附件 2

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）研究和讨论与赛题有关的问题。

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

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

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

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：B

我们的参赛报名号为（如果组委会设置报名号的话）：20201800100B

所属学校（请填写完整的全名）：国防科技大学

参赛队员 (打印并签名)： 1. 曾伟健

2.

3. 贺惠晖

指导教师或指导教师组负责人 (打印并签名):

日期：2020年 11 月 30 日

评阅编号（由组委会评阅前进行编号）：

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。）

基于嵌套遗传算法的定向越野路线设计模型

摘　要

定向越野的路线设计是一个在一定约束条件下，制定终点、检查点、出发间隔时间、人员出发顺序等方案的多约束目标规划问题。本文根据附件和题目中所提供的检查点信息和选手速度等数据，抽象出路线规划过程中的决策变量，主要运用遗传算法和总路径数从少到多的寻优策略，在各个问题的具体约束条件下寻找符合目标的最优路线方案和选手分配方案。经过简单测试，总路径数在 3 到 7 之间为最优。本文通过逐一计算固定的总路径数下的最优方案，对适应度值进行比较，最终得到最优的路线设计方案。

问题一：本问的决策变量为每条路径的对检查点的选择情况、时间间隔、终点和必经检查点。**在选定每条路线的检查点的情况下，将其抽象成 TSP 问题，运用遗传算法求解检查点顺序，从而计算出约束中所需要判断的统计量。**然后计算每个个体的适应度和违反约束程度，根据二元锦标赛的选择策略进行精英选择，嵌套使用遗传算法，获得固定路径数下的最优方案。最终比较不同路径数的结果，得到**路径数为 3** 时，参与人数最多，为 **204** 人。

问题二：问题二和问题一的情景基本相同，目标和决策变量不变，在问题一的基础上增加了爬高量均匀的约束。得到**路径数为** 4 时，参与人数最多，为 **198** 人。

问题三：在问题二的基础上增加 120 维的决策变量，表示 120 名选手选择的路径。经过分析，制定了每条路线均按照先慢后快的出发顺序。改变了选手到达每个节点的时间的计算方法，同时目标变换为使比赛的总完成时间最短。最终比较不同路径数的结果，发现**路径数为 3** 时，比赛的总完成时间为 **118 分钟**，综合评价最优。

关键词：遗传算法、TSP 问题、邻域法、路线设计

1 问题重述和分析

1.1 问题背景

在定向越野比赛中，比赛路线一般由一个起点、若干个检查点和一个终点构成，希望根据所提供的候选检查点，设计若干条路线，满足以下要求：

1）不同路线长度尽可能均匀。

2）不同路线打卡总难度尽可能均匀。

3）不同路线的爬高量军可能均匀。

4）不同路线的路段重复率尽可能低，同时避免选手之间相距过近。

5）每条路线都要经过 2 个固定的检查点，并且总检查点数量不低于 20 个。

6）一条路线中相距过近的检查点尽量连续，检查点排序时要使得总路线长度尽量短。

7）到达同一检查点的选手数量需要控制在每分钟 5 人以内。

每条路线可以安排多名选手，按照一定的时间间隔错时出发。选手需要按照规定的打卡顺序依次前往检查点打卡。每个候选检查点都具备以下信息：平面坐标、高度、打卡难度。打卡难度是根据在该点进行打卡所花费的时间进行衡量，花费时间越多，打卡难度越大。检查点与检查点之间均存在路径可以直线连接，选手可以从任意检查点直线前往下一个检查点。

1.2 问题分析

本文中我们主要对以下问题进行研究和讨论：

问题一：根据附件 1 中的候选检查点平面坐标和打卡难度数据，设计有固定起点的若干条路线。要求每条路线选手完成时间不低于 2 小时，比赛总进行时间不超过 3 小时，使得本次比赛的参赛选手数量最大化。

分析：本问的目标是使得本次比赛参赛选手数量最大化，已知条件包括：起点、候选检查点的位置信息和打卡难度、参赛选手的平均速度。需要决策的内容包括：终点、总路线数量、每条路线的检查点、每条路线的参赛人数、每条路线的出发时间间隔。约束条件包括：要求 1）-7）（其中不考虑高程信息，因此要求 3）可以不计）、总比赛时间、每条路线的完成时间。可以归纳为一个多约束目标规划问题。

问题二：根据附件 2 中候选点的三维坐标，在问题一的基础上加入对爬高量的约束。

分析：基本做法与问题一类似，已知条件增加了候选检查点的高程信息，约束条件增加了要求 3）。

问题三：在问题 2 的基础上，增加了 120 名选手具有不同的速度的因素。

分析：由于选手速度不同，每个选手可以选择 1-N 条路径中的任意一条，同时需要对每条路径的选手的出发顺序进行排序。约束条件保持不变，在问题二的基础上需要添加每个选手选择的路径这一决策变量，以及制定每条路线上选手的出发顺序策略。

2 假设和说明

1） 在问题一中，假设在任意检查点之间的运动速度都是恒定的，点与点之间可以直线运动，计算两点之间距离时按照空间直线距离计算；
2） 假设每条路线均严格遵循固定间隔时间出发一名选手，不考虑带策略的间隔时间设计来进行同一条路线上的运动员的出发调度；
3） 不考虑地形、天气等因素对任务完成时间的影响；
4） 在第二、三问中，路段$\langle i,j\rangle$的长度仍旧由i,j两点的坐标决定，高程引起的路段长度变化忽略不计。

3 符号说明

集合：

N：表示为所有点的集合；

E_s：表示为第s条路线上边的集合，元素为$e_s=\langle i,j\rangle$；

K_s：表示为第s条路线上点的集合；

S：表示为线路的集合；元素为$s=\{K_s,E_s\}$；

变量：

m_s：表示为第s条路线上的参赛选手；

y_{ijs}：表示为路段$\langle i,j\rangle$是否被包含于路线s中，$y_{iis}\in\{0,1\}$，当$\langle i,j\rangle\in E_s$时，$y_{ijs}=1$，表示路段$\langle i,j\rangle$被包含于路线$s$中，否则$y_{ijs}=0$；

x_{is}：表示为点i是否被包含于路线s中，$x_{is}\in\{0,1\}$，当$i\in K_s$时，$x_{is}=1$，表示点i被包含于路线s中，否则$x_{is}=0$；

Δt_s：表示为第s条路线中，选手出发的间隔时间，单位为$\min$；

t_{ir}：表示为第r名运动员到达点i的时刻，单位为$\min$；

$x_{k_{|K_s|}}$：表示为第s条路线的终点，$k_{|K_s|}$表示为路线s的点集中的最后一个点，如果编号为$k_{|K_s|}$的点被选作第s路线的终点，则$x_{k_{|K_s|}}=1$，否则$x_{k_{|K_s|}}=0$；

$x_{k^s_{\text{interval1}}}$、$x_{k^s_{\text{interval2}}}$：表示为第$s$条路线的两个必经的检查点，$k^s_{\text{interval1}}$、$k^s_{\text{interval2}}$表示为第$s$条路线中两个必经检查点的编号，如果编号为$k^s_{\text{interval1}}$、$k^s_{\text{interval2}}$的点被选作第$s$条路线的两个必经检查点，则否则$x_{k^s_{\text{interval1}}}=1$、$x_{k^s_{\text{interval2}}}=1$，否则$x_{k^s_{\text{interval1}}}=0$、$x_{k^s_{\text{interval2}}}=0$；

$\bar{D}$：表示为比赛路线的平均长度，单位为m；

$\bar{P}$：表示为比赛路线的平均难度；

$\bar{H}$：表示为比赛路线的平均高程；

u_i：TSP 问题模型的辅助变量，为一个实数；

常数：

d_{ij}：表示为路段$\langle i,j \rangle$的路线长度，单位为m；

h_{ij}：表示为路段$\langle i,j \rangle$的高程差，单位为m；

P_i：表示为检查点i的打卡难度；

α：表示为路线长度均匀阈值；

β：表示为路线难度均匀阈值；

ρ：表示为路线重复阈值；

γ：表示为路线高程均匀阈值；

v：运动员的速度，单位为$m/\min$；

4 模型的建立、求解和分析

4.1 定向越野路线约束分析

定向越野的路线由一个起点、若干检查点和一个终点构成。根据路线设计规则 5），每条路线均存在 20 个以上的检查点，同时路线与路线之间拥有共同的起点、终点以及两个必经的打卡检查点。在基于选择检查点的路线设计过程中，起点、终点以及两个检查点需要单独作为决策变量。

