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。）

定向越野比赛的路线设计研究

摘要

本文对定向越野比赛的路线设计问题进行了研究，针对不同情况的具体问题，分别建立了相应的多目标规划数学模型，利用遗传算法，给出了合理的比赛路线设计方案。

针对问题一，在不考虑比赛路线爬高量的条件下，本文考虑多个约束条件，建立多约束条件下各路线总长度差最小、总难度差最小和加入到竞赛中的参赛选手数最大的多目标优化模型，通过加权处理，将多目标规划模型转化为单目标规划模型。利用遗传算法对其进行求解，首先得出 10 条路线的设计方案见表 5.1，之后根据平均距离方差与平均难度方差两个指标，对模型进行优化，最终得出 5 条打卡点数量相当的路线设计方案，见表附件 4“第 1 问结果”，路线的终点是序号为 84 的检查点，必经检查点是序号为 25 和 98 的检查点，总参赛选手数量为 273 名，各条路线的总长度见表 5.8，各条路线的总难度见表 5.9,各条路线的参赛选手数量见表 5.10。

针对问题二，考虑三维空间中的路线设计问题，在问题一的基础上，增加不同路线爬高量之差最小这一目标函数，生成新的多目标规划模型。利用改进的遗传算法求解，得出 5 条路线的设计方案见表附件 4“第 2 问结果”，总参赛选手数量为 281 名，各条路线的总长度见表 5.11，各条路线的总难度见表 5.12，各条路线的爬高量见表 5.14,各条路线的参赛选手数量见表 5.15。

针对问题三 ，与问题二不同的是，参赛选手数量已知，且不同选手行进速度不同，在问题二的基础上，目标函数变为路线总数最少与完成时间最短，建立多目标规划模型，求解得出使用的路线最少为 2 条，路线设计方案见表附件 4“第 3 问结果”，比赛完成的时间最短为 185.9408 分钟，各路线总长度见表 5.16，各路线总难度见表 5.18，各条路线的参赛选手数量见表 5.19，各路线爬高量见表 5.20。

最后，通过对模型应用的结果分析，本文对所建立的模型优缺点进行了评价。

关键词：定向越野；路线设计；多目标规划；聚类；改进遗传算法

1. 问题重述

1.1 问题背景

定向越野对培养人的环境适应能力、创新意识和创新能力、竞争意识和挑战意识等方面有着重要作用，是一项集实用性、知识性、趣味性和锻炼性于一体的智慧型户外运动项目。随着全民健身运动体育政策的逐渐贯彻和落实，以及国民生活水平的提高，现代生活方式的改变，休闲类体育运动愈加受到广大民众的支持和追捧，定向越野作为休闲类体育中新成长的一项户外运动，以其独有的特质，为社会大众迅速了解和喜爱。[1]

路线设计作为定向越野核心内容，在定向越野专项技术训练中具有重要的指导作用。科学合理的路线设计，可以有效控制比赛的总完成时间，把握赛事进程安排，实现对参赛选手的竞赛相对公平性。因此，定向越野路线的设计与规划有着重要的现实意义。

1.2 问题提出

题目通过介绍定向越野比赛的路线常规设计方式及比赛规则的普遍要求，提供 7 条设计规则，给出 3 个数据文件，要求利用该数据解决如下三个问题：

问题一：在假定所有参赛选手的行进速度相等的条件下，不考虑爬高量，将三维空间中路线设计问题降维至二维平面中讨论。某次定向越野赛中，起点确定，已知 99 个候选检查点的地理位置坐标和难度系数，在选手完成比赛任务的平均时间不低于 2 小时，赛事总完成时间不超过 3 小时的条件下，要求加入比赛的选手数量尽可能多，给出设计方案（包括参赛选手总数量、各条路线参赛选手数量、检查点如何选择、路线总长度及总难度）。

问题二：在问题一的基础上，考虑爬高量，在三维空间中进行讨论，候选点增至 149 个，在每条路线的爬高量尽可能均匀的限制条件下，其他约束条件与问题一相同，给出此次竞赛的设计方案。

问题三：已知起点和 199 个候选检查点的相关数据，以及 120 名参赛选手的速度数据，在选手完成比赛任务的平均时间不低于 2 小时，赛事总完成时间不超过 3.5 小时的条件下，要求使用的路线尽可能少、完成时间尽可能短，给出设计方案。

2. 问题分析

本文研究定向越野比赛路线设计问题。自 20 世纪 80 年代定向运动传入中国以来，便以其健康性、挑战性深受人们喜爱，关于路线设计相关的文献也较多。大部分都是在在缩短总路线的基础上，考虑人员安全和参赛选手数量最多等一些限制条件；在路线设计的基础上，进一步减少路线使用数量。这类问题的解决一般有两类方法：一是传统算法，其得到的结果较精确，但随着模型复杂度和问题规模的增加，算法时间复杂度也更大，效率也随之降低；另一类是智能算法，如粒子群算法、模拟退火算法、遗传算法等，这些算法计算速度较快，适应性高，但容易陷入局部最优解。

针对上述背景下的多约束条件路线设计问题，本文从目标函数的角度入手，分析了各种约束条件下路线设计具体情形并建立了带约束的多目标优化模型。同时，根据问题的限制条件具体分析，考虑了多种约束条件，采用遗传算法，并根据实际情况和计算结果对算法进行改进，得到更优解。

2.1 问题一的分析

首先根据题目要求不同路线的起点终点相同，通过分析，确定终点位置。其次对区域进行划分，通过各区域中难度低且到其他打卡点距离和最小这一条件，确定两个必经的打卡点。然后根据确定的这四点，对剩余候选点进行划分与聚类。结合约束条件，建立多目标规划模型，通过加权处理，将多目标规划问题转化为单目标规划问题。最后利用遗传算法求解，得出合理的路线设计方案。

2.2 问题二的分析

在问题一的基础上，考虑不同打卡点之间的高程差，增加一个目标函数及相应的限制条件，结合新增约束条件，重新构造多目标规划函数，对问题一所建立的模型进行相关部分的修改，改进遗传算法，再进行求解，得出合理的设计方案。

2.3 问题三的分析

与前两个问题有所不同，问题三中已知参与竞赛的选手数量，且选手行进速度有所差别，总完成时间变为不超过 3.5 个小时，要求使用的路线尽可能少、完成的时间尽可能短，即多目标规划中目标与限制条件发生改变，重新建立多目标规划模型，再进行求解。

3. 模型假设

为简化问题，题目已对参赛选手行进速度、不同检查点之间路径进行了假设，除此之外，我们进一步作出如下假设和要求：

（1）不考虑打卡点附近地形和打卡具体要求，以所需花费时间来刻画打卡点难度。

（2）同一路线中相邻的打卡点间距离不能过大，以防长距离回头行进，导致路线过长。

（3）在设计路线时，除设置必经打卡点之外，不考虑参赛选手除识图能力、体能等其他意外情况的影响。

（4）问题一、二中选手行进速度均按照平均速度计算。

（5）设计路线时，不考虑天气等客观因素对选手的影响。

（6）所有路线除起点、终点、必经打卡点外没有其他公共打卡点。

4. 符号说明

符号	说明
p_i	第i个检查点
x_i, y_i, z_i	第i个检查点的三维坐标
L_i	第i条路线总长
d_{ij}	第i, j个检查点之间的距离
Q_i	第i条路线总打卡难度
q_i	第i个检查点的难度
H_i	第i条路线的总爬高量

h_{ii}	第i,j个检查点之间的高程差
$\bar{v}$	平均速度
v_i	第i个运动员的行进速度
n_i	第i条路线的参赛选手数量
N	总参赛选手数量
M	路线数
$\bar{t}$	平均完成时间
t_i	第i个参赛者完成时间
t_0	出发时间间隔
T	比赛总完成时间
D_i	划分区域中第i个区域
S_i	第i条路线打卡点数量
a_i	顺序编码后的打卡点编号
c	选手出发总批次

5. 模型的建立与求解

5.1 问题一模型建立与求解

5.1.1 模型准备

（1）数据预处理

本题中要计算欧氏距离，且涉及到一些距离计算的算法，为提升模型的收敛速度、提升模型的精度，让各个特征对结果做出的贡献相同，我们对所提供的数据进行了归一化处理[2]。对难度向量进行了 z-score 标准化，对距离矩阵进行了最小最大归一化处理。

