = 10;             %最大迭代次数
Pc = 0.9;                %交叉概率，相当于基因遗传的时候染色体交叉
Pm = 0.05;               %染色体变异
GGAP = 0.9;              %这个是代沟，通过遗传方式得到的子代数为父代数
*GGAP
D = Distance(xx);      %通过这个函数可以计算 i,j 两点之间的距离
N = size(D,1);         %计算有多少个坐标点
%%初始化种群
Chrom = InitPop(NIND,N);      %Chrome 代表的种群



%%画出随机解得路线图
%DrawPath(Chrom(1,:),xx)
%pause(0.0001)
%输出随机解的路线和总距离
  disp('初始种群中的一个随机值')
OutputPath(Chrom(1,:));%其中一个个体
Rlength = PathLength(D,Chrom(1,:));
disp(['总距离;',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%优化
gen = 0;
figure;
hold on;
box on;
xlim([0,MAXGEN])
title('优化过程')
xlabel('迭代次数')
ylabel('当前最优解')
ObjV = PathLength(D,Chrom);%计算当前路线长度，即上面随机产生的那些个体
路径
preObjV = min(ObjV);%当前最优解
```

```
while gen<MAXGEN
    %%计算适应度
    ObjV = PathLength(D,Chrom);         %计算路线长度
    line([gen - 1,gen],[preObjV,min(ObjV)]);
    pause(0.0001);
    preObjV = min(ObjV);
    FitnV = Fitness(ObjV);
    %选择
    SelCh = Select(Chrom,FitnV,GGAP);
    %交叉操作
    SelCH = Recombin(SelCh,Pc);
    %变异
    SelCh = Mutate(SelCh,Pm);
    %逆转操作
    SelCh = Reverse(SelCh,D);
    %重插入子代的新种群
    Chrom = Reins(Chrom,SelCh,ObjV);
    %更新迭代次数
    gen = gen + 1;
end

for i=1:NIND
th1=Chrom(i,1);
th2=Chrom(i,n);
label1=find(Chrom(i,:)==1);
label2=find(Chrom(i,:)==n);

Chrom(i,1)=1; Chrom(i,label1)=th1;
Chrom(i,n)=n; Chrom(i,label2)=th2;
end




%画出最优解的路线图
ObjV = PathLength(D,Chrom);         %计算路线长度
[minObjV,minInd] = min(ObjV);
%DrawPath(Chrom(minInd(1),:),xx)
%hold on



for i=1:n
lujin(1,i)=xyz(Chrom(minInd(1),i),4);
end
str=['A'];
str=[str num2str(ss)];
%str=['A1','A2','A3','A4','A5','A6','A7','A8','A9','A10','A11A12A13A14A15A16A17A
18A19A20A21A22A23A24A25A26A27A28A29A30'];
```

```
%str=['A1';'A2';'A3';'A4';'A5';'A6';'A7';'A8';'A9';'A10';'A11';'A12';'A13';'A14';'A15';'A
16';'A17';'A18';'A19';'A20';'A21';'A22';'A23';'A24';'A25';'A26';'A27';'A28';'A29';'A30'];

xlswrite('lujin1.xls',
lujin,sheet,str)                    %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%


%%输出最优解的路线和距离
disp('最优解:')
p = OutputPath(Chrom(minInd(1),:));
disp(['xxxxx',num2str(ObjV(minInd(1)))]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
end


  end

  function [D]=DIS(x1,x2,y1,y2,z1,z2)
D=((x1-x2)^2+(y1-y2)^2+(z1-z2)^2)^0.5;
  end

  function D = Distance(a)
%计
%输入的数据
row = size(a,1);
D = zeros(row,row);
for i = 1:row
        for j = i+1:row
                D(i,j) = ((a(i,1) - a(j,1))^2 + (a(i,2)-a(j,2))^2+(a(i,3)-a(j,3))^2)^0.5;
                D(j,i) = D(i,j);
        end
end
  end

  function DrawPath(Chrom,X)
%输入：
%待画路线
%坐标位置
%输出：
%
n=length(Chrom);
R =    [Chrom(1,:) Chrom(1,n)]; %第一个随机解（个体面又起点”
figure;
hold on
plot3(X(:,1),X(:,2),X(:,3),'bo')%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐标
%plot(X(:,1),X(:,2),'o','color',[1,1,1])%X(:,1)，X(:,2)分别代表的 X 轴坐标和 Y 轴坐
标，
```