1）路线长度均匀规则

根据要求 1），路线之间长度应尽可能均匀，通常可以使用方差来描述数据之间的离散程度。假设有S条路线，则S条路线的相对误差应当低于一个阈值，或者作为优化过程中的惩罚函数。根据附件 1，所有候选检查点的平面坐标已知，同时点与点之间可以直线连接，则可以根据检查点以及检查点之间的连接顺序计算整条路线的长度。

2）路线打卡难度均匀规则

根据要求 2），路线之间总打卡难度应尽可能均匀。与长度约束类似，所有路线的总打卡难度的相对误差应当低于一个阈值，或者作为优化过程中的惩罚函数。

3）路线高程均匀规则

根据要求 3），路线之间的爬高量应尽可能均匀。根据附件 2，所有候选检查点的高程数据已知，路线的爬高量定义为当后一个点高于前一个点时，后一个点与前一个点的高程差之和。与 1）和 2）类似，所有路线的爬高量的相对误差应当低于一个阈值，或者作为优化过程中的惩罚函数。

4）避免重复规则（防“跟跑”规则）

根据要求 4），不同路线之间重复路段不能过多，选手之间距离不能过近。首先是路线的重复路段，由于是所有路线彼此之间不能重复过多，因此不能对所有路线的总重复率进行约束，而需要针对每对路线之间的重复率进行约束。可以针对每对路线计算重复率，对最高的一组设置阈值，或者将其作为优化过程中的惩罚函数。

其次是选手之间的距离不能过近。选手与选手之间距离的计算存在两种情况，其一是该节点没有其他路线经过，则距离只与本路线中选手的速度、出发时间相关。其二是该节点有其他路线经过，则需要计算出所有经过该节点的选手的到达时间，并进行排序，判断是否存在违反约束的情况。

5）检查点选择规则

根据要求 5），每条路线的检查点数量应大于 20，同时终点的打卡难度应低于一定值。因此我们在难度 2 以下的点里选择终点。与终点类似，所有路线需要共同经过两个检查点。由于是作为补给水和医疗保障，因此在条件允许的情况下，应当选择难度较低同时相隔一定距离的点作为保障检查点。

6）检查点连续约束及最短路线约束

根据要求 6），存在以下两方面约束：其一，如果一条路线中经过的检查点已知，则根据打卡点顺序连续的要求，一个检查点的下一个检查点应当是距离该点较近的点；其二，制定打卡顺序时要使得总路线尽可能短。在这个问题中，第一项约束与第二项约束在某些情况下可能存在冲突，如图（1）

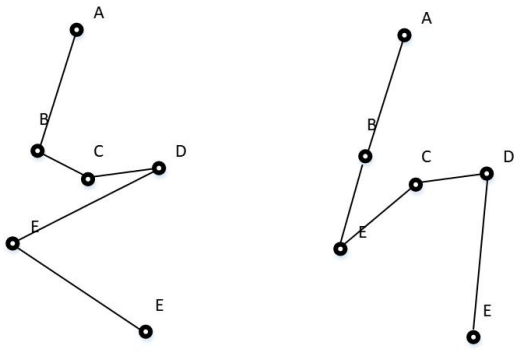


图 1 两种策略的连线方案

若按照最邻近节点相连的策略，每次都寻找当前节点最近的节点相连，则距离 B 最近的是 C，离 C 最近的是 D，以此类推，会得到左图的连接方案；若按照寻找最短路径的策略，显然会得到右图的连接方案。在两种方案中，按照最短路径的方法设计路线，虽然非每个节点都找最近的节点相连，但可以基本满足相邻节点尽可能相连。因此，在确定一条路线中的检查点后，我们选择寻找最短路径的策略，将其抽象为 TSP 问题求解。

7）同时到达约束

根据要求 7），同时到达一个检查点的参赛选手数量应控制在不超过 5 人。在问题一和问题二中，选手的行进速度均匀，且第一批次出发时间确定，因此可以根据间隔时间和路线计算出每名选手到达各个节点的时间。对这些到达时间进行排序，如果 5 名之后的人的到达时间，减去他的前面第五名的到达时间小于 1 分钟，则满足约束，否则违反约束；在问题三中，选手的行进速度不同，需要制定选手的出发排序策略，之后按照相同的方法计算到达时间，以及判断是否满足约束。

4.2 基于最大参与人数的路线设计问题 s

4.2.1 数学模型的构建

根据问题一的要求，我们将定向越野的路线设计问题抽象成一个具有多约束的目标优化问题。要使得本次比赛中的参与人数最多，实际上就是在满足以上约束的前提下，通过对总路线数、每条路线中的检查点、检查点的排列顺序、终点、每条路线的出发间隔时间进行选择，通过计算得出最大参与人数，并提供相应的参赛路线设计的方案。

目标函数

首先目标很明确，是使得本次比赛的参与人数最多。假设 m_s 表示为第 s 条比赛路线上的参赛人数，则目标函数表示为：

$$\max \sum_{s} m_s \tag{1}$$

约束条件

1）长度均匀规则

针对题目中路线设计规则（1）“不同路线的长度应尽可能均匀”，即最终得到 s 条路线中的任意条路线总长度与平均路线长度之间的差值应该控制在一定阈值，则本文设定路线长度均匀约束如式（2）所示：

$$\frac{\left|\sum_{\langle i,j\rangle \in E_s} d_{ij} y_{ijs} - \bar{D}\right|}{\bar{D}} \le \alpha, \forall s \in S \tag{2}$$

$$\bar{D} = \frac{\sum_{s}\sum_{\langle i,j\rangle \in E_s} d_{ij} y_{ijs}}{S} \tag{3}$$

式（2）中，E_s 表示为第 s 条路线中边的集合，其中元素 $e_s = \langle i, j\rangle$ 表示为第 s 条路线中存在从点 i 到点 j 的路段；$y_{iis} \in \{0,1\}$，当 y_{iis}=1时表示路段 $\langle i, j\rangle$ 包含在路线 E_s 中，否则 $y_{iis} = 0$。$\bar{D}$ 表示为路线平均长度，可由式（3）求得。在本文中，我们将长度均匀阈值设置为 $\alpha = 10\%$。

2）难度均匀规则

针对题目中路线要求（2）“不同路线的总打卡难度应尽可能均匀”，即最终得到 s 条路线中的任意两条路线总难度之间的差值应该控制在一定阈值，则本文设定路线难度均匀约束如式（4）所示：

$$\frac{\left|\sum_{i\in K_s} p_i x_{is} - \bar{P}\right|}{\bar{P}} \le \beta, \forall s, s' \in S, \tag{4}$$

$$\bar{P} = \frac{\sum_{s}\sum_{i\in K_s} p_i x_{is}}{S} \tag{5}$$

式（4）中，K_s 表示为第 s 条路线中点的集合；$x_{is} \in \{0,1\}$，当 $x_{is}=1$ 时，表示点 i 包含在路线 s 中，否则 $x_{is}=0$。$\overline{P}$ 表示为路线的平均难度，可由式（5）求得。在本文中，我们将阈值设置为 $\beta=10\%$。

3）避免重复规则

针对题目中路线设计规则（4）“不同路线之间应尽可能避免包含较多重复路段”，即最终得到的 s 条比赛路线中的任意两条路线应尽可能地减少相同的路段，使得相同路段的数量控制在一定的阈值，因此本文将避免重复规则转化为约束如式（6）所示：

$$\frac{\sum_{i\in N}\sum_{j\in N}\left(y_{ijs}-y_{ijs'}\right)}{\sum_{i\in N}\sum_{j\in N}y_{ijs}} \geq 1-\rho, \forall s,s' \in S \tag{6}$$