（2）确定终点

观察附件 1 所给起点、候选点难度信息，我们发现这些打卡点的难度分为四个等级。如图 5.1 所示，其中红色点代表难度为 1 的检查点，蓝色点代表难度为 2 的检查点，绿色点表示难度为 3 的检查点，黑色点表示难度为 4 的检查点。

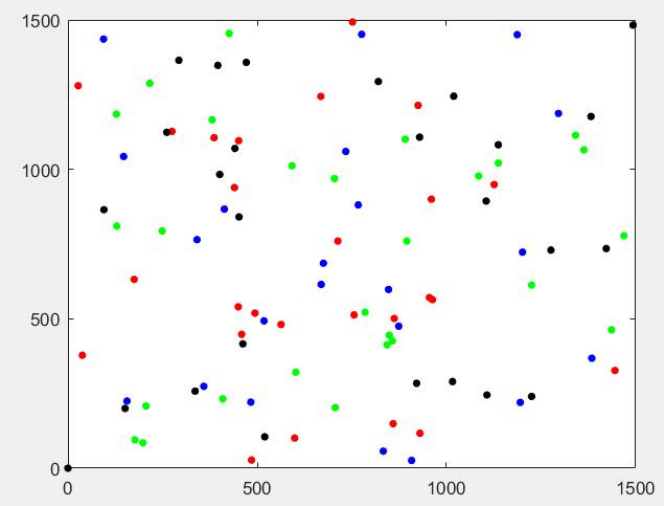

图 5.1 检查点分布情况图

由图可知，各等级检查点分布较为混乱，不成层状分布，不可直接进行选取。

由于题目要求不同路线长度及打卡难度尽量均匀，我们选取距离起点最远的点，即序号为 84 的检查点为终点，即点 p_{84}，如图 5.2 所示：

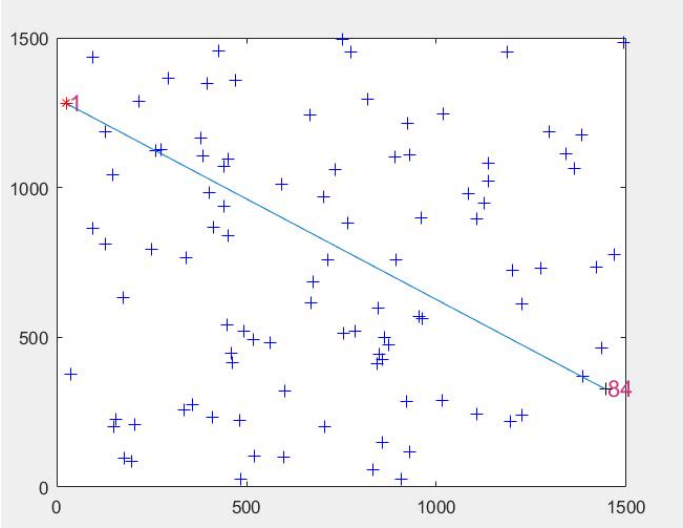

图 5.2 终点选取示意图

首先，其难度系数为 1，满足题目所给“终点设置在难度较低的检查点”这一要求。其次，距离出发点最远，这样所设计的路线大致方向可以为从 1 号检查点到 84 号检查点，在行进过程中，减少反向行进对参赛者造成的不利影响，以便对选手完成比赛时间进行预测以及更确切地把控整个竞赛持续时长。最后，选择最远点，在设计路线时，可进行区域划分、分层聚类，更好地满足题目所给“不同路线总长度尽可能均匀”的要求。

（3）选取必经打卡点

要求设置两个所有路线都必须经过的打卡点。根据题目说明及实际情况，因为必经打卡点需要及时给予选手水源补充，并且能够最大化程度做到医疗保障，对突发状况做到及时反映且不会发生同时出医造成的人员浪费，即要相对其他候选点距离都比较小，因此两个必经打卡点位置不能过近。我们按如下步骤选取必经打卡点：

Step 1:区域划分

通过上述分析我们知道两必经打卡点分布需要相对较远，需相对均匀地分布于竞赛场地。故考虑对候选点进行区域划分，在不同区域中进行必经打卡点的选取，如图 5.3 所示：

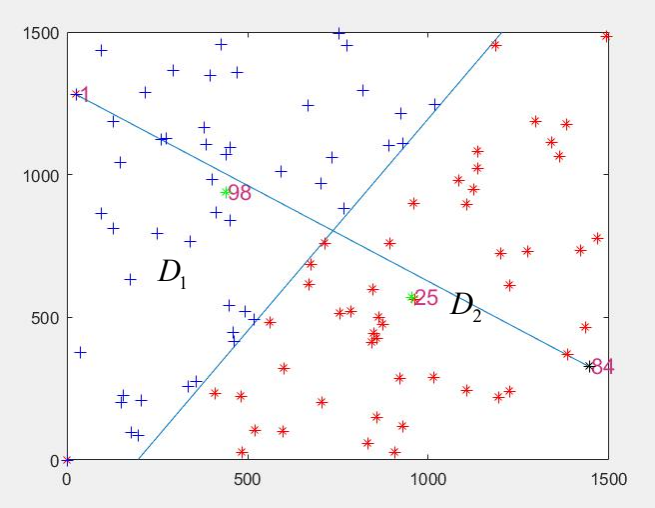

图 5.3 必经打卡点选取示意图

连接起点 p_1 与终点 p_{84}，做连线段 p_1p_{84} 的垂直平分线，将候选点分为 D_1, D_2 两个区域。

Step 2:选取难度较低打卡点

选取低难度打卡点，可以避免选手在此做不必要的停留，从而降低拥堵可能性，且尽可能满足题目所给约束条件（7）。故在每个区域中，首先确定难度最低的打卡点。

Step 3:确定距其他点最近的打卡点

然后计算步骤 2 中选取的难度最低的点到该区域其他打卡点的距离和，通过求解

$$\min \sum_{j \in D_1} d_{ij}, \text{其中} \forall i \in D_1, q_i = 1,$$

得到区域 D_1 中聚点为点 p_{98}，区域 D_2 中聚点为点 p_{25}，且由图可得，这两个聚点也大致分布于路线三分之一与三分之二处，满足必经打卡点的各种条件，故选取这两点作为必经打卡点的方案是可行的。

（4）确定总路线数及出发时间间隔

题目所给规则（7）要求同时到达同一个检查点的选手数量不超过 5 人，在选取难度最低的检查点为必经打卡点的前提下，尽可能降低滞留必经打卡点的堵塞性，即避免同时到达打卡点人员过多的情形。考虑路线上其他打卡点，在速度一定的情况下，设定合理出发间隔时间，可确定同一路线前后出发的选手不会同时到达同一打卡点，故只需考虑不同路线选手到达同一打卡点的情况。因此我们暂定路线共有 10 条，这样做首先避免了过多约束条件使模型求解陷入局部最优解，其次减少了路线过少，不同路线打卡点重复的可能性。

题目要求使得参赛选手尽可能多，因此我们按最大比赛完成时间$T=3$小时（180 分钟）进行规划。由于题目假设选手行进速度为$6\,km/h$，且要求选手完成比赛时间不低于 2 小时，为避免出现“相跟”现象的发生，我们首先设置时间间隔t_0=2 min，如后续模型改进需要，我们对t_0进行修改优化即可。

5.1.2 路线规划模型建立

对候选点的坐标数据处理得到打卡点分层聚类图，如图 5.4 所示：

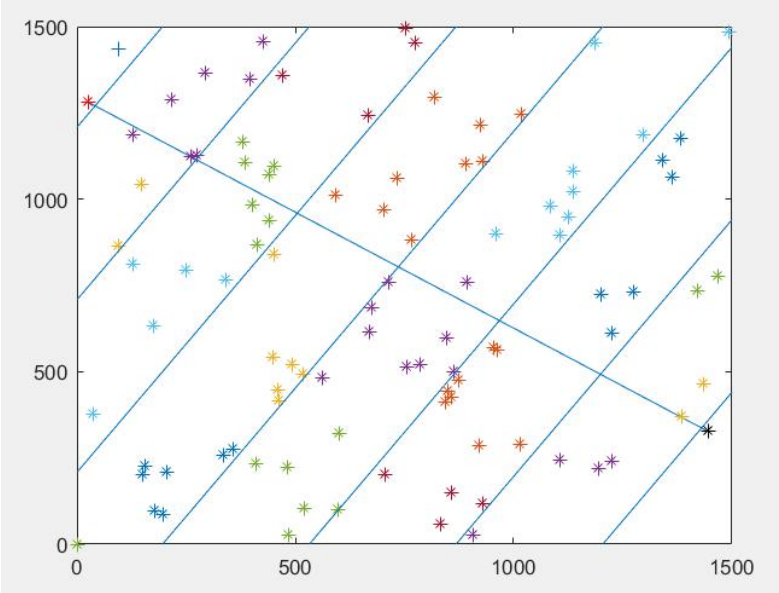

图 5.4 候选点分层示意图

选手在从起点p_1到终点p_{84}的行进过程中需满足平均时间不低于 2 小时、比赛的总完成时间最长不超过 3 个小时的同时，尽可能多的让参赛选手参与比赛。结合题目所要求的规则（1）和（2），这是一个多目标动态规划问题。