```
plot3(X(Chrom(1,1),1),X(Chrom(1,1),2),X(Chrom(1,1),3),'rv','MarkerSize',20)%标记起点
A = X(R,:);                                    %A 是将之前的坐标顺序用 R 打乱后重新存入 A 中
row = size(A,1);                               %row 为坐标数+1
for i = 2:row
    [arrowx,arrowy,arrowz] = dsxy2figxy(gca,A(i-1:i,1),A(i-1:i,2),A(i-1:i,3));    %dsxy2figxy 坐标转换函数，记录两个点
    annotation('textarrow',arrowx,arrowy,arrowz,'HeadWidth',8,'color',[0,0,1]);%  将这两个点连接起来
end
hold off
xlabel('横坐标 x')
ylabel('纵坐标 y')
zlabel('纵坐标 z')
title('图')
box on
 end

 function varargout = dsxy2figxy(varargin) %%坐标转换函数
if length(varargin{1}) == 1 && ishandle(varargin{1}) ...
                                    && strcmp(get(varargin{1},'type'),'axes')
    hAx = varargin{1};
    varargin = varargin(2:end);
else
    hAx = gca;
end
if length(varargin) == 1
    pos = varargin{1};
else
    [x,y,z] = deal(varargin{:});
end
axun = get(hAx,'Units');
set(hAx,'Units','normalized');
axpos = get(hAx,'Position');
axlim = axis(hAx);
axwidth = diff(axlim(1:2));
axheight = diff(axlim(3:4));
if exist('x','var')
    varargout{1} = (x - axlim(1)) * axpos(3) / axwidth + axpos(1);
    varargout{2} = (y - axlim(3)) * axpos(4) / axheight + axpos(2);
else
    pos(1) = (pos(1) - axlim(1)) / axwidth * axpos(3) + axpos(1);
    pos(2) = (pos(2) - axlim(3)) / axheight * axpos(4) + axpos(2);
    pos(3) = pos(3) * axpos(3) / axwidth;
    pos(4) = pos(4) * axpos(4 )/ axheight;
    varargout{1} = pos;
end
```

```
set(hAx,'Units',axun)
 end

 function FitnV = Fitness(len) %适应度函数
%输入：
%len 个体的长度（TSP 的距离）
%输出：
%FitnV 个体的适应度值
FitnV = 1./len;
 end

 function [D,path,min1,path1]=floyd(a,start,terminal)
D=a;n=size(D,1);path=zeros(n,n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;
        end,
    end,
end
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end,
        end,
    end,
end
if nargin==3
    min1=D(start,terminal);
    m(1)=start;
    i=1;
    path1=[ ];
    while     path(m(i),terminal)~=terminal
        k=i+1;
        m(k)=path(m(i),terminal);
        i=i+1;
    end
    m(i+1)=terminal;
    path1=m;
end
 end

function [gaocha]=GC(luu,data)
n=length(luu);
gaocha=0;
for i=1:n-1
```

```
    if data(luu(i+1),4)>data(luu(i),4)
        gaocha=gaocha+data(luu(i+1),4)-data(luu(i),4);
    end

end
end

function Chrom = InitPop(NIND,N)%初始化种群，
%输入：
%NIND：种群大小
%N：个体染色
%输出：
%初始种群
Chrom = zeros(NIND,N);   %用于存储种群