在式（6）中，如果 $\langle i,j\rangle \in E_s$ 并且 $\langle i,j\rangle \in E_{s'}$，则有 $y_{iis}-y_{iis'}=0$，表明在路线 s 与路线 s' 中存在重复路段 $\langle i,j\rangle$；如果 $\langle i,j\rangle \in E_s$ 但 $\langle i,j\rangle \notin E_{s'}$，则有 $y_{iis}-y_{iis'}=1$。在本文中，我们将阈值设置为 $\rho=20\%$。

4）检查点选择规则

针对题目中路线要求（5）中“每条路线中检查点的数量不低于 20 个”，本文将其转化为模型约束如式（7）所示：

$$\sum_{i\in N} x_{is} \geq 20, \forall s \in S \tag{7}$$

针对题目中路线设计规则（5）中“在这些检查点中应设置 2 个检查点是所有路线都必经的打卡点”，即每条路线中都包含两个公共点，因此，模型包含约束如式（8）、（9）所示：

$$x_{k^s_{\text{interval1}}} = x_{k^{s'}_{\text{interval1}}}, \forall s,s' \in S \tag{8}$$

$$x_{k^s_{\text{interval2}}} = x_{k^{s'}_{\text{interval2}}}, \forall s,s' \in S \tag{9}$$

针对题目中路线设计规则“每条路线的终点都相同”以及“终点设置在难度较低的检查点”，则模型因包含约束如式（10）、（11）所示：

$$x_{k_{|K_s|}} = x_{k_{|K_{s'}|}} \tag{10}$$

$$P_{k_{|K_s|}} \leq 2 \tag{11}$$

5）路段选择规则

针对题目中路线设计规则（6）中“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”，当路线 s 中的点都确定时，路线 s 的路段选择问题转换为点确定的 TSP 问题，根据 TSP 问题中路径选择的贪心原则，当从当前点选择下一点构成新的路段时，会倾向于选择能够构成尽可能短的路段的检查点，因此，“同一路线中距离相近的检查点的打卡顺序尽可能连续”相当于“制定检查点打卡顺序时要考虑路线的长度尽可能短”条件的附属特点，对于规则（6），本文设定模型各条路线检查点、路段约束如下所示：

$$y_{ijs} \in \arg\left\{ \min \sum_{\langle i,j \rangle \in E_s} d_{ij} y_{ijs} \right\}$$

$$s.t. \sum_{\substack{i=1 \\ i \neq j}}^{N} y_{ijs} = 1, \forall j \in K_s \quad (12.a)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{N} y_{ijs} = 1, \forall i \in K_s \quad (12.b)$$

$$u_{js} \geq u_{is} + |K_s|\left(y_{ijs} - 1\right) + 1 \quad (12.c)$$

$$u_{is}, u_{js} \geq 0, \ u_{is}, u_{js} \in R \quad (12.d)$$

对于式（10.a）、（10.b），表示第s条路线中的点集K_s中的每个点都被经过一次，对于式（10.c），是为了防止构造 TSP 路径时出现回路。如果检查点i和检查点j构成回路，则有：

$$y_{ijs} = 1, y_{jis} = 1$$

因此有：

$$u_{is} - u_{js} \leq -1, u_{js} - u_{is} \leq -1$$

从而有$0 \leq -2$，出现矛盾。同理，如果检查点i、j和k构成回路，则有：

$$y_{ijs} = 1, y_{jks} = 1, y_{kis} = 1$$

因此有：

$$u_{is} - u_{js} \leq -1, u_{js} - u_{ks} \leq -1, u_{ks} - u_{is} \leq -1$$

从而有$0 \leq -3$，出现矛盾。因此，式（12.c）有效保证了路线s不会出现回路。

6）“同时到达”人数限制规则

针对题目中路线设计规则（7）中“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”，在题目中，“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。假设路线s中的点、路段以及每条路线的间隔时间都确定，则可以计算出经过每个路线点的运动员到达时间，假设t_{ir}表示为第r名运动员到达检查点i的时刻，则有：

$$t_{ir} - t_{i(r+5)} \leq 1, \forall i \in N \quad (13)$$

式（13）表示第r名运动员到达检查点i的时刻与第$r+5$名运动员到达检查点i的时刻的差值小于1分钟，这样就能够有效地保证同时到达同一个检查点的参赛选手数量不超过5个人。t_{ir}具体的计算流程将在下一节算法应用和问题求解

中进行详细描述。

7）比赛完成时间规则

针对问题一要求的“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小时”，则对于第s条路线上的第w_s名运动员，有：

$$120 \le t_{路途} + t_{逗留} \tag{14}$$

$$t_{路途} + t_{等待} + t_{逗留} \le 180 \tag{15}$$

式（14）表示每名运动员完成任务的时间都不低于2个小时（120分钟），式（15）表示对于每条比赛路线s，竞赛的总完成时间不超过3个小时（180分钟）。其中，$t_{路途}$表示为运动员从起点到达终点在路段上运动的时间，$t_{等待}$表示为运动员在起点等待出发的时间，$t_{逗留}$表示为运动员受到检查点难度的原因在检查点停留的时间。$t_{路途}$、$t_{等待}$、$t_{逗留}$可由下式求得：

$$t_{路途} = \frac{\sum_{i \in N}\sum_{j \in N} d_{ij} y_{ijs}}{v} \tag{16}$$

$$t_{等待} = \Delta t_s \cdot (m_s - 1) \tag{17}$$

$$t_{逗留} = \sum_{i \in K_s} p_i x_{is} \tag{18}$$

则式（16）、（17）可转换为式（19）、（20）如下所示：

$$120 \le \frac{\sum_{i \in N}\sum_{j \in N} d_{ij} y_{ijs}}{v} + \sum_{i \in K_s} p_i x_{is}, \forall s \in S \tag{19}$$

$$\frac{\sum_{i \in N}\sum_{j \in N} d_{ij} y_{ijs}}{v} + \Delta t_s \cdot (m_s - 1) + \sum_{i \in K_s} p_i x_{is} \le 180, \forall s \in S \tag{20}$$

4.2.2 算法应用和问题求解

根据问题模型的特点，我们运用嵌套的遗传算法求解该多约束目标优化问题。

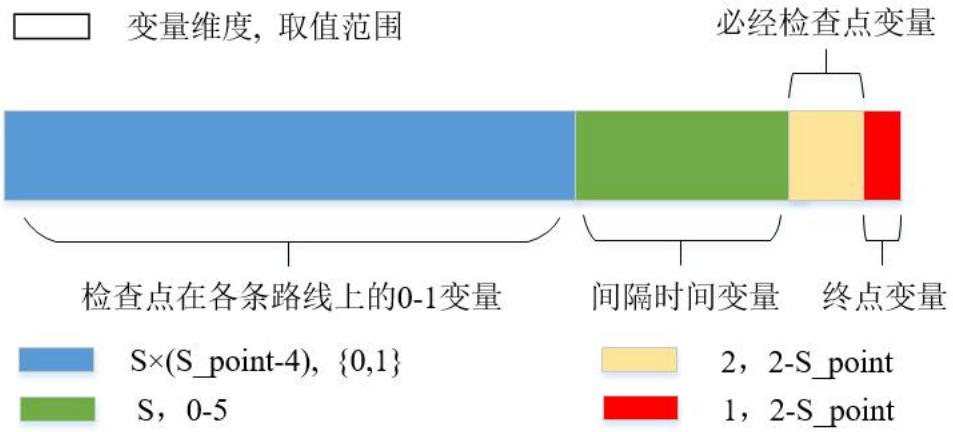


图 2 决策变量

1）总路线数 S

我们采用从少到多的路径数的策略逐一进行实验测试，因此在优化求解过程中，总路线数固定为 S 。 S 的取值在 3 到 8 之间较为合理，因此分别从 $S=3\ldots8$ 进行实验测试。

2）检查点变量 x_{is}

第二类变量是 $S\times\left(S_{\text{point}}-4\right)$ 维的 0-1 变量，表示每个条路线上是否存在相应的检查点。 S 是总的路线数， S_point 是总的检查点数，由于有起点、终点以及两个必经检查点需要另外考虑，因此总共有 $S_\text{point}-4$ 个检查点在 S 条路线上进行决策。每条路线中的检查点数量待定，同时检查点的选择也待定。因此我们定义 0-1 变量 $x_{is}\left(i=1\ldots100,s=1\ldots S\right)$，在路线 s 中，若候选检查点 i 属于路线 s，则 $x_{is}=1$，否则为 0。该变量可以同时表示路线中的检查点的数量和检查点的选择。