设第i条设计的路线路径$p_1\to\ldots\to p_i\to\ldots\to p_{98}\to\ldots\to p_j\to\ldots\to p_{25}\to\ldots\to p_{84}$为$p_1\ldots p_{98}\ldots p_{25}\ldots p_{84}$，其含有$S_i$个打卡点，我们将其按顺序编码为$p_{a_1}p_{a_2}\ldots p_{a_{S_i}}$。

目标：

不同路线的长度尽可能均匀，即不同路线长度差之和最小：

$$\min \sum_{i<j,i,j\in M}\left(L_i-L_j\right) \tag{5.1}$$

其中，第i条路线的总长度为$L_i=\sum_{j=1}^{S_i-1}d_{a_j,a_{j+1}}$，$S_i$为该路线所含打卡点数量。

不同路线的总打卡难度尽可能均匀，即不同路线总打卡难度差之和最小：

$$\min\sum_{i<j}\left(Q_i-Q_j\right) \tag{5.2}$$

其中，第i条路线的总难度为$Q_i=\sum_{j=1}^{S_i}q_{a_j}$。

尽可能多参赛选手能够加入到此次比赛，即总参赛选手数量最大：

$$\max N \tag{5.3}$$

其中，$N=\sum_{i=1}^{M} n_i$，n_i为第i条路线选手数量。

约束条件：

参赛选手完成任务的平均时间不低于 2 小时（120 分钟），即：

$$\bar{t} \geq 120，\ \bar{t}=\sum_{i=1}^{N} t_i \Big/ N \tag{5.4}$$

其中，若第i个选手挑战第j条赛道，则该选手完成比赛所花费的时间t_i为$t_i = L_i / \bar{v} + Q_i$。

每条路线中打卡点数量不低于 20 个，即：

$$S_i \geq 20 \left(\forall i \in M\right) \tag{5.5}$$

竞赛的总完成时间最长不超过 3 个小时（180 分钟），即：

$$T \leq 180 \tag{5.6}$$

其中，总完成时间为最后一名选手完成比赛所需要的时间与其距离第一批出发所间隔的时间，即$T = t_{last} + (c-1)t_0$，其中c为最后一批选手出发批次。

综上所述，路线设计的多目标规划为：

$$\begin{aligned} &\min \sum_{i<j, i, j \in M} \left(L_i - L_j\right) \\ &\min \sum_{i<j} \left(Q_i - Q_j\right) \\ &\max N \end{aligned} \tag{5.7}$$

$$s.t \begin{cases} \bar{t} \geq 120 \\ S_i \geq 20, \forall i \in M \\ T \leq 180 \end{cases} \tag{5.8}$$

5.1.3 改进遗传算法[3]的路线规划求解

（1）遗传算法流程

本问题我们选用遗传算法来寻找路径，遗传算法是一种智能算法，其基本原理是仿效生物界中的“物竞天泽、适者生存”的演化法则。遗传算法是把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及变异等运算来交换种群中染色体信息，最终生成符合优化目标的染色体。

遗传算法多目标规划问题的流程图如图 5.5 所示：

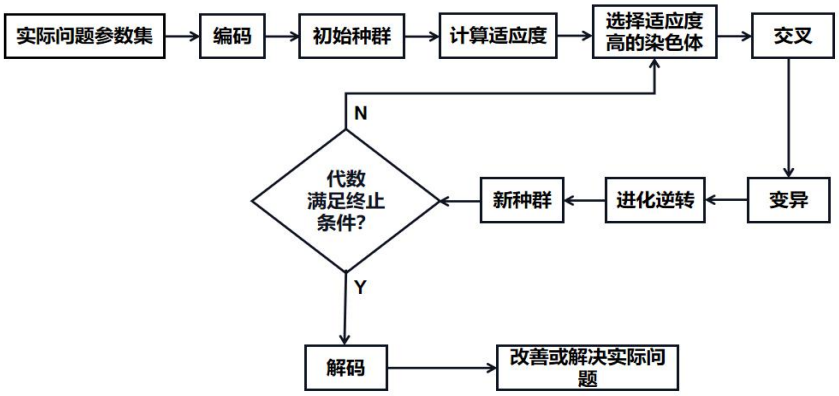

图 5.5 遗传算法多目标规划问题的求解示意图

我们对本题用到的遗传算法参数进行设定如下：

a）种群大小 $M_{种群}=200$；

b）最大代数 $G=200$；

c）交叉率 $P_0=1$，交叉概率为 1 能保证种群充分进化；

d）变异率 $P_V=0.1$，一般而言，变异发生的可能性比较小。

（2）遗传算法实现

Step 1:编码

采用整数排列编码方法，对 n 个打卡点的路径寻找问题，染色体分为 n 段，其中每一段为对应的打卡点的编号，如对 10 个打卡点的路径寻找问题 $\{1,2,3,4,5,6,7,8,9,10\}$，则 $1|10|2|4|5|6|7|9|3$ 就是一个合法的染色体。但在本问题中，起点终点以及必经点已经确定，故在编码时染色体应为 $1|...|98|...|25|...|84$。

Step 2:种群初始化

在完成染色体编码后，必须产生一个初始种群作为起始解，所以首先需要决定初始化种群的数目。初始化种群的数目一般根据经验得到，一般情况下种群的数量视打卡点的规模大小而确定。群体规模越大，搜索的范围越广，但是每一代的遗传操作时间越长。反之，群体规模越小，每代的运算时间越短，然而，搜索空间越小。用正整数向量集 $(P_1,P_2,...,P_{200})$ 作为初始种群，初始群体的每个个体时随机产生的。但在本问题中，种群染色体需满足第一段为 1（起点），最后一段为 84（终点）。然后判断染色体 $P_n(n=1,2,...,200)$ 是否符合约束条件。

Step 3:适应度函数

适应度是衡量个体优劣、执行遗传算法“优胜劣汰”的依据。在本问题中，采用整数编码，设 $a_1|a_2|...|a_i|...|a_n$ 为一个染色体 P_i，$d_{a_ia_j}$ 为打卡点 a_i 到 a_j 的距离，q_{a_i} 打卡点 a_i 的打卡难度。则行进路途所用时间

$$T=\sum_{i=1}^{n-1}d_{a_ia_j}\Big/100$$

打卡所用的时间：

$$Q=\sum_{i=1}^{n}q_{a_i}$$

则在种群$(P_1,P_2,...,P_N)$中，选手平均路途完成时间为$\overline{T}_{路途}$，平均打卡时间为$\overline{T}_{打卡}$，第i个选手路途时间方差为：

$$\sigma_T^2=\frac{(T-\overline{T}_{路途})^2}{2}$$

打卡时间方差为：

$$\sigma_Q^2=\frac{(Q-\overline{T}_{打卡})^2}{2}$$

则适应度函数为：

$$fitness=\frac{1}{\sigma_T+\sigma_Q} \tag{5.9}$$

即适应度函数为路径与平均时间的标准差与平局打卡时间的标准差的和的倒数。优化的目标就是选择适应度函数值尽可能大的染色体，适应度函数值越大的染色体越优质，反之越劣质。

Step4:选择操作

选择操作即从旧群体中以一定的概率选择个体到新群体中，个体被选中的概率跟适应度值有关，个体适应度值越大，被选中的概率越大。

Step5:交叉操作

在遗传算法中，交叉是产生新个体的主要手段。在整数优化问题中，采用综合交换的方法。首先，随机确定交换点λ。为满足约束条件,在区间[1,100]内产生均匀分布的随机整数，该整数便是交换点的位置。将父代的个体写作：

$$父代个体1:(P_1,P_2,...,P_{\lambda-1},P_\lambda,P_{\lambda+1},...,P_{100})$$
$$父代个体2:(P_1',P_2',...,P_{\lambda-1}',P_\lambda',P_{\lambda+1}...,P_{100}')$$