for i = 1:NIND

    Chrom(i,:) = randperm(N);%随机生成初始种群,randperm 函数的用法是返回一行 1~N 的整数，这 N 个数是不同的

end
end

function [a,b] = intercross(a,b)
%输入：
%a 和 b 为两个待交叉的个体
%输出：
%a 和 b 为交叉后得到的两个个体
L = length(a);
r1 = randsrc(1,1,[1:L]);
r2 = randsrc(1,1,[1:L]);
if r1~=r2
    a0 = a;
    b0 = b;
    s = min([r1,r2]);
    e = max([r1,r2]);
    for i =s:e
        a1 = a;
        b1 = b;
        a(i) = b0(i);
        b(i) = a0(i);
        x = find(a==a(i));
        y = find(b==b(i));
        i1 = x(x~=i)
        i2 = y(y~=i);
        if ~isempty(i1)
            a(i1)=a1(i);
        end
```

```
            if ~isempty(i2)
                b(i1)=b1(i);
            end
        end
    end
    end

    function [jiange]=JG(jli,pai,sudu,i)
        lc=jli(i,1);
        n=size(pai);
        for j=1:n(1,1)

            ti(j,1)=lc/(sudu(pai(j,i),2)*100);
        end
        nn=size(ti);
        for i=1:nn(1,1)-1
            jg(i,1)=ti(i)-ti(i+1);
        end
        jiange=ceil(max(jg));
    end

    function [luu]=LUJIN(lu1,lu2,lu3,ts)
    jj1=length(lu1);
    jj2=length(lu2);
    jj3=length(lu3);
    kk=1;
        for j=1:jj1
            if isnan(lu1(ts,j))== 0
                luu(1,kk)=lu1(ts,j);
                kk=kk+1;
            end
        end
        for j=1:jj2
            if isnan(lu2(ts,j))== 0
                luu(1,kk)=lu2(ts,j);
                kk=kk+1;
            end
        end
          for j=1:jj3
            if isnan(lu3(ts,j))== 0
                luu(1,kk)=lu3(ts,j);
                kk=kk+1;
            end
          end
    end

    function SelCh = Mutate(SelCh,Pm)
    %变异操作
    %输入：
    %SelCh    被选择的个体
```

```
%Pm    变异概率
%输出：
%SelCh    变异后的个体

[NSel,L] = size(SelCh);
for i = 1:NSel
    if Pm >= rand
        R = randperm(L);
        SelCh(i,R(1:2)) = SelCh(i,R(2:-1:1));
    end
end
end

function p=OutputPath(R)
%打印路线函数
%以 1->2->3 的形式在命令行打印路线

R = [R];
N = length(R);
p = num2str(R(1));
for i = 2:N
    p = [p,'->',num2str(R(i))];
end
disp(p)
end

function len = PathLength(D,Chrom)
%计算所有个体的路线长度
%输入
%D 两
%Chrom 个体的轨迹

[~,col] = size(D); %返回 D 的列数
NIND = size(Chrom,1);%NIND 等于初始种群
len = zeros(NIND,1);%初始化一个大小等于 NOND 的 len 来记录长度
for i = 1:NIND
    p = [Chrom(i,:) Chrom(i,1)];%构造 p 矩阵保存路线图 将第一行路线提出 再
加上第一个构成回路
    i1 = p(1:end-1);%i1 从第一个开始遍历到倒数第二个
    i2 = p(2:end);%i2 从第二个开始遍历到倒数第一个
    len(i,1) = sum(D((i1-1)*col+i2));%计算出每种路线（初始种群的个体）的长度
end
end

function SelCh = Recombin(SelCh,Pc)
%交叉操作
%输入：
%SelCh  被选择的个体
```

```
%Pc       交叉概率
%输出：
%SelCh  交叉后的个体

NSel = size(SelCh,1);
for i = 1:2:NSel - mod(NSel,2)
      if Pc>=rand %交叉概率 PC
            [SelCh(i,:),SelCh(i+1,:)] = intercross(SelCh(i,:),SelCh(i+1,:));
      end
end
end

function Chrom = Reins(Chrom,SelCh,ObjV)
%%重插入子代的种群
%输入：
%Chrom          父代的种群
%SelCh          子代的种群
%ObjV           父代适应度
%输出：
%Chrom          组合父代与子代后得到的新种群
NIND = size(Chrom,1);
NSel = size(SelCh,1);
[TobjV,index] = sort(ObjV);
Chrom =    [Chrom(index(1:NIND-NSel),:);SelCh];
end

function SelCh = Reverse(SelCh,D)
%进化逆转函数
%输入：
%SelCh    被选择的个体
%D    各城市的距离矩阵
%输出：
%SelCh    进化逆转后的个体