3）间隔时间 Δt_s

第三类变量是 S 维的变量，取值范围为 0 到 5，表示每条路线的出发间隔时间取值在 0 分钟到 5 分钟之间。0 到 5 是在经过测试后，我们认为较为合理的取值范围。

4）必经检查点和终点

第四类变量是 2 维变量，取值范围为 2 到 S_point，表示两个必经检查点的选择。第五类变量是 1 维的变量，表示终点的选择。这两类变量均可以在所有检查点中进行选择，因此取值范围为 2 到 S_point 。

根据上述决策变量，结合约束条件，就可以确定一个方案中的最大参与人数。总路线数 S 已知，在每条路线中所含的检查点和终点已知的情况下，结合要求 6）和 TSP 问题的求解方法，可以确定每条路线中的检查点顺序。在满足约束条件的情况下，每条路线已知，出发间隔时间已知，则最大参与人数可得。

4.2.2.2 基于锦标赛选择法的精英个体选择策略

遗传算法中，锦标赛选择法是每次从种群中取出一定数量个体（放回抽样），然后选择其中最好的一个进入子代种群。重复该操作，直到新的种群规模达到原来的种群规模。对个体两两进行比较适应度和违约值，选择策略如下：

情况一：违约值均为 0，选择适应度值高的个体。

情况二：违约值一个为 0，一个大于 0，选择违约值等于 0 的个体。

情况三：违约值均大于 0，选择违约值较小的个体。

4.2.2.3 基于 TSP 问题的检查点排序求解

根据要求 6），制定检查点顺序时要使得总路线长度尽可能短。若终点和检查点已知，则对检查点的排序可以当作一个 TSP 问题：即从一个点出发，经过 n 个点，最后到达终点。每个点只拜访一次，求其中最短的路径。为了避免陷入局部最优解，采用邻域法构造初始解。

在该方法中,从某个城市出发其下一站不是其最近城市,而在比最近城市稍远的邻域范围进行随机选取。邻域法既能提取局部优化路径特征信息,又具有多样性。

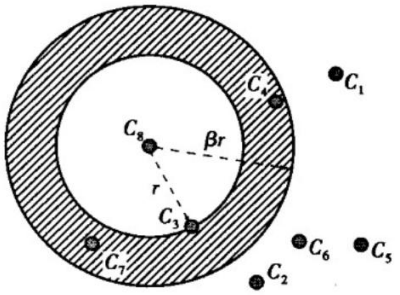


图 3 邻域法

图中以C_8为起点，距离最近的是C_3，距离为 r。以C_8为圆心，r 为内圆，βr 为外圆，外圆和内圆间的环形区域称为C_8的邻域。为了避免最近邻法陷入局部最优的缺陷，下一个点将在邻域内随机选择。之后以此类推，直到所有点都去过后完成搜索。将这一搜索的路径作为初始解带入遗传算法计算最短路径，作为这一个体的路径。流程如下：

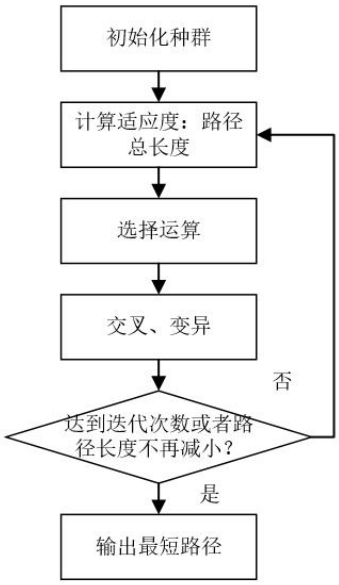


图 4 遗传算法流程

4.2.2.4 选手到达节点的时刻计算

在问题一中，每名选手的行进速度均为 6km/h。假设早上 8 点为 0 时刻，则在路线 s 中，选手 r 到达节点 k 的时刻为

$$t_{kr}=\Delta t_s\cdot\left(r-1\right)+\frac{\sum_{\langle i,j\rangle\in E_s}^{j=k}d_{ij}y_{ijs}}{v} \tag{21}$$

对所有运动员到达 k 点的时刻由小到大进行排序，得到时间序列 $\{t_{kp}\}$，约束为当到达 k 点的人数超过 5 时，第 5 人往后，所有人的到达时间减去前面第 5 个人的到达时间应小于 1 分钟，即

$$\left|t_{k(p+5)}-t_{kp}\right|\le 1,\forall t_{kp},t_{k(p+5)}\in t_{kp} \tag{22}$$

式（22）中，t_{kp}、$t_{k(p+5)}$ 分别表示为第 p 个、第 $p+1$ 个人到达检查点 k。

4.2.2.5 算法流程

本文采用遗传算法对上述建立的数学模型问题进行求解。由于约束中涉及对 TSP 问题进行求解，得到最短路径后才能计算和判断是否满足约束，因此我们嵌套使用遗传算法对 TSP 问题进行求解。整体算法的流程如图（5）所示：