接下来计算交叉子代的染色体：

$$父代个体1:(P_1,P_2,...,P_{\lambda-1},P_\lambda',P_{\lambda+1}',...,P_{100}')$$
$$父代个体2:(P_1',P_2',...,P_{\lambda-1}',P_\lambda,P_{\lambda+1},...,P_{100})$$

交叉完成后，验证产生的新个体是否满足约束提交，若不满足则重新执行交叉操作，知道满足为止。

Step6:变异操作

变异是实现群体多样性的一种手段，同时也是全局寻优的保证。按照给定的变异概率，对选定变异的个体，随机选取两个点，将其对换位置。产生两个[1,100]范围内的随机整数r_1和r_2，确定两个位置，将其对换位置，如$r_1=2$，$r_2=28$

$$a_1|a_2|a_3|...|a_{27}|a_{28}|a_{29}|...|a_n$$

变异后为

$$a_1 | a_{28} | a_3 | ... | a_{27} | a_2 | a_{29} | ... | a_n$$

Step7:进化逆转操作

为改善遗传算法的局部搜索能力，在选择、交叉、变异之后引进连续多次的进化逆转操作，这里的“进化”是指逆转算子的单方向性，即只有经逆转后，适应度值有提高的才接受下来，否则逆转无效。

产生两个[1,100]范围内的随机整数 r_1 和 r_2，确定两个位置，将其对换位置，如 $r_1=2$，$r_2=28$

$$a_1 | a_2 | a_3 | ... | a_{27} | a_{28} | a_{29} | ... | a_n$$

进化逆转后为

$$a_1 | a_{28} | a_3 | ... | a_{27} | a_{28} | a_{29} | ... | a_n$$

（3）模型求解

将打卡点坐标数据、难度数据、选手行进速度等已知数据代入上述算法，我们得到十条路线的设计方案如下所示：

表 5.1 各路线检查点表

路线	1	2	3	4	5	6	7	8	9	10
起点	1	1	1	1	1	1	1	1	1	1
2 号	9	60	9	9	60	60	63	60	9	63
3 号	24	93	24	24	93	93	28	93	24	28
4 号	47	57	47	47	57	57	85	57	47	14
5 号	4	14	4	4	50	50	18	23	4	98
6 号	36	98	50	23	23	23	98	26	29	87
7 号	72	65	23	26	56	26	65	56	36	59
8 号	46	6	56	90	41	90	6	41	46	61
9 号	42	12	41	41	98	98	12	98	42	99
10 号	98	3	98	98	65	8	3	65	98	92
11 号	87	48	6	17	6	20	77	6	87	11
12 号	59	77	12	27	3	96	58	3	59	81
13 号	61	83	3	68	76	45	100	76	61	16
14 号	99	44	76	62	31	78	44	20	99	10
15 号	20	70	31	83	92	10	69	96	5	71
16 号	96	52	11	79	11	71	95	45	80	82
17 号	45	64	81	94	81	82	55	78	7	25
18 号	78	21	2	44	10	25	75	69	35	19
19 号	10	25	10	10	71	19	25	54	66	91
20 号	71	37	71	71	82	91	34	95	73	89
21 号	82	32	82	82	25	89	84	55	53	84
22 号	25	22	25	25	34	84	0	25	25	0
23 号	37	84	91	91	0	0	0	91	91	0
24 号	22	0	89	89	0	0	0	89	89	0
25 号	33	0	84	84	0	0	0	84	84	0
26 号	84	0	0	0	0	0	0	0	0	0

其中 0 代表以及完成比赛，到达终点 84 号，无需继续设置检查点。

通过具体路线，我们可得各路线总长度为：

表 5.2 各路线总长度表

路线	总长度（米）	路线	总长度（米）
1	5575.4378	6	6419.1779
2	5977.1139	7	7291.6050
3	6259.3307	8	5670.9699
4	6188.7459	9	5816.3699
5	6629.0550	10	7234.1027

各路线总难度为：

表 5.3 各路线总难度表

路线	总难度（分钟）	路线	总难度（分钟）
1	65	6	57
2	65	7	49
3	66	8	65
4	63	9	63
5	61	10	53

各路线不同难度打卡点个数统计如下：

表 5.4 路线难度点数量表

路线	难度 1	难度 2	难度 3	难度 4	路线	难度 1	难度 2	难度 3	难度 4
1	10	3	7	7	6	7	1	8	6
2	5	5	6	8	7	7	5	4	5
3	6	4	12	4	8	7	5	8	6
4	9	3	8	6	9	10	2	7	7
5	4	5	9	5	10	7	2	6	6

各路线参赛选手数量为：

表 5.5 各路线参赛选手数量表

路线	选手数量（名）	路线	选手数量（名）
1	30	6	30
2	28	7	29
3	26	8	29
4	28	9	30
5	27	10	28

由上表可得，总参赛选手数量 $N=285$。

5.1.4 模型优化

（1）优化思想及过程

由表 5.1 得，路线最多含有 26 个检查点，最少含有 21 个打卡点，过多过少关卡造成对不同选手的区别对待。观察表 5.2 发现路线最长为 7291.6050 米，最短为 5575.4378

米，虽整体均值满足完成比赛不低于 2 小时的要求，但是相差达到了 1716.672 米，差距过大。由表 5.3 发现，路线总难度最大为 65，最小为 49，难度差达到 16。以上这些设计均导致对参赛选手不公平的后果。故我们进一步对模型进行修改。

首先观察表 5.6 不同路线长度距离方差数据：

表 5.6 不同路线长度距离方差表（前）

距离方差	1	2	3	4	5	6	7	8	9	10
1	0	4.0335	11.6927	9.4036	27.7527	17.7974	73.6307	0.2281	1.4512	68.7792
2	4.0335	0	1.9911	1.1197	10.6256	4.8855	43.1971	2.3431	0.64596	39.5005
3	11.6927	1.9911	0	0.1245	3.4174	0.6387	26.6397	8.6542	4.9053	23.7545
4	9.4036	1.1197	0.1245	0	4.8468	1.3274	30.4074	6.7022	3.4665	27.3192
5	27.7527	10.6256	3.4174	4.8468	0	1.1012	10.9743	22.9481	16.5114	9.1520
6	17.7974	4.8855	0.6387	1.3274	1.1012	0	19.0282	13.9953	9.0844	16.6025
7	73.6307	43.1971	26.6397	30.4074	10.9743	19.0282	0	65.6614	54.4079	0.0826
8	0.2281	2.3431	8.6542	6.7022	22.9481	13.9953	65.6614	0	0.5285	61.0846
9	1.4512	0.6459	4.9053	3.4665	16.5114	9.0844	54.4079	0.5285	0	50.2491
10	68.7792	39.5005	23.7545	27.3192	9.1520	16.6025	0.0826	61.0846	50.2491	0

我们发现编号为 1、2、7、8、10 的路线长度与其他五条路线长度距离方差过大，即其长度与剩余路线相差很大，对参赛选手相对不公，故我们考虑去除这五条路线，选择剩余五条更佳符合条件的路线。减少线路后，将出发时间间隔从 2 分钟减少至 1 分钟，通过计算可得每条路线上选手的数量。

修改模型之后，不同路线长度距离方差数据如表 5.7 所示：

表 5.7 不同路线长度距离方差表（后）

距离方差	1	2	3	4	5	6	7	8	9	10
1	0	0	0.25	1	4	16	64	0	1	36
2	0	0	0.25	1	4	16	64	0	1	36
3	0.25	0.25	0	2.25	6.25	20.25	72.25	0.25	2.25	42.25
4	1	1	2.25	0	1	9	49	1	0	25
5	4	4	6.25	1	0	4	36	4	1	16
6	16	16	20.25	9	4	0	16	16	9	4
7	64	64	72.25	49	36	16	0	64	49	4
8	0	0	0.25	1	4	16	64	0	1	36
9	1	1	2.25	0	1	9	49	1	0	25
10	36	36	42.25	25	16	4	4	36	25	0