[row,col] = size(SelCh);
ObjV = PathLength(D,SelCh);
SelCh1 = SelCh;
for i = 1:row
      r1 = randsrc(1,1,[1:col]);
      r2 = randsrc(1,1,[1:col]);
      mininverse = min([r1 r2]);
      maxinverse = max([r1 r2]);
      SelCh1(i,mininverse:maxinverse) = SelCh1(i,maxinverse:-1:mininverse);
end
ObjV1 = PathLength(D,SelCh1);%计算路线长度
index = ObjV1<ObjV;
SelCh(index,:)=SelCh1(index,:);
end
```

```
function SelCh = Select(Chrom,FitnV,GGAP)
%输入：
%Chrom 种群
%FitnV 适应度值
%GGAP 选择概率
%输出：
%SelCh 被选择的个体
NIND = size(Chrom,1);%种群个体总数
NSel = max(floor(NIND * GGAP+.5),2);%确定下一代种群的个体数，如果不足二个就计为二个
ChrIx = Sus(FitnV,NSel);
SelCh = Chrom(ChrIx,:);
end

function NewChrIx = Sus(FitnV,Nsel)
%输入：
%FitnV 个体的是适应度值
%Nsel 被选个体的数目
%输出：
%NewChrIx 被选择个体的索引号
[Nind,ans] = size(FitnV);%Nind 为 FitnV 的行数也就是个体数 ans 为列数 1
cumfit = cumsum(FitnV);%对适应度累加 例如 1 2 3 累加后 1 3 6 用来计算累积概率
trials = cumfit(Nind)/Nsel * (rand + (0:Nsel-1)');%cumfit(Nind)表示的是矩阵 cumfit 的第 Nind 个元素 A.'是一般转置，A'是共轭转置 rand 返回一个在区间 (0,1) 内均匀分布的随机数
%cumfit(Nind)/Nsel 平均适应度
Mf = cumfit(:,ones(1,Nsel));%将生成的累积概率 复制 90 份 生成一个 100*90 的矩阵
Mt = trials(:,ones(1,Nind))';
[NewChrIx,ans] = find(Mt<Mf & [zeros(1,Nsel);Mf(1:Nind-1,:)]<= Mt);%寻找非零元素
[ans,shuf] = sort(rand(Nsel,1));%生成 Nsel*1 的随机数矩阵 按升序对 A 的元素进行排序 返回选择的 shuf
NewChrIx = NewChrIx(shuf);%将 shuf 返回
end

function [dis]=ZDIS(luu,data)
n=length(luu);
dis=0;
for i=1:n-1
    x1=data(luu(i),2); y1=data(luu(i),3); z1=data(luu(i),4);
    x2=data(luu(i+1),2);y2=data(luu(i+1),3);z2=data(luu(i+1),4);
    dis=dis+DIS(x1,x2,y1,y2,z1,z2);
```

```
end
end

function [znan]=ZN(luu,data)
n=length(luu);
znan=0;
for i=1:n

    znan=znan+data(luu(i),5);

end
end

%人工筛查
lu1=xlsread('D:\桌面\省建模\解答 1.xls','sheet1')
lu2=xlsread('D:\桌面\省建模\解答 1.xls','sheet2')
lu3=xlsread('D:\桌面\省建模\解答 1.xls','sheet3')

delete '第三问解答 1'.xls
k=1;

%lu1 筛选
k=1;
lu=lu1;
for i=1:length(lu)

    if lu(i,1) ==1
        n=size(lu);
      luu=zeros(1,n(1,2)-1);
      for j=1:n(1,2)-1
        luu(1,j)=lu(i,j+1);
      end
      y00=isempty(find(luu(1,:)==1));
        if y00==1
            luu=[1 luu];
            str=['A'];
            str=[str num2str(k)];
            xlswrite('第三问解答 1.xls', luu,'sheet1',str)
            k=k+1;
        end
    end
end