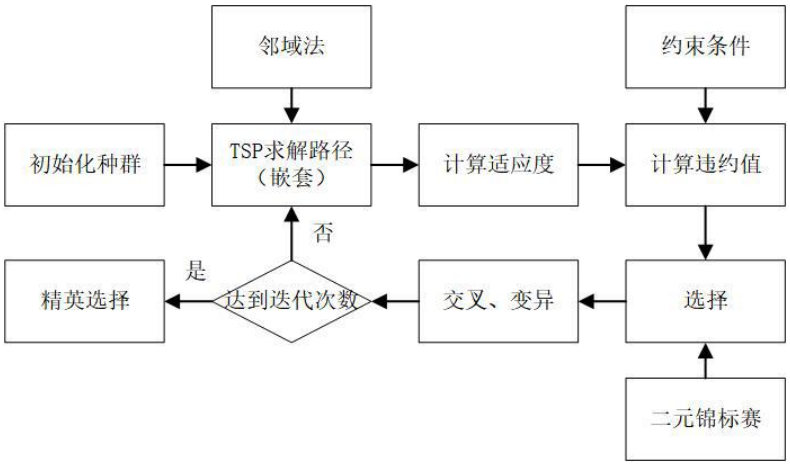


图 5 算法流程图

步骤一：初始化种群

步骤二：适应度计算

步骤三：基于邻域法的 Tsp 问题求解及违约值计算

根据 4.2.3 所述方法求得该种群下的路线方案，包括路线中的检查点及其顺序。根据所得的路线和间隔时间变量，计算每个人到每个点的到达时间，并判断以下约束：

1）检查点总数超过 20 个点

2）单个人的最短路径完成时间大于 2h

3）最后一个人的完成时间小于 3h

4）计算每个节点经过的人的到达时间，每分钟到达人数不大于 5 人

5）路径与路径之间的相似程度不大于 10%

6）路径与路径之间难度相差不大于 10%

7）路径与路径之间爬高量不大于 10%（在进行第二问的求解时才使用该约束）

每违反一条，违约值加 1。

步骤四：二元锦标赛选择精英个体

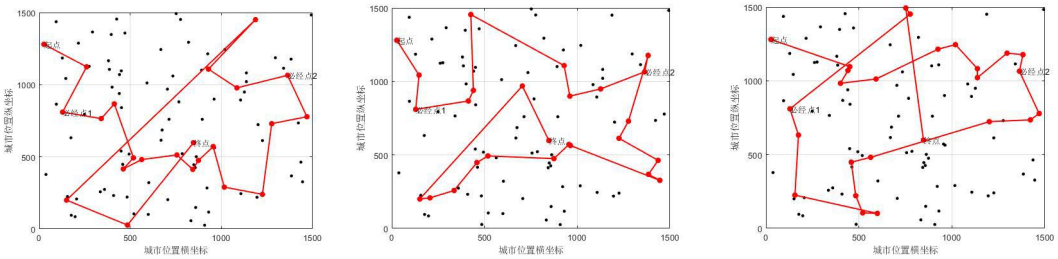
步骤五：交叉变异

对于新构造的种群中的个体，进行 9%的均匀交叉，并进行 10%的变异。

步骤六：达到迭代次数，得到参与人数最多的结果。

4.2.4 结果分析

1）总路线数 S=3 时



（1）　　（2）　　（3）

图 6　1-3 路线结果

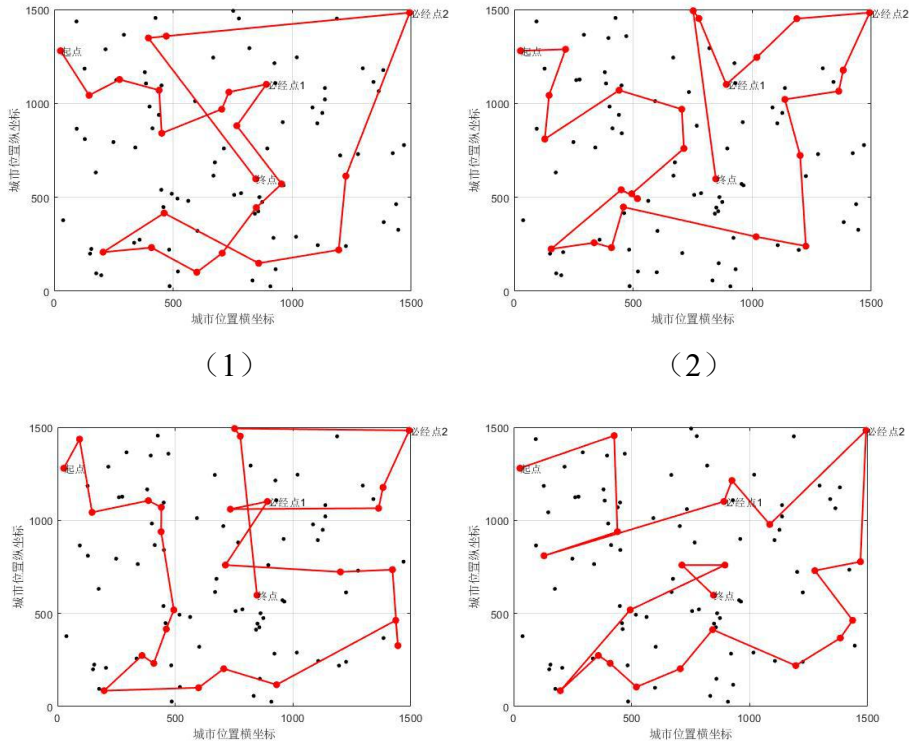
当总路线数为 3 时，并且从图 6 可以看出，图（3）路线比较曲折，来回绕路的情况比较多，说明该路线在对约束 6）的满足程度较低；图（2）整体效果较好，三条路线的长度差异、重复率、节点数、爬高量差异等统计如下：

表 1 1-3 统计结果

N	终点	必经检查点 1		必经检查点 2		参加最大人数	
3	39	50		82		204	
路线	节点数	长度	长度差异	难度	难度差异	最大重复率	完成时间
1	23	7190.30	5.10%	50	8.54%	0.00%	120.90
2	26	6651.73	2.77%	57	4.27%	0.00%	122.51
3	26	6681.62	2.33%	57	4.27%	0.00%	122.81

总体长度差异保持在 6%以下，难度差异保持在 10%以下，较为均匀，整体路线设计较为公平。路线重复率均为 0，可以有效减少参赛选手相遇的情况。最大参与人数为 204 人。

2）总路线数 S=4 时



（1）　　（2）

（3）　　（4）

图 7　1-4 路线方案

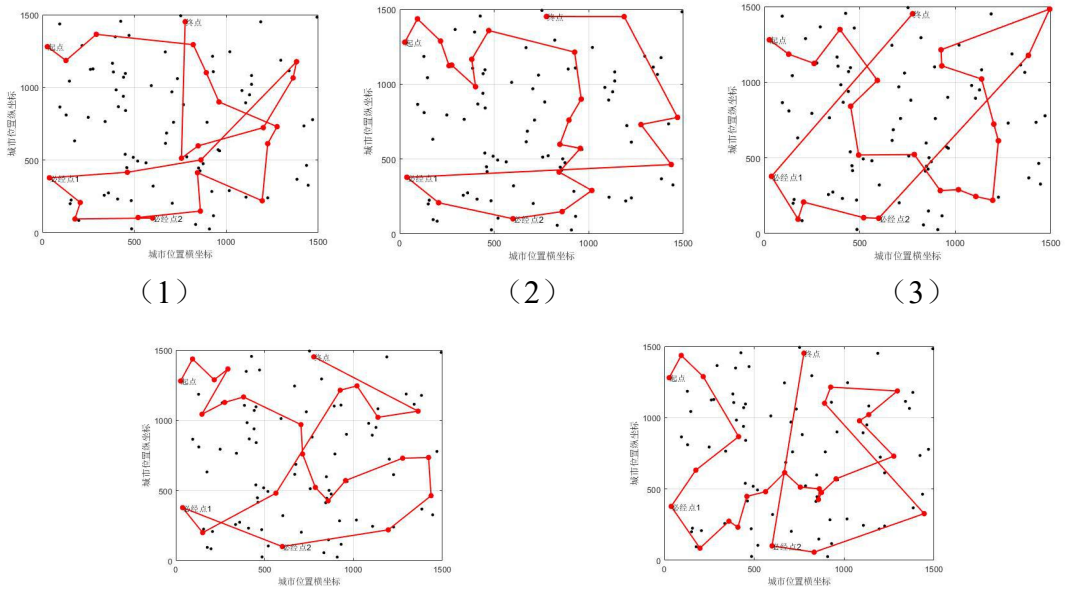
当总路线为 4 时，图中四条路线的曲折程度较低，绕路的情况不多。四条路线的长度差异、重复率、节点数、爬高量差异等统计如下：

表 2　1-4 统计结果

N	终点	必经检查点 1		必经检查点 2		参加最大人数	
4	39	59		100		151	
路线	节点数	长度	长度差异	难度	难度差异	最大重复率	完成时间
1	23	7444.74	1.15%	50	9.91%	0.00%	124.44
2	27	7265.88	3.53%	61	9.91%	7.69%	133.65
3	27	7964.93	5.75%	61	9.91%	8.33%	140.64
4	23	7450.58	1.07%	50	9.91%	8.33%	124.50

总体长度差异保持在 6%以下，难度差异偏大，但仍保持在 10%以下。路线重复率较高，最大参与人数为 151 人。各方面指标都低于 S=3 的情况。

3）总路线数 S=5 时



（1）　（2）　（3）

（4）　（5）

图 8　1-5 路线方案

表 3　1-5 统计结果

N	终点	必经检查点 1		必经检查点 2		参加最大人数	
5	18	14		16		116	
路线	节点数	长度	长度差异	难度	难度差异	最大重复率	完成时间
1	24	7696.62	2.07%	53	5.02%	4.35%	129.96
2	24	6939.40	7.97%	53	5.02%	8.51%	122.39
3	25	7906.98	4.86%	56	0.36%	4.26%	135.06
4	25	7386.15	2.04%	56	0.36%	12.00%	129.86
5	27	7771.89	3.07%	61	9.32%	12.00%	138.71

当总路线数为 5 时，最大参与人数为 116 人。并且长度差异、难度差异和重复率指标均劣于 S=3 的情况。

我们还进行了 S=6，7，8 的实验，后三者的约束满足情况较差。综上，当 S=3 时，可以得到对约束条件符合程度高、最大参与人数较大的方案。最大参与人数为 204，路线中的检查点及其顺序在附件中给出。

不足之处及原因分析

图中存在不少交叉线路的情况，与 TSP 问题的求解特征相矛盾。原因在于设计算法时受限于时间要求和设备条件，设置的迭代次数相对较少，没能每次都找到 TSP 问题的最优解。因此可能出现路线并非最短路径的情况。

