2020 年湖南省研究生数学建模竞赛

竞赛承诺书

我们仔细阅读了湖南省研究生数学建模竞赛的竞赛规则。

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

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

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

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

我们参赛选择的题号是（从组委会提供的试题中选择一项填写）：

我们的参赛报名号为（如果组委会设置报名号的话）：

所属学校（请填写完整的全名）：

参赛队员（打印并签名）：1.

2.

3.

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。）

湖南省人民政府学位委员会办公室
湖南省教育厅学位管理与研究生教育处

2020年湖南省高校第四届研究生数学建模竞赛

题 目：定向越野比赛的路线设计

摘 要：

本文就定向越野路线规划设计的相关问题展开了分析与讨论并对相应问题进行了验证与回答。

具体而言，为制定出一组满足约束且比赛容量尽可能大的路线集，这里采用了先生成可行路线后优化评价的解耦分析方法。进一步分析题中约束后，提炼出了路线检查点个数、路线完成时间以及尽可能连续和顺序打卡的三点约束，并以此为基础采用马尔科夫链方法混合考虑距离信息的概率矩阵来随机生成待选路线。同时考虑连续性要求，进一步的对所得的待选路线采用了贪心算法进行最短路径为目标的顺序重排。

通过调整概率函数的方差，可以获得海量满足基本约束的路线。故这里可以通过数据筛选与聚类分析找到同一终点、里程长度相近、总难度相近并且多样性丰富的路线集。进一步的，这些得到的路线集将是优化部分的输入参数。

针对问题一、二，为获得最大的比赛容量，这里采用基于遗传算法的三层优化嵌套来确定达到最大比赛容量所需的路线数、具体路线以及不同路线的时间间隔。主要考虑“同时到达”约束，即更多的路线可能会带来更多的重复点进而不满足比赛路线要求。优化后问题一可以实现 27 条路线容纳 724 名参赛选手，路线平均总里程 6 公里，平均总难度 63 分钟；问题二可实现 30 条路线容纳 525 名选手参与，平均里程 6.5 公里，平均总难度 75 分钟，平均爬高 62 米。

进一步分析表明，由于该方法未耦合考虑路线生成与优化选择，目前较为受限于路线集的生成，并且多层嵌套的优化过程大大降低了运算效率，文中为提高速度，路线数和时间间隔均采用整数形式优化，从待选路线集中选择 n 条路线时采用随机方法替代。

关键字：路线生成器；大数据；优化；马尔科夫链；贪心算法

1 问题重述

1.1 问题背景

定向越野是一项借助地形图和指北针，按规定的顺序独立地打卡若干个检查点，并以最短的时间完成任务的一项户外运动。在定向越野比赛中，比赛路线一般由一个起点，若干个检查点和一个终点构成。一次比赛安排若干条从起点到终点的比赛路线，所有路线的起点和终点相同。每一条比赛路线可安排一个或多个参赛选手，选择同一条路线的参赛选手按照一定的时间间隔错时出发，不同路线的首发参赛选手同时出发。所有参赛选手须按照所选路线中检查点的顺序依次打卡，不同检查点的打卡难度不一样，通常与打卡点附近的地形和打卡要求有关某次定向越野比赛已知起点和候选检查点，在活动的目标、场地限制等的约束下设计出总参赛选手数量以及路线图。