%lu2 筛选
k=1;
lu=lu2;
for i=1:length(lu)

    if lu(i,1) ==137
```

```
            n=size(lu);
        luu=zeros(1,n(1,2)-1);
        for j=1:n(1,2)-1
            luu(1,j)=lu(i,j+1);
        end
        y00=isempty(find(luu(1,:)==137));
            if y00==1
                luu=[137 luu];
                str=['A'];
                str=[str num2str(k)];
                xlswrite('第三问解答 1.xls', luu,'sheet2',str)
                k=k+1;
            end
      end
end


%lu3 筛选
k=1;
lu=lu3;
for i=1:length(lu)

      if lu(i,1) ==35
            n=size(lu);
        luu=zeros(1,n(1,2)-1);
        for j=1:n(1,2)-1
            luu(1,j)=lu(i,j+1);
        end
        y00=isempty(find(luu(1,:)==35));
            if y00==1
                luu=[35 luu];
                str=['A'];
                str=[str num2str(k)];
                xlswrite('第三问解答 1.xls', luu,'sheet3',str)
                k=k+1;
            end
      end
end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 3.xlsx')

delete '第三问最终解答.xls'

% %%%画基底图
% n=length(data);
% x=[];y=[];z=[];
% label={};
% for i=1:n
%         x(i,1)=data(i,2);
```

```
%         y(i,1)=data(i,3);
%         z(i,1)=data(i,4);
%         label(1,i)=raw(i+1,1);
% end
% iz=20; jz=26;
%    figure(1)
%    line(x,y,z);
% plot3(x,y,z,'r.','MarkerSize',20);
% hold on
% line(x(iz),y(iz),z(iz));
% plot3(x(iz),y(iz),z(iz),'b.','MarkerSize',20);          %第一个比打卡点
% hold on
% line(x(jz),y(jz),z(jz));
% plot3(x(jz),y(jz),z(jz),'b.','MarkerSize',20);          %第二个比打卡点
% text(x+0.02,y+0.02,z+0.02,label);
% hold on



lu1=xlsread('D:\桌面\省建模\第三问解答 2.xls','sheet1')
lu2=xlsread('D:\桌面\省建模\第三问解答 2.xls','sheet2')
lu3=xlsread('D:\桌面\省建模\第三问解答 2.xls','sheet3')
n=size(lu1);

lu=[lu1(:,1:n(1,2)-1) lu2(:,1:n(1,2)-1) lu3];
for i=1:n(1,1)
       diss=ZDIS(lu(i,:),data);
       nann=ZN(lu(i,:),data);
       gaocha=GC(lu(i,:),data);
       luu(i,:)=[lu(i,:) diss gaocha nann];
end


xlswrite('第三问最终解答.xls', lu,'sheet1')

% %%画三合一图
% for i=1:n(1,1)
%         for j=1:length(lu)
%         xxx(j,1)=data(lu(i,j),2);
%         yyy(j,1)=data(lu(i,j),3);
%         zzz(j,1)=data(lu(i,j),4);
%         end
%
%         plot3(xxx,yyy,zzz);
%         hold on
% end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 3.xlsx')
n=length(data);
```

```
x=[];y=[];z=[];
label={};
for i=1:n
    x(i,1)=data(i,2);
    y(i,1)=data(i,3);
    z(i,1)=data(i,4);
    label(1,i)=raw(i+1,1);
end

lu=xlsread('D:\桌面\省建模\代码\第三问\第三问最终解答.xls')
n=22;
nn=size(lu);