4.3 考虑高程的路线设计问题

4.3.1 数学模型的构建

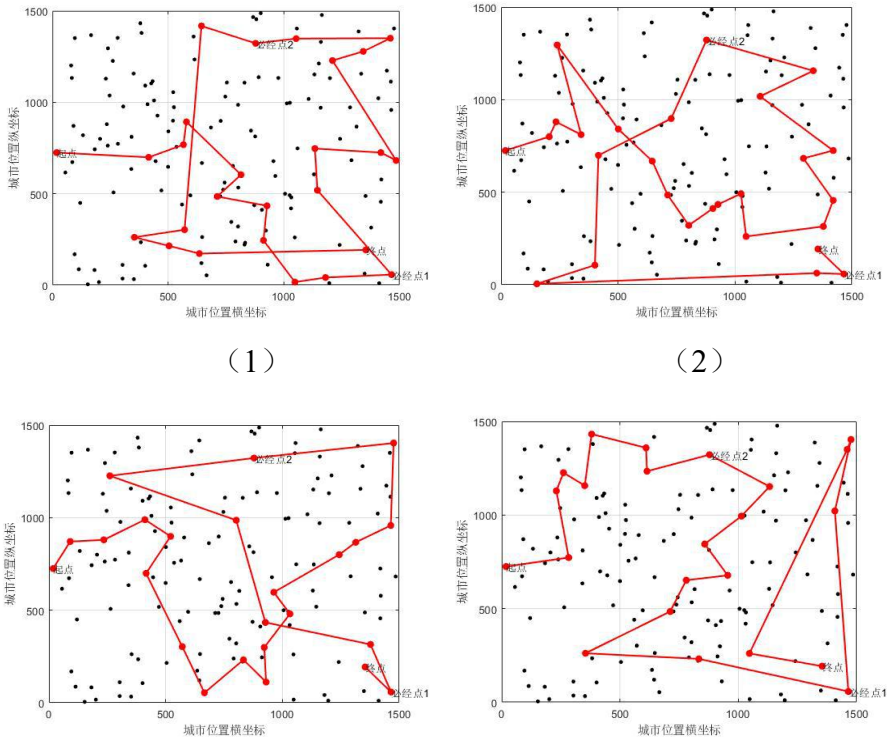
在问题一的基础上，加入爬高量约束，即在计算违约值时将“路径与路径之间爬高量不大于 10%”纳入违约范围。其余的目标函数、决策变量以及约束保持不变。则针对此路线设计的高程均匀规则，模型中需添加高程差约束如下式所示：

$$\frac{\left|\sum_{\langle i,j\rangle\in E_s} h_{ij}y_{ijs}-\bar{H}\right|}{\bar{H}}\le\gamma,\forall s,s'\in S \tag{23}$$

即在路线集合 S 中，任意两条路线的总高程差的差值控制在比例 γ 内。其中 $\bar{H}$ 表示为路线的平均高程，可由式（23）求得。

4.3.2 结果分析

本处的结果过程与第一问相同，下面给出针对第二问的最优解及其约束指标：



（1）　（2）

（3）　（4）

图 9　2-4 路线方案

表 4　2-4 统计结果

N	终点	必经检查点 1		必经检查点 2		参加最大人数	
4	19	106		16		198	
路线	节点数	长度	长度差异	难度	难度差异	爬高量	爬高差异
1	26	7548.58	1.75%	60	3.90%	41	5.13%
2	27	7777.83	4.84%	63	9.09%	41	5.13%
3	24	6869.16	7.40%	55	4.76%	38	2.56%
4	23	7478.42	0.81%	53	8.23%	36	7.69%
路线	最大重复率		完成时间				
1	0.00%		135.48				
2	4.08%		140.77				
3	4.08%		123.69				

当 N=4 时，最大参与人数为 198，长度差异保持在 10%以下，难度差异保持在 10%以下，爬高差异保持在 8%以下，最大重复率保持在 5%以下。约束条件满足情况较好，并且从路线图上看，路线总体连贯，没有临近点断开的情况。

4.4 考虑速度分组的路线设计问题

在问题二的基础上，由于新增加了 120 名参赛选手的速度数据，在越野路线方案不变的情况下，每名选手选择的路径会影响比赛完成的时间。同时，不同的出发顺序也会影响计算的结果。

4.4.1 选手出发顺序策略

在路径 s 上有 m 名参赛选手，他们的速度依次为$\{v_r,\mathrm{i}=1\cdots m\}$，该路径的出发时间间隔为$\Delta t_s$。若选手 r 的出发次序为n_j，则该路径的完成时间为

$$T_s=\max\left(\left(n_j-1\right)\cdot\Delta t_s+\frac{S_s}{v_r}\right),(r\in m) \tag{24}$$

所有路径的总完成时间为

$$T=\max T_s,(s\in S) \tag{25}$$

因此但所有路径的完成时间都尽可能小时，总完成时间也会接近最小值。在式（25）中，应尽量避免某位选手的出发时间与完成路径时间同时偏大的情况，减少极值出现的概率。因此，在同一条路线中，速度越快的选手出发次序越大，速度越慢的选手出发次序越小。在一条路线上的选手，按照速度大小从高到低排序，对应的出发次序从低到高排序。在降低总完成时间的同时，也方便计算每名选手到达各个节点的时间，判断是否满足相应的约束。

4.4.2 数学模型的构建

针对问题三，问题的中心在于如何设计路线以及运动员的分配使得在尽可能少的路线上容纳 120 名运动员，并且每条路线上的比赛完成时间尽可能短。则结合问题一设定的路线设计规则，问题三的模型如下所示：

$$\min \max_{s}\left(\frac{\sum_{i\in N}\sum_{j\in N} d_{ij}y_{ijs}}{v_r}+\Delta t_s\left(w_s-1\right)+\sum_{i\in K_s} p_i x_{is}\right) \tag{27}$$

$$s.t.\ (2)\text{-}(13),(19),(20)$$

同时，根据题目要求，由于每名运动员的速度都不相同，因此，任务完成时间约束转变为：

$$120 \le \frac{\sum_{i\in N}\sum_{j\in N} d_{ij}y_{ijs}}{v_{r_s}}+\sum_{i\in K_s} p_i x_{is}, \forall s\in S, r_s \le m_s \tag{28}$$

$$\frac{\sum_{i\in N}\sum_{j\in N} d_{ij}y_{ijs}}{v_{m_s}}+\Delta t_s\left(m_s-1\right)+\sum_{i\in K_s} p_i x_{is} \le 210, \forall s\in S \tag{29}$$

式（18）表示，路线 s 上的第 r_s 名运动员完成任务的时间不超过 2 个小时，同时，对于任意路线 s 上的最后一名运动员，即第 m_s 名运动员完成任务时间加上等待时间不超过 3.5 个小时，因此能够保证比赛总完成时间不超过 3.5 个小时。

4.4.3 模型求解

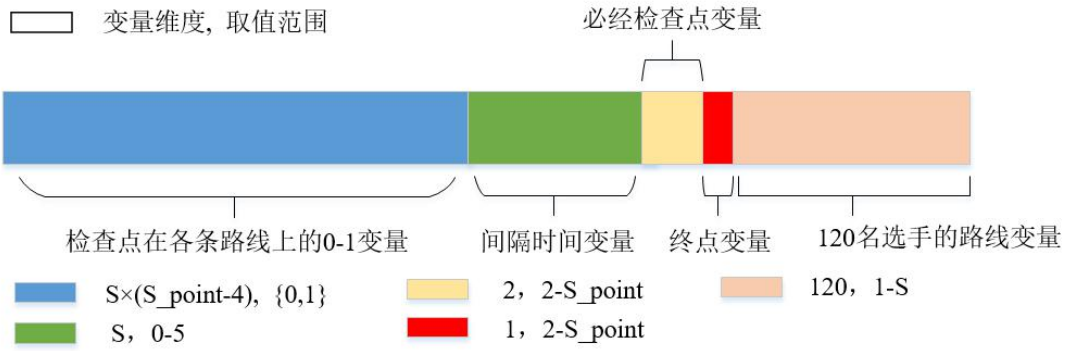


图 10 决策变量

在问题一决策变量的基础上，新增加 120 名选手选择的路线变量。这部分变量一共有 120 维，取值范围为（1，S），S 是总的路线数。

在计算约束 6）时，由于每个人的速度不同，则在路线 s 中，选手 r 到达节点 k 的时刻为

$$t_{kr}=\Delta t_s\cdot\left(r-1\right)+\frac{\sum_{\langle i,j\rangle\in E_s}^{j=k} d_{ij}y_{ijs}}{v_r} \tag{30}$$