对比观察可见，方差有明显大幅度减小。

并且，经检验，在筛选之前平均距离方差为 32.5078，平均难度方差为 30.4100；在筛选之后平均距离方差降低为 7.2678，平均难度方差降低为 8.8，且不同路线打卡点数最大仅相差 3，路线总长度与总难度更为均匀，更加符合题目要求。

（2）优化后结果

优化之后，所得路径为表 5.1 中编号为 3、4、5、6、9 的五条路线，重新对其标号为 1、2、3、4、5，将结果存于附件 4“第 1 问结果”表中。五条路径的图像如下所示：

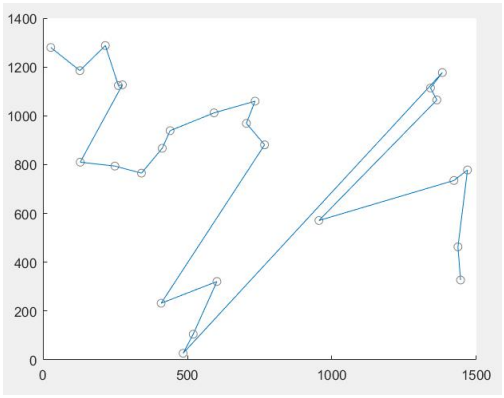

图 5.6 编号为 1 的路线示意图

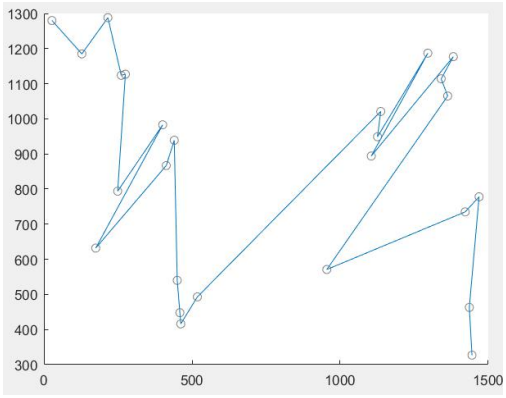

图 5.7 编号为 2 的路线示意图

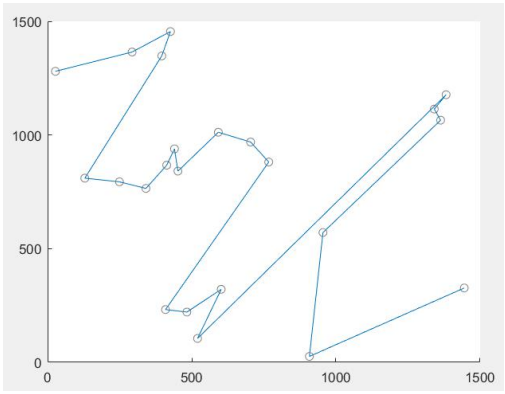

图 5.8 编号为 3 的路线示意图

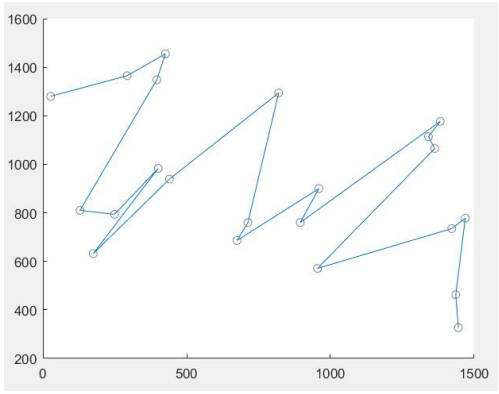

图 5.9 编号为 4 的路线示意图

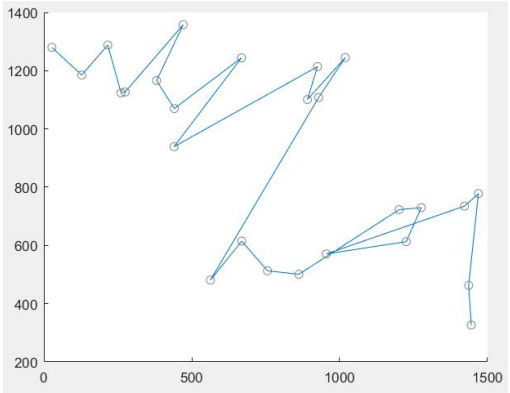

图 5.10 编号为 5 的路线示意图

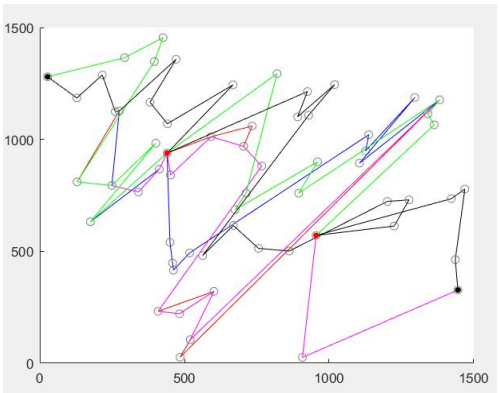

图 5.11 五条路线总示意图

经验证，我们所得设计的五条路线中各个打卡点同时到达选手数量都不超过 5，满足题目所提出的要求（7）。

各路径总长分别为：

表 5.8 各路线总长度表

路线	1	2	3	4	5
总长度(米)	6259.3307	6188.7459	6629.0550	6419.1779	5816.3699

各路线总难度分别为：

表 5.9 各路线总难度表

路线	1	2	3	4	5
总难度	66	63	61	57	63

各路径参赛选手数量分别为：

表 5.10 各路线参赛选手数量表

路线	1	2	3	4	5
选手数量（名）	59	55	51	55	53

总参赛选手数量为 273 名，与优化前总人数相比，数量减少了 12 名，但是各路线的设计情况，如总长度、总难度及打卡点数量更加均匀，较全面的满足题目要求，故该结果是可接受的。

5.2 问题二模型建立与求解

问题二仅在问题一的基础上，给出打卡点 Z 轴数据，增加了一个目标函数，其余条件均未改变。故对问题二的模型建立与求解过程与问题一类似，只需修改所建立的多目标规划模型，修改算法中相应输入条件即可。问题二的数据处理、模型准备以及终点和必经点的选取过程与问题一步骤相同，不再重复叙述，得到的结果为：距离出发点相对最远且难度最低的候选检查点 p_{131} 为终点，检查点 p_{29} 和 p_{115} 为必经打卡点，如图 5.12、图 5.13 所示：

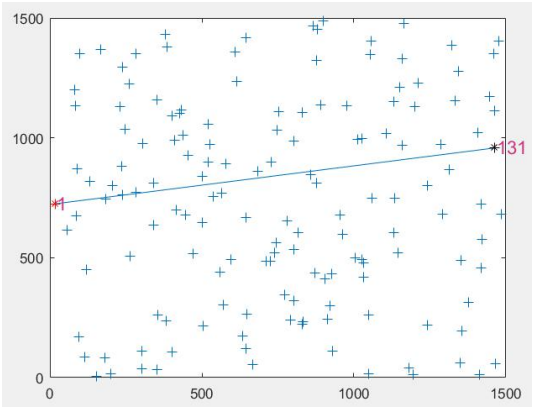

图 5.12 终点选取示意图

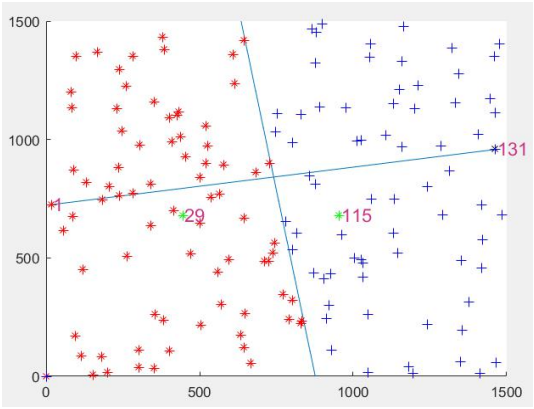

图 5.13 必经打卡点选取示意图

5.2.1 问题二模型建立

结合题目所要求的规则（3），问题二仍然是一个多目标动态规划问题。新增目标函数为：

每条比赛路线的爬高量尽可能均匀，即不同路线爬高量的差之和最小，即：

$$\min \sum_{i<j,\ i,j\in M} \left(H_i - H_j\right) \tag{5.10}$$

其中，第i条路线的爬高量$H_i=\sum_{j=1}^{S_i-1}h_{a_j,a_{j+1}}$，$h_{a_j,a_{j+1}}=\left|z_{a_j}-z_{a_{j+1}}\right|$。