iz=137; jz=35;
for i=1:nn(1,1)
    %figure(i)
    for j=1:n
    xxx(j,1)=data(lu(i,j),2);
    yyy(j,1)=data(lu(i,j),3);
    zzz(j,1)=data(lu(i,j),4);
    end
  % line(x,y,z);
plot3(x,y,z,'r.','MarkerSize',20);
hold on
%line(x(iz),y(iz),z(iz));
plot3(x(iz),y(iz),z(iz),'^','MarkerFaceColor','b','MarkerSize',10);        %第一个比打卡点
hold on
%line(x(jz),y(jz),z(jz));
plot3(x(jz),y(jz),z(jz),'v','MarkerFaceColor','b','MarkerSize',10);        %第二个比打卡点
plot3(x(1),y(1),z(1),'p','MarkerFaceColor','g','MarkerSize',10);
plot3(x(36),y(36),z(36),'d','MarkerFaceColor','g','MarkerSize',10);
text(x+0.02,y+0.02,z+0.02,label);
    plot3(xxx,yyy,zzz);
    hold on
    legend('检查点','第一补给点','第二补给点','起点','终点');
end

[data,txt,raw]=xlsread('D:\桌面\省建模\题目\B 题\附件 3.xlsx')

lu=xlsread('D:\桌面\省建模\最终终解答第三问.xls')
n=size(lu);


for i=1:n(1,1)
    diss=ZDIS(lu(i,:),data);
```

```
    nann=ZN(lu(i,:),data);
    gaocha=GC(lu(i,:),data);
    luu(i,:)=[lu(i,:) diss gaocha nann];
end

xlswrite('最终终解答第三问长宽高.xls', luu,'sheet1')

[sudu2,txt,raw]=xlsread('D:\桌面\省建模\代码\第三问\速度.xls')
[sudu,txt,raw]=xlsread('D:\桌面\省建模\代码\第三问\附件 3 速度.xlsx')
lu=xlsread('D:\桌面\省建模\代码\第三问\最终终解答第三问长宽高.xls')
jjj=size(lu);
for i=1:jjj(1,1)
    jli(i,1)=lu(i,23);
end
n=length(sudu);
zu=11;
ge=floor(n/zu);



for i=1:ge
    for j=1:zu
        pai(i,j)=sudu(j+(i-1)*zu+10,1);
    end
end


% pai1=[sudu(1,1);pai(:,1)];
% pai2=[sudu(2,1);pai(:,2)];
% pai3=[sudu(3,1);pai(:,3)];
% pai4=[sudu(4,1);pai(:,4)];
% pai5=[sudu(5,1);pai(:,5)];
% pai6=[sudu(6,1);pai(:,6)];
% pai7=[sudu(7,1);pai(:,7)];
% pai8=[sudu(8,1);pai(:,8)];
% pai9=[sudu(9,1);pai(:,9)];
% pai10=[sudu(10,1);pai(:,10)];
for i=1:zu
  jli1(i,:)=jli(i+39,:);
end
% jiange(1,1)=JG1(jli,pai1,sudu,1);
% jiange(2,1)=JG1(jli,pai2,sudu,2);
% jiange(3,1)=JG1(jli,pai3,sudu,3);
% jiange(4,1)=JG1(jli,pai4,sudu,4);
% jiange(5,1)=JG1(jli,pai5,sudu,5);
% jiange(6,1)=JG1(jli,pai6,sudu,6);
% jiange(7,1)=JG1(jli,pai7,sudu,7);
% jiange(8,1)=JG1(jli,pai8,sudu,8);
% jiange(9,1)=JG1(jli,pai9,sudu,9);
% jiange(10,1)=JG1(jli,pai10,sudu,10);
```

```
%
% jiange(11,1)=JG1(jli1,pai(:,11),sudu2,11);


for i=1:zu
    jiange(i,1)=JG(jli1,pai,sudu2,i);
end
function [jiange]=JG1(jli,pai,sudu,i)
    lc=jli(i,1);
    n=size(pai);

        for j=1:n(1,1)

          ti(j,1)=lc/(sudu(pai(j,1),2)*100);
    end
    nn=size(ti);
    for i=1:nn(1,1)-1
          jg(i,1)=ti(i)-ti(i+1);
    end
    jiange=ceil(max(jg));
end
```