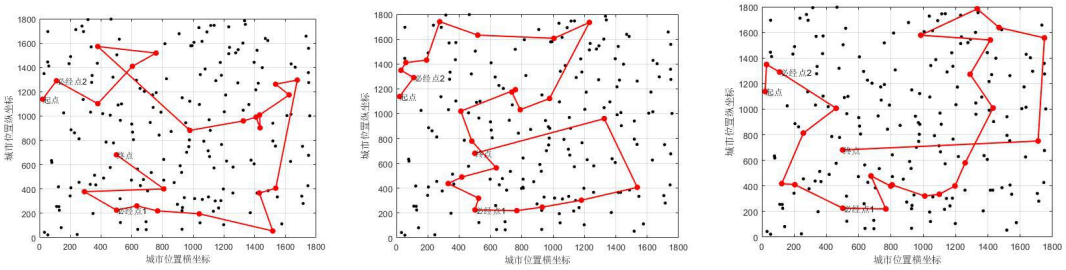
对所有运动员到达 k 点的时刻由小到大进行排序，得到时间序列 $\{t_{kp}\}$，约束为当到达 k 点的人数超过 5 时，第 5 人往后，所有人的到达时间减去前面第 5 个人的到达时间应小于 1 分钟，即

$$\left|t_{k(p+5)}-t_{kp}\right|\le 1,\forall t_{kp},t_{k(p+5)}\in t_{kp} \tag{31}$$

式（22）中，t_{kp}、$t_{k(p+5)}$ 分别表示为第 p 个、第 $p+1$ 个人到达检查点 k。

4.4.3 结果分析

我们进行了 S 从 3 到 8 的测试和实验，通过综合分析违约情况、路线数以及比赛完成时间等指标，得到 S=3 时为最优方案。S 取 3 时，最优路线方案如下所示：



（1）　　（2）　　（3）

图 11　3-3 路线方案图

如图（1）到图（3）所示，路线的整体趋势平滑，除了图（3）的个别路段，基本没有绕路的情况，相邻节点连续性较好。

表 5　3-3 统计结果

N	终点	必经检查点 1		必经检查点 2		完成时间（min）	
3	19	190		37		118	
路线	节点数	长度	长度差异	难度	难度差异	爬高量	爬高差异
1	24	7114.81	1.55%	90	3.57%	67	0.00%
2	26	6949.63	3.83%	96	2.86%	67	0.00%
3	25	7615.73	5.38%	94	0.71%	67	0.00%
路线	最大重复率		完成时间	分配人数			
1	4.17%		135.48	31			
2	4.17%		140.77	42			
3	4.08%		123.69	47			

S=3 时得到的最优方案中，长度差异、难度差异和爬高差异指标都很低，同时完成时间是 118 分钟，符合小于 3.5 小时的要求，也是所有方案中的最短时间。各条路线上分配的人数相对均匀，重复路段少。

5 优缺点分析

5.1 优点

（1）使用了遗传算法，减少了计算量，避免了经典优化方法的复杂计算；
（2）对变量进行了合理编码，有效的简化了模型，提高了运算效率；
（3）考虑了多种路径情况，较全面的实现了全局最优解的寻找。

5.2 缺点

（1）嵌套模型计算成本过高，导致里层 TSP 问题计算规模较小，得到的路径解可能不是最优；
（2）在用数学语言表达约束条件时，部分约束条件没有实现，导致搜索空间人为扩大；
（3）算法类型较为单一，没有考虑其他智能算法是否优于遗传算法。

参考文献

[1]王吉权,程志文,张攀利,代伟婷.求解有约束优化问题的实数遗传算法改进研究[J].控制与决策,2019,34(05):937-946.
[2]牛向阳,高成修.基于混合编码的混合遗传算法[J].数学杂志,2008(04):399-403.
[3]李志宾,侯世旺,程厚虎.一种求解 TSP 初始化种群问题的方法[J].计算机工程与应用,2016,52(17):172-176.
[4]王书勤,黄茜.军事定向越野路径优化问题建模及混合蚁群算法求解[J].运筹与管理,2018,27(04):105-111.

附 录

一、主函数

main.m

```
clc;
clear all;
global n_point;
n_point = 100; %点的个数
Gm = 200;%迭代次数
Np = 20; %种群数
Pc = 0.9; %交叉概率
Pm = 0.1;%变异概率
n = 3; % 路径数
x = ini(Np,n);
for j = 1 : Np;
    [fit(j), tc(j)] = fitness(x(j,:),n);
end;

for i = 1:Gm;
    %select
    x1 = select(x,fit,tc);
    %crossover
    x2 = crossover(x1,Pc);
    %mutation
    x3 = mutate(x2,Pm,n);

    %精英选择
    for j = 1 : Np;
        [fit1(j),tc1(j)] = fitness(x3(j,:),n);
    end;
    fit2 = [fit, fit1];
    tc2 = [tc, tc1];
    x4 = [x; x3];
    list1 = find(tc2 == 0);
    if size(list1,2) >= Np;
        [fit3, index] = sort(fit2(list1),'descend');
        fit = fit2(list1(index(1:Np)));
        tc = tc2(list1(index(1:Np)));
        x = x4(list1(index(1:Np)),:);
    else
        n1 = Np - size(list1,2);
        list2 = find(tc2 ~= 0);
        [tc3, index] = sort(tc2(list2));
```

```
        tc = [tc2(list1), tc2(list2(index(1:n1)))];
        fit = [fit2(list1),fit2(list2(index(1:n1)))];
        x = x4([list1,list2(index(1:n1))],:);
    end;
    if isempty(find(tc == 0));
        max_fit(i) = 0;
        [min_tc(i),index] = min(tc);
        opt_x(i,:) = x(index,:);
    else
        list3 = find(tc == 0);
        [max_fit(i),index] = max(fit(list3));
        min_tc(i) = 0;
        opt_x(i,:) = x(list3(index),:);
    end;
    i
    min_tc(i)
    max_fit(i)
end;

[aa,bb,cc,dd] = fitness(opt_x(Gm,:),n);
%画出代数跟最优函数值之间的关系图
figure(1)
subplot(2,1,1)
plot(1:Gm,max_fit,'b');
subplot(2,1,2)
plot (1:Gm,min_tc,'k');

plot_x(opt_x(Gm,:),n)
```