结合目标函数（5.7）与约束条件（5.8），得到新的多目标规划模型为：

目标函数：

$$\begin{aligned}&\min\sum_{i<j,i,j\in M}\left(L_i-L_j\right)\\&\min\sum_{i<j}\left(Q_i-Q_j\right)\\&\max N\\&\min\sum_{i<j,\ i,j\in M}\left(H_i-H_j\right)\end{aligned}\qquad(5.11)$$

约束条件：

$$s.t\begin{cases}\bar{t}\geq 120\\S_i\geq 20,\forall i\in M\\T\leq 180\end{cases}\qquad(5.12)$$

5.2.2 问题二模型求解

考虑上述目标函数和约束条件，与问题一的处理步骤类似，结合打卡点坐标数据、难度数据、选手行进速度等已知条件，利用遗传算法求解，得到5条路线的设计方案见附件4“第2问结果”表中。五条路径的图像如下所示：

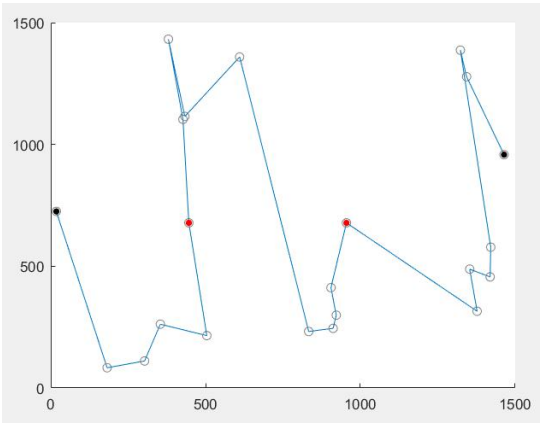

图5.14 编号为1的路线示意图

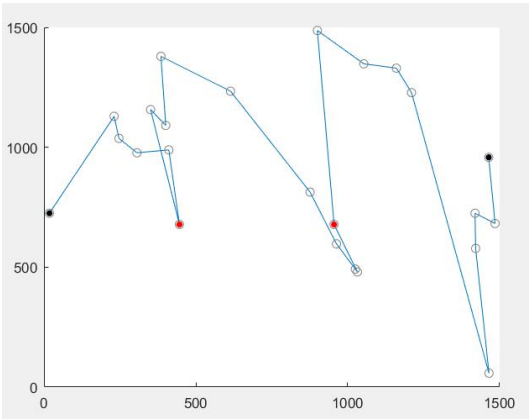

图5.15 编号为2的路线示意图

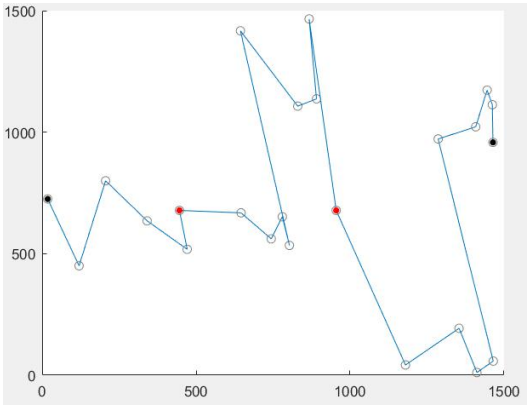

图 5.16 编号为 3 的路线示意图

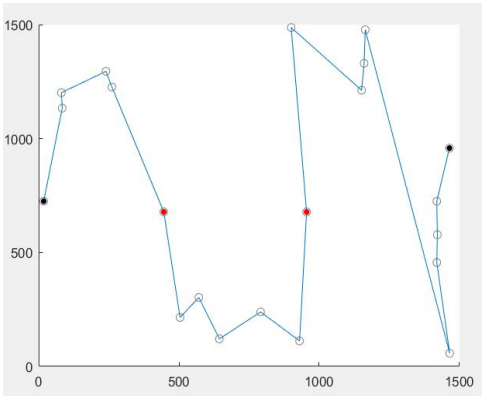

图 5.17 编号为 4 的路线示意图

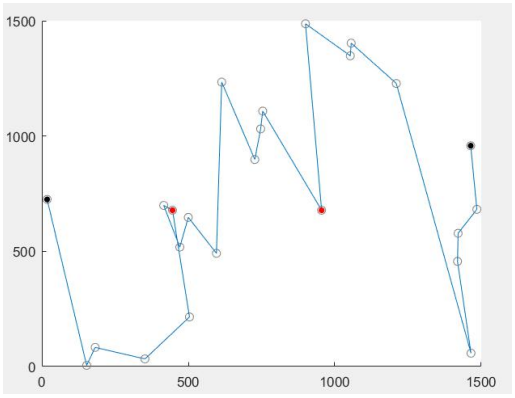

图 5.18 编号为 5 的路线示意图

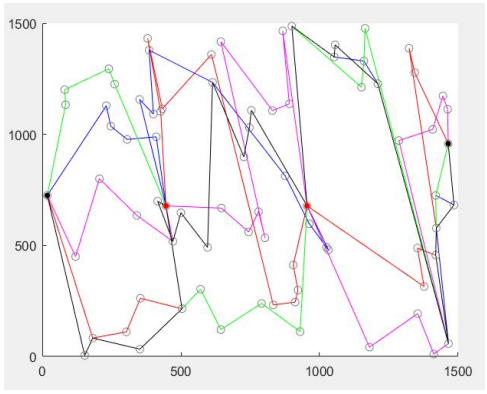

图 5.19 五条路线总示意图

通过具体路线信息，我们可得各路线总长度为：

表 5.11 各路线总长度表

路线	1	2	3	4	5
总长度（米）	6813.8799	6722.1017	6819.5667	6839.3868	7446.8900

各路线总难度为：

表 5.12 各路线总难度表

路线	1	2	3	4	5
总难度（分钟）	52	53	58	52	59

各路线不同难度打卡点个数统计如下：

表 5.13 路线难度点数量表

路线	1	2	3	4	5
难度 1	9	7	7	6	7
难度 2	2	7	6	4	6
难度 3	5	8	5	6	4
难度 4	6	2	6	5	7

各路线爬高量为：

表 5.14 各路线爬高量表

路线	1	2	3	4	5
爬高量（米）	36	38	38	35	38

各路线参赛选手数量为：

表 5.15 各路线参赛选手数量表

路线	1	2	3	4	5
选手数量（名）	60	60	54	60	47

由上表可得，总参赛选手数量 $N=281$。

5.3 问题三模型建立与求解

问题三的数据处理、模型准备以及终点和必经点的选取过程与问题一步骤相同，不再重复叙述，得到的结果为：距离出发点相对最远且难度最低的候选检查点 p_{176} 为终点，检查点 p_3 和 p_5 为必经打卡点。

5.3.1 问题三模型建立

已知参赛选手总数为 $N=120$，各参赛选手的行进速度不同，记为 $v_i, i=1,...,120$ 且是已知量，起点选择序号为 1 的检查点，附件 3 给出起点和候选点的三维坐标及难度数据。

分析可得问题三的目标函数有：

目标函数一：使得使用的路线尽可能少，即

$$\min M \tag{5.13}$$

目标函数二：完成的时间尽可能短，即

$$\min T \tag{5.14}$$

目标函数三：不同路线长度尽可能均匀，即不同路线长度差之和最小：

$$\min \sum_{i<j,i,j\in M}\left(L_i-L_j\right) \tag{5.15}$$

其中，第 i 条路线的总长度为 $L_i=\sum_{j=1}^{S_i-1} d_{a_j,a_{j+1}}$，$S_i$ 为该路线所含打卡点数量。

目标函数四：不同路线总打卡难度尽可能均匀，即不同路线总打卡难度差之和最小：

$$\min \sum_{i<j}\left(Q_i-Q_j\right) \tag{5.16}$$

其中，第 i 条路线的总难度为 $Q_i=\sum_{j=1}^{S_i} q_{a_j}$。

目标函数五：不同路线的爬高量尽可能均匀，即不同路线爬高量的差之和最小，即：

$$\min \sum_{i<j,\ i,j\in M}\left(H_i-H_j\right) \tag{5.17}$$

其中，第i条路线的爬高量$H_i=\sum_{j=1}^{S_i-1}h_{a_j,a_{j+1}}$，$h_{a_j,a_{j+1}}=\left|z_{a_j}-z_{a_{j+1}}\right|$。

分析可得问题三的约束条件有：

约束条件一：平均完成时间不低于 2 小时（120 分钟），即：

$$\bar{t}\geq 120，\ \bar{t}=\sum_{i=1}^{N}t_i\Big/N \tag{5.18}$$

约束条件二：比赛总完成时间不超过 3.5 个小时（210 分钟），即

$$T\leq 210 \tag{5.19}$$

其中，总完成时间为最后一名选手完成比赛所需要的时间与其距离第一批出发所间隔的时间，即$T=t_{last}+(c-1)t_0$，其中c为最后一批选手出发批次。

约束条件三：每条线路中检查点数量不低于 20 个，即

$$S_i\geq 20(\forall i\in M) \tag{5.20}$$

综上所述，可得问题三所建立的多目标规划为：

$$\begin{aligned}&\min M\\&\min T\\&\min\sum_{i<j,i,j\in M}(L_i-L_j)\\&\min\sum_{i<j}(Q_i-Q_j)\\&\min\sum_{i<j,\ i,j\in M}(H_i-H_j)\end{aligned} \tag{5.21}$$

$$s.t\begin{cases}\bar{t}\geq 120\\T\leq 210\\S_i\geq 20(\forall i\in M)\end{cases} \tag{5.22}$$

5.3.2 问题三模型求解

考虑上述目标函数和约束条件，与问题一的处理步骤类似，结合打卡点坐标数据、难度数据、选手行进速度等已知条件，利用遗传算法求解，得到路线最少为 2 条，两条路线的详细信息存于表附件 4“第 3 问结果”中，比赛完成的时间最短为 185.9408 分钟，各路线相关情况如下所示：

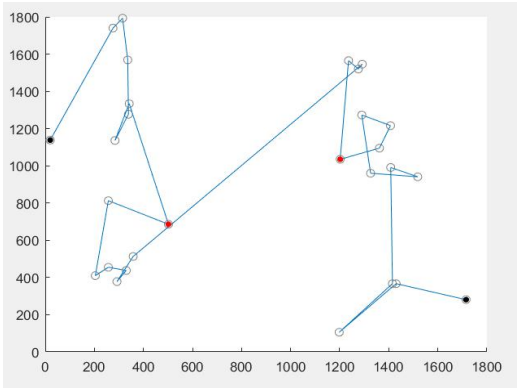

图 5.20 编号为 1 的路线示意图

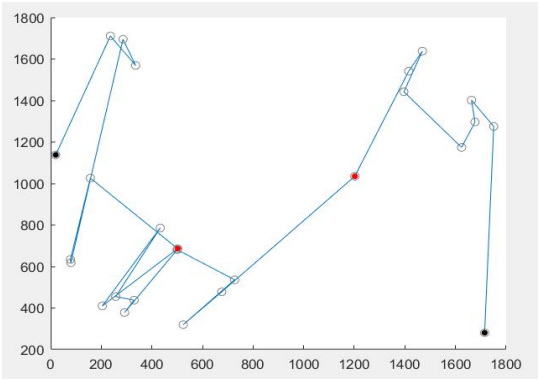

图 5.21 编号为 2 的路线示意图

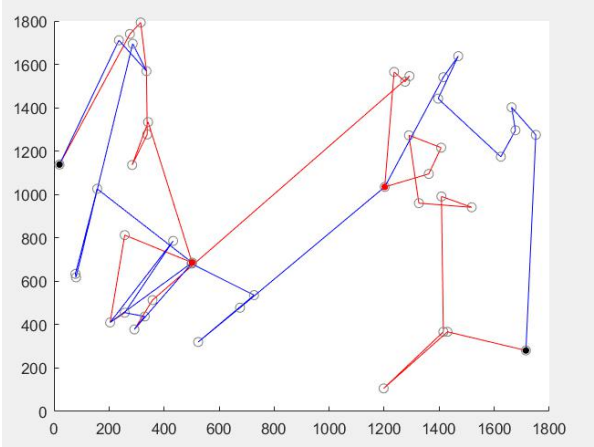

图 5.22 两条路线总示意图

通过具体路线，我们可得各路线总长度为：

表 5.16 各路线总长度表

路线	1	2
总长度（米）	79.9408	88.1457

各路线总难度为：

表 5.17 各路线总难度表

路线	1	2
总难度（分钟）	106	105

各路线不同难度打卡点个数统计如下：

表 5.18 各路线不同难度打卡点个数表

路线	难度 1	难度 2	难度 3	难度 4	难度 5	难度 6
1	1	6	5	7	4	5
2	1	2	6	6	8	3

各路线爬高量为：

表 5.19 各路线参赛选手数量表

路线	1	2
选手数量（名）	65	55

各路线参赛选手数量为：

表 5.20 各路线爬高量表

路线	1	2
爬高量（米）	74	67

6. 模型评价

6.1 模型的优点

首先，基于本题背景所建立的模型逻辑性强，模型建立过程严谨合理，更大程度地把题中所有条件都考虑进去，尽量少的使用假设。这样可以使得该模型的扩展性好，更加贴合实际，更容易被推广到其他问题上去，给其他问题提供了一种较为普适的建模思路。

其次，对模型中的输入变量加入一个比较小的误差扰动，再通过模型求解以后发现解的变化并不是很大，这说明了该模型的稳定性较好。

最后，我们在建模的过程中，发现解空间太大，直接求解会导致很难求得比较好的解，因此在设计求解算法的时候，基于本题对遗传算法的产生新解的规则做了改进，即加入了解的预判方法，当某一个解的适应度值不高从而面临着被淘汰的地步时，下一代该解向量以及解空间中与其相近的解向量都不会再次出现，这样一来，随着遗传算法迭代次数的增加，解空间的规模会极速减少。

6.2 模型的缺点

首先，本题中采用的这种建模方式虽然简单易懂，但是如果不对解空间的规模加以控制作那么将很难求出较好的解，虽然针对本题我们对遗传算法进行了改进，但是对于现实生活中另外一个问题我们就不一定能够很好的控制解空间规模。

其次，模型的建立过程中，为了更方便求解，将参赛人员数量和他们经过的路径节点都假定为已知输入参数，而且令每个人经过路径的节点数量尽量都是一样的，这样虽然可以求出比较好的解，但是这种假定是不太符合现实的。这一点在以后还有待改进。

7. 参考文献

[1] 叶灶荣.我国休闲类定向运动赛事的组织管理研究[D].中国地质大学(北京),2018.
[2] 吴斌.离散数据的归一化处理在计税核价系统中应用研究[J].微型电脑应用，2020，36(10):165-167.
[3] 郁磊，史峰，王辉，胡斐.MATLAB 智能算法 30 个案例分析（第 2 版）[M].北京：北京航空航天大学出版社，2015.8.

[4]贾普俊.山地定向越野比赛场地选择与路线图设计规律研究[D].山西师范大学,2012.
[5]曹园,潘思齐,王涛.基于优化蚁群算法的景区“有时间窗”浏览路线设计模型——以安略湖风景区浏览路线设计为例[J].中国商论,2020(03):241-242.
[6]田增瑞,赵阳,赵袁军.基于遗传算法与蚁群算法的最佳旅游路线设计[J].数学的实践与认识,2016,46(24):41-48.
[7] MathWorks Inc..Genetic Algorithm and Direct Search Toolbox--MATLAB® version 2.3, User's guide, 2008.
[8] KALYANMOY D,AMRIT P,SAMEERA,et al.A Fast and Elitist Multiobjective Genetic Algorithm:NSGA-II[J].IEEE Transactions on Evolutionary Computation,2002,6(2):182-197.