二、适应值与违约值计算函数

fitness.m

```
function [fit, tc, yy, TT] = fitness(x1,n)
    load('initial.mat')
    global n_point;
    %检查点01向量x
    end_p = x1((n_point - 3)*n+1);
    c_p1 = x1((n_point - 3)*n+2);
    c_p2 = x1((n_point - 3)*n+3);
    v = 100; %速度,单位m/min
    x_k = [1,end_p,c_p1,c_p2];%起点终点必经检查点
    x_num = 1:n_point;
    x_num(x_k) = [];
    for i = 1 : n;
        x(i,:) = x1((n_point - 4)*i-(n_point - 5):(n_point - 4)*i);
```

```
    end;
    delta_t = x1((n_point - 4)*n+1:(n_point - 4)*n+n);
    num = [];

    %路径y
    c(1:4) = 0;
    for k = 1:n;
        y = [];
        check_p = [x_num(find(x(k,:))),c_p1,c_p2];%除起点终点外所有检
查点
%         check_p1 = check_p;
%         y(1) = 1;
%         for i = 2 : size(check_p,2)+1;
%             [~,index] = min(d(y(i-1),check_p1));
%             y(i) = check_p1(index);
%             check_p1(index) = [];
%         end;
%         y(size(check_p,2)+2) = end_p;

        y = ga_tsp(check_p,end_p);
        %约束惩罚值
        dc(1) = 16 - sum(x(k,:));
        dc(2) = 0;
        for i = 3:size(y,2);
            [~,index] = min(d(y(i),y(1:i-1)));
            if index ~= i-1;
                dc(2) =dc(2) + 1;
            end;
        end;
        T = [];
        T(1) = 0;
        total_d(k) = 0;%总长度
        total_h(k) = 0;%总难度
        %total_hh(k) = 0;%总高程
        for i = 2:size(y,2);
           T(i) = T(i-1) + t(i-1) + d(y(i-1),y(i))/v;
           total_d(k) = total_d(k) + d(y(i-1),y(i));
           total_h(k) = total_h(k) + t(i-1);
%             if h(i) > h(i-1);
%                 total_hh(k) = total_hh(k) + (h(i) - h(i-1));
%             end;
        end;
        total_h(k) = total_h(k) + t(end_p);
        end_T = T(size(y,2)) + t(end_p);
```

```
        dc(2) = 0;%去掉约束2
        dc(3) = 2*60 - end_T;
        dc(4) = end_T - 3*60;
        num(k) = floor((3*60 - end_T)/delta_t(k)) + 1;
        if num(k) <= 0;
            num(k) = 0;
        end;
        dc(find(dc<0)) = 0;
        c(1:4) = c(1:4) + dc(1:4);
        yy{k} = y;
        TT{k} = T;
    end;
    c(5) = 0;
    for i = 2:n_point;
        if i ~= end_p;
            point_t = [];
            j = 1;
            for k = 1:n;
                if (~isempty((find(yy{k} == i))));
                    point_t(j:(j+num(k)-1)) = TT{k}(find(yy{k} ==
i))+(0:(num(k)-1))*delta_t(k);
                    j = j+num(k);
                end;
            end;
            point_t = sort(point_t);
            if size(point_t,2) >= 5;
                for m = 5 : size(point_t,2);
                    if (point_t(m) - point_t(m-4)) <= 1;
                        dc(5) = 1 - point_t(m) + point_t(m-4);
                        if dc(5) < 0;
                            dc(5) = 0;
                        end;
                        c(5) = c(5) + dc(5);
                    end;
                end;
            end;
        end;
    end;
    total_d_a = mean(total_d);
    total_h_a = mean(total_h);
%    total_hh_a = mean(total_hh);
    c(6) = sum((abs(total_d - total_d_a)/total_d_a)>0.1) +
sum((abs(total_h - total_h_a)/total_h_a)>0.1) ;%+sum((abs(total_hh -
total_hh_a)/total_hh_a)>0.1);
```

```
    c(7) = 0;
    for k = 1:n;
        for m = (k+1):n;
            n1 = size(yy{k},2);
            n2 = size(yy{m},2);
            lujing1 = [(yy{k}(1:n1-1))',(yy{k}(2:n1))'];
            lujing2 = [(yy{m}(1:n2-1))',(yy{m}(2:n2))'];
            lujing = [lujing1;lujing2];
            lujing3 = unique(lujing,'rows');
            n3 = size(lujing3,1);
            if (n1+n2-2-n3)*2/(n1+n2-2) > 0.2;
                c(7) = c(7) + 1;
            end;
        end;
    end;
    if t(end_p) > 2;
        c(8) = 1;
    end;
    fit = sum(num);
    tc = sum(c);

end
```

三、选择、交叉、变异函数

select.m

```
function x1 = select(x,fit,tc)
    %锦标赛选择法
    n = size(x,1);
    for i = 1 : n;
        x2 = randi(n,[1,2]);
        if tc(x2(1)) == 0 && tc(x2(2)) == 0;
           [a, b] = max(fit(x2));
           x1(i,:) = x(x2(b),:);
        elseif tc(x2(1)) > 0 && tc(x2(2)) > 0;
            [a, b] = min(tc(x2));
            x1(i,:) = x(x2(b),:);
        else
            a = x2(find(tc(x2) == 0));
            x1(i,:) = x(a,:);
        end;
    end;
end
```

crossover.m

```
function x2 = crossover(x1,Pc)
    n = size(x1,1);
    index = randperm(n);
    x1 = x1(index,:);
    x2 = x1;
    for i = 1 :2: n-1;
        r = rand();
        if r < Pc
            for j = 1 : size(x1,2)-3;
               p = rand();
               if p < 0.5;
                   x2(i,j) = x1(i+1,j);
                   x2(i+1,j) = x1(i,j);
               end;
            end;
            x2(i,size(x1,2)-2:size(x1,2)) =
x1(i+1,size(x1,2)-2:size(x1,2));
            x2(i+1,size(x1,2)-2:size(x1,2)) =
x1(i,size(x1,2)-2:size(x1,2));
        end;
    end;
end
```

mutate.m

```
function x3 = mutate(x2,Pm,n)
    [m,k] = size(x2);
    global n_point;
    p = rand(m,k);
    x3 = x2;
    for i = 1 : m;
        for j = 1 : n*(n_point - 4);
            if p(i,j) < Pm;
                x3(i,j) = 1 - x3(i,j);
            end;
        end;
        for j = n*(n_point - 4)+1 : n*(n_point - 4)+n;
            if p(i,j) < Pm;
                x3(i,j) = rand()*5;
            end;
        end;
        if rand()<5*Pm;
            x3(i,(n_point - 3)*n+1:(n_point - 3)*n+3) =
randperm((n_point - 1),3)+1;
        end;
```

```
        n1 = sum(x3(i,1:n*(n_point - 4)));
        if n1 > 21*n;
            r1 = randperm(n1);
            r2 = r1(1:(n1-21*n));
            dp = find(x3(i,1:n*(n_point - 4)) == 1);
            x3(i,dp(r2)) = 0;
        end;
    end;
end
```

三、TSP 计算最短路径函数

ga_tsp.m

```
function y1 = ga_tsp(check_p,end_p)
    load('initial.mat');
    city_num = size(check_p,2);
    maxGEN = 200;
    popSize = 10; % 遗传算法种群大小
    crossoverProbabilty = 0.9; %交叉概率
    mutationProbabilty = 0.1; %变异概率

    %初始化种群
    for i = 1:popSize;
        check_p1 = check_p;
        y = [];
        y(1) = 1;
        for j = 2 : (city_num+1);
            [~,index] = sort(d(y(j-1),check_p1));
            if size(check_p1,2) >= 3 && i ~= 1;
                bb= index(randi(3));
            else
                bb = index(1);
            end;
            y(j) = check_p1(bb);
            check_p1(bb) = [];
        end;
        y(city_num+2) = end_p;
        pop(i,:) = y(2:city_num+1);
    end;

    fit = fitness1(pop,end_p,d);

    for i = 1 : maxGEN;
        %选择
        v_sum = sum(1./fit);
```

```
cc = (1./fit)./v_sum;
pc = cumsum(cc,2);
for j = 1:popSize;
    r = rand();
    tar = find(pc>=r);
    pop1(j,:) = pop(tar(1),:);
end;
%交叉
index = randperm(popSize);
pop1 = pop1(index,:);
for j = 1:2:(popSize-1);
    a = pop1(j,:);
    b = pop1(j+1,:);
    r = rand();
    if r < crossoverProbabilty;
        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 ii=s:e;
                a1=a;b1=b;
                a(ii)=b0(ii);
                b(ii)=a0(ii);
                x=find(a==a(ii));
                y=find(b==b(ii));
                i1=x(x~=ii);
                i2=y(y~=ii);
                if ~isempty(i1);
                    a(i1)=a1(ii);
                end
                if ~isempty(i2);
                    b(i2)=b1(ii);
                end;
            end;
        end;
    end;
    pop2(j,:) = a;
    pop2(j+1,:) = b;
end;
%变异
for j = 1:popSize;
```

```
            r = rand();
            if r < mutationProbabilty;
                xx = randperm(city_num);
                x1 = pop2(j,xx(1));
                pop2(j,xx(1)) = pop2(j,xx(2));
                pop2(j,xx(2)) = x1;
            end;
        end;

        %精英选择
        fit1 = fitness1(pop2,end_p,d);
        fit2 = [fit,fit1];
        [~,index] = sort(fit2);
        pop3 = [pop;pop2];
        fit = fit2(index(1:popSize));
        pop = pop3(index(1:popSize),:);

        [min_fit(i),index1] = min(fit);
        best_pop(i,:) = pop(index1,:);
        %min_fit(i)
    end;
    y1 = [1,best_pop(maxGEN,:),end_p];
    %plot(point(y1,1),point(y1,2),'r*-');
end
```