附　录

附录一：　问题一主函数 problem1_new.m

```
clear
clc
load X
load Y
load  data_2
load Ha
load Data1
XA=D1;
NIND=200;        %种群大小
MAXGEN=200;
Pc=0.9;          %交叉概率
Pm=0.05;         %变异概率
GGAP=0.9;       %代沟(Generation gap)
D=Distanse(XA);  %生成距离矩阵
N=size(D,1);

Chrom=InitPop(NIND,N);  %种群数量
gen=0;
figure;
hold on;box on
xlim([0,MAXGEN])
title('优化过程')
xlabel('代数')
ylabel('最优值')
ObjV=PathLength(D,Chrom);  %计算路线长度
preObjV=min(ObjV);
while gen<MAXGEN
    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);
    SelCh=Select(Chrom,FitnV,GGAP);
    SelCh=Recombin(SelCh,Pc);
    SelCh=Mutate(SelCh,Pm);
    SelCh=Reverse(SelCh,D);
    Chrom=Reins(Chrom,SelCh,ObjV);
    gen=gen+1 ;
end
PL=[3 4 5 4 4 3]; %每层的类数
PM=[2 4 3 0 0    %每类的个数
    6 2 6 1 0
```

```
    4 1 5 5 7
    6 6 8 4 0
    6 3 5 3 0
    4 3 1 0 0];
Chrom=zeros(1,30);
Chrom(1,1)=1;
XX = [X,Y];
% DrawPath_1(Chrom ,XX)
ans1 = zeros(5,30);
g = 1;
while g<6
    flag=2;
    for i=1:length(PL)
        Ram=0;
             Ramk=randperm(PL(i));
             Ram= Ramk(1);
             if i == 3
                 Chrom(flag)=98;
                 flag = flag+1;
             end
             if i == 6
                 Chrom(flag)=25;
                 flag = flag+1;
             end
             if PM(i,Ram)<5
                 for k=1:4
                     if data_2(Ram,k,i)~=0
                     Chrom(flag)=data_2(Ram,k,i);
                     flag=flag+1;
                     else
                         break
                     end
                 end
             else
                 ch=4;
                 Ram1=random1(PM(i,Ram),ch);
                 for k=1:ch
                     Chrom(flag)=data_2(Ram,Ram1(k),i);
                     flag=flag+1;
                 end
             end

    end
    Chrom(flag)=84;
```

```
    Chrom
    flag;
    DIS=Distanse_1(Chrom,X,Y);
     H=Hard_1(Chrom(1:flag),Ha );
     SUM=DIS+H;
    DrawPath_1(Chrom(1:flag) ,XX);
    set(gca,'XLim',[0,1500]);
    set(gca,'YLim',[0,1500]);

    if SUM>90
        ans1(g,1:flag)=Chrom(1:flag);
        g = g+1;
    end
end
g
```

附录二：问题二主函数 problem2_new.m

```
clear
clc
load X_2
load Y_2
load  wt2data
load Ha_2
[n,m,r] =size(wt2data);
PM = zeros(5,5);
load Data1
XA=D1;
NIND=200;        %种群大小
MAXGEN=200;
Pc=0.9;          %交叉概率
Pm=0.05;         %变异概率
GGAP=0.9;       %代沟(Generation gap)
D=Distanse(XA);  %生成距离矩阵
N=size(D,1);

Chrom=InitPop(NIND,N);  %种群数量
gen=0;
figure;
hold on;box on
xlim([0,MAXGEN])
title('优化过程')
xlabel('代数')
ylabel('最优值')
ObjV=PathLength(D,Chrom);  %计算路线长度
```

```
preObjV=min(ObjV);
while gen<MAXGEN
    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);
    SelCh=Select(Chrom,FitnV,GGAP);
    SelCh=Recombin(SelCh,Pc);
    SelCh=Mutate(SelCh,Pm);
    SelCh=Reverse(SelCh,D);
    Chrom=Reins(Chrom,SelCh,ObjV);
    gen=gen+1 ;
end
for t = 1:r
    for i = 1:n
        if wt2data(i,1,t)==0
            break;
        end
        for j=1:m
            if wt2data(i,j,t)==0
                break;
            end
            PM(t,i)=j;
        end
    end
end
PL=[4 5 5 5 4]; %每层的类数
XX=[X_2 ,Y_2];

ans2 = zeros(15,30);
g = 1;
while g<6
    Chrom=zeros(1,30);
    Chrom(1,1)=1;
    flag=2;
    for i=1:length(PL)     %i 为在第几层
            Ram=0;
            Ramk=randperm(PL(i));
            Ram= Ramk(1);
            if i == 2
                Chrom(flag)=29;
                flag = flag+1;
            end
            if i == 4
```

```
                Chrom(flag)=115;
                flag = flag+1;
            end
            if PM(i,Ram)<4
                for k=1:4
                    if wt2data(Ram,k,i)~=0
                    Chrom(flag)=wt2data(Ram,k,i);
                    flag=flag+1;
                    else
                        break
                    end
                end
            else
                ch=4;
                Ram1=random1(PM(i,Ram),ch);
                for k=1:ch
                    Chrom(flag)=wt2data(Ram,Ram1(k),i);
                    flag=flag+1;
                end
            end

    end
    Chrom(flag)=131;
    Chrom
    flag
    DIS=Distanse_1(Chrom(1:flag+1),X_2,Y_2);
    H=Hard_1(Chrom(1:flag),Ha_2 );
    SUM=DIS+H;
    %DrawPath_1(Chrom(1:flag) ,XX);
    set(gca,'XLim',[0,1500]);
    set(gca,'YLim',[0,1500]);
    if SUM>120
        ans2(g,1:flag)=Chrom(1:flag);
        DrawPath_1(Chrom(1:flag) ,XX);
        g = g+1;
    end
end
ans2_jg = ans2;
```

附录三：问题三主函数 problem3_new.m

```
clear
clc
load ans3_jg
load X_3
```

```
load Y_3
load  data_3
load Ha_3
[n,m,r] =size(data_3);
PM = zeros(8,6);
load Data1
XA=D1;
NIND=200;        %种群大小
MAXGEN=200;
Pc=0.9;          %交叉概率
Pm=0.05;         %变异概率
GGAP=0.9;       %代沟(Generation gap)
D=Distanse(XA);  %生成距离矩阵
N=size(D,1);

Chrom=InitPop(NIND,N);  %种群数量
gen=0;
figure;
hold on;box on
xlim([0,MAXGEN])
title('优化过程')
xlabel('代数')
ylabel('最优值')
ObjV=PathLength(D,Chrom);  %计算路线长度
preObjV=min(ObjV);
while gen<MAXGEN
    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);
    SelCh=Select(Chrom,FitnV,GGAP);
    SelCh=Recombin(SelCh,Pc);
    SelCh=Mutate(SelCh,Pm);
    SelCh=Reverse(SelCh,D);
    Chrom=Reins(Chrom,SelCh,ObjV);
    gen=gen+1 ;
end
for t = 1:r
    for i = 1:n
        if data_3(i,1,t)==0
            break;
        end
        for j=1:m
            if data_3(i,j,t)==0
```

```
                break;
            end
            PM(t,i)=j;
        end
    end
end
PL=[2,3,2,4,4,5,3,3]; %每层的类数
XX=[X3 ,Y3];
% DrawPath_1(Chrom ,XX)
ans_3 = zeros(4,30);
g = 1;
while g<5
    Chrom=zeros(1,30);
    Chrom(1,1)=1;
    flag=2;
    for i=1:length(PL)     %i 为在第几层
            Ram=0;
            Ramk=randperm(PL(i)); %随机数，选择第 i 层的类
            Ram= Ramk(1);
            if i == 3
                Chrom(flag)=3;
                flag = flag+1;
            end
            if i == 6
                Chrom(flag)=5;
                flag = flag+1;
            end
            if PM(i,Ram)<2
                for k=1:2
                    if data_3(Ram,k,i)~=0
                    Chrom(flag)=data_3(Ram,k,i);
                    flag=flag+1;
                    else
                        break
                    end
                end
            else
                ch=3;
                Ram1=random1(PM(i,Ram),ch);%随机数，选择第 i 层的第 Ram 类的
前 3 个
                for k=1:ch
                    Chrom(flag)=data_3(Ram,Ram1(k),i);
                    flag=flag+1;
                end
```

```
            end

    end
    Chrom(flag)=176;
    Chrom
    flag
    DIS=Distanse_1(Chrom(1:flag+1),X3,Y3);
    H=Hard_1(Chrom(1:flag),Ha3 );
    SUM=DIS+H;
    %DrawPath_1(Chrom(1:flag) ,XX);
    set(gca,'XLim',[0,2000]);
    set(gca,'YLim',[0,2000]);
    if SUM>120
        ans_3(g,1:flag)=Chrom(1:flag);
        DrawPath_1(Chrom(1:flag) ,XX);
        g = g+1;
    end
end
ans3_jg = ans_3;
```