1.2 问题提出

对于问题 1：

该比赛所已知的起点和候选检查点如图 1 所示：

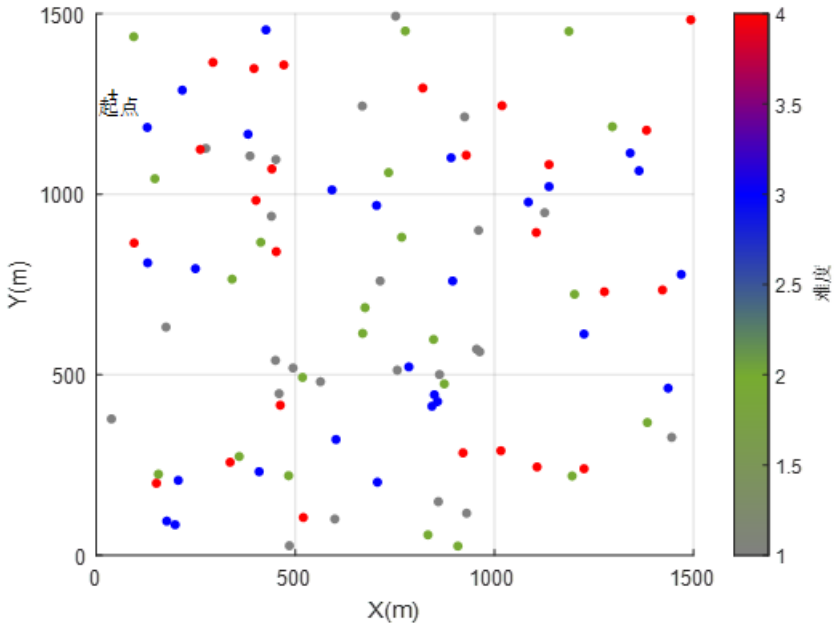


图 1 比赛已知的起点和候选检查点

图中不同的颜色代表不同的难度。考虑为二维平面的路线设计，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，

对于问题 2：考虑三维平面的路线设计，其他同问题 1。示意图如图 2.

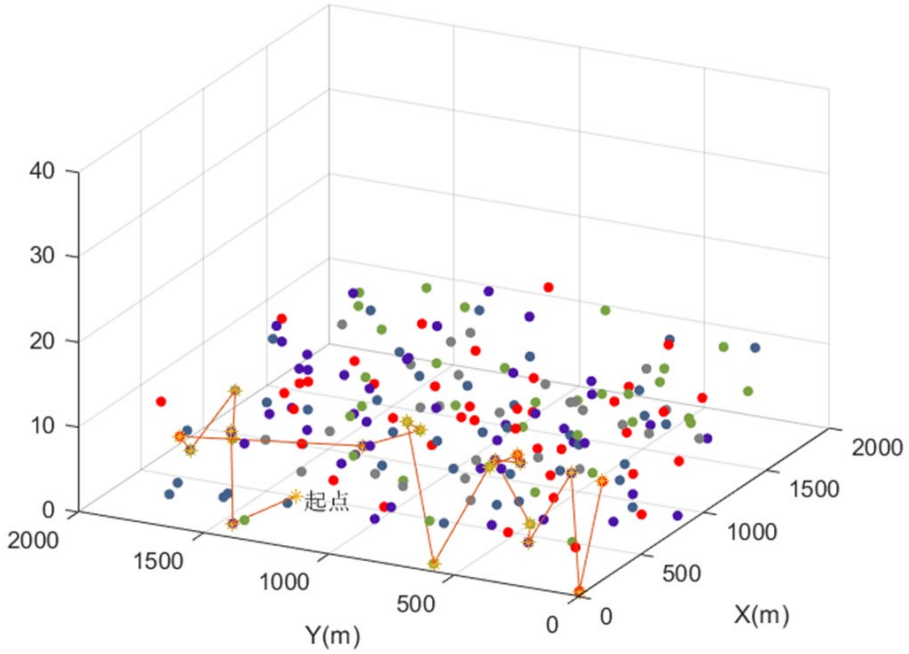


图 2 三维空间各个检查点分布情况

对于问题 3：120 名参赛选手参赛，其识图能力以及体能情况已知，希望各参赛选手平均完成时间不低于两小时，总完成时间不超过 3.5 小时，设计出完成时间尽可能少，数量尽可能少的路线。所有参赛选手的平均速度为 6km/h，与问题 1,2 一致。

以上路线设计考虑以下规则:

(1) 不同路线的长度尽可能均匀。

(2) 不同路线的总打卡难度尽可能均匀。

(3) 不同路线的爬高量应尽可能均匀。

(4) 不同路线之间尽可能避免包含较多重复路段，也应避免参赛选手在完成任务过程中距离过近。

(5) 检查点数量不低于 20 个，包含两个必经的打卡点。重点设置在难度较低的检查点。

(6) 同一路线中相近的检查点打卡顺序尽可能连续。

(7) 到达同一检查点的时间差不超过 1 分钟的人数不超过 5 人。

2 问题分析

2.1 问题 1 的分析

二维平面中的路线设计中，生成的路线满足一下约束：对于不同路线长度尽可能均匀即为每条路线中检查点连线总长度相近；总打卡难度尽可能均匀即为所有检查点时间总和相近；由于选手的速度都是一样的，避免参赛选手跟跑和相互交流即为每条路线出发间隔要适当；每条检查点数量不少于 20 个，所有路线至少有两个共同检查点，终点所在的检查点难度较低；对于任一条路线，路线的总长度尽可能短，对于相近的检查点，其打卡顺序要连续；不同的参赛选手到达同一检查点的时间差不超过 1 分钟的人数不超过五人；要求完成任务平均时间不低于 2 小时，又因为每人的行进速度一样，而且每条路线的长度基本一致，因此可认为是每条路线的总时间不少于两小时，即行进时间和找点时间总和大于等于 2 小时，所有路线的最后一个完成者时刻应是 11 点之前，即距离比赛开始 3 小时内。目标是参赛选手最多。

2.2 问题 2 的分析

三维空间中的路线设计中相比问题 1 中约束，再增加一项约束，每条路线的爬高量相近。

2.3 问题 3 的分析

与前两问不一样，参赛选手数量已知，所有路线的总后完成时刻距离比赛开始不能超过 3.5 小时，目标是使线路数目最少，比赛结束时间尽可能早。

3 模型假设

为便于研究分析，现对题目中某些条件进行简化和合理假设：

(1)对于单条路线而言，所有检查点都是顺序排列且无重复的，即比赛时路线中的检查点只打卡一次；

(2)“同时到达”即为参赛者到达同一检查点时间差不超过一分钟；

(3)参赛选手行进速度即为平均速度。

(4)参赛选手在完成任务过程中按照任意两检查点之间的直线行进。

(5)假设每条路线的第一个参赛者出发的起始时刻为 0。

(6)假设每条路线的出发时间间隔为整数。

(7)假设检查点的难度用参赛选手需要在该点打卡花费的时间来衡量，其值越高，难度越大。

4 符号说明

表 1 符号说明

符号	含义	单位
P_i	参赛队编号标识	/
$u(P_i)$	不同参赛队平均速度	m / min
$\overline{u}$	不考虑差异参赛队差异的平均速度	m / min
$R_k\left(N\right)_{1\times n}$	第 k 条路线由顺序排列的检查点序号表示	/
N	所有检查点的序号索引	/
n	路线检查点数量	
Sum_t_k	第 k 条路线的完成用时（$\overline{u}$）	min
Sum_dis_k	第 k 条路线的总里程	m
Sum_hard_k	第 k 条路线的总难度	min
D_{ij}	检查点 i、j 间的距离矩阵	m
Δt	间隔出发时间	min
T_hard	检查点难度	min
S_i	路线标识号	/
R_i	第 i 条路线参赛者人数	/
d_i	第 i 条路线上参赛者出发时间间隔	s
A	路线的共同检查点集合	/
a_m	线路共有的第 m 个检查点	/
T^m	到达第 m 个共有的检查点时刻的集合	/
t_i^m	第 i 条路线上首发参赛者到达所有路线共有的第 m 个检查的时刻	s
R'	所有参赛者的总数	kg

5 模型的建立与求解

5.1 建模分析

由上述分析可知，在未给定路线检查点总数、路线数、路线选手数以及间隔出发时间等具体参数下，该路线设计问题具有较大的自由度，与之而来就是整体优化设计的复杂性以及目标的多元性。深入分析不难发现，比赛路线的规划设计实际是一个耦合了路线生成与路线评价的双重问题。并且二者之间的信息是不对等传递的，即设计出的路线可以为参赛容量等指标提供基础，而评价部分本身是很难反馈信息去指导优化路线生成的（只反馈对此路线的接受与否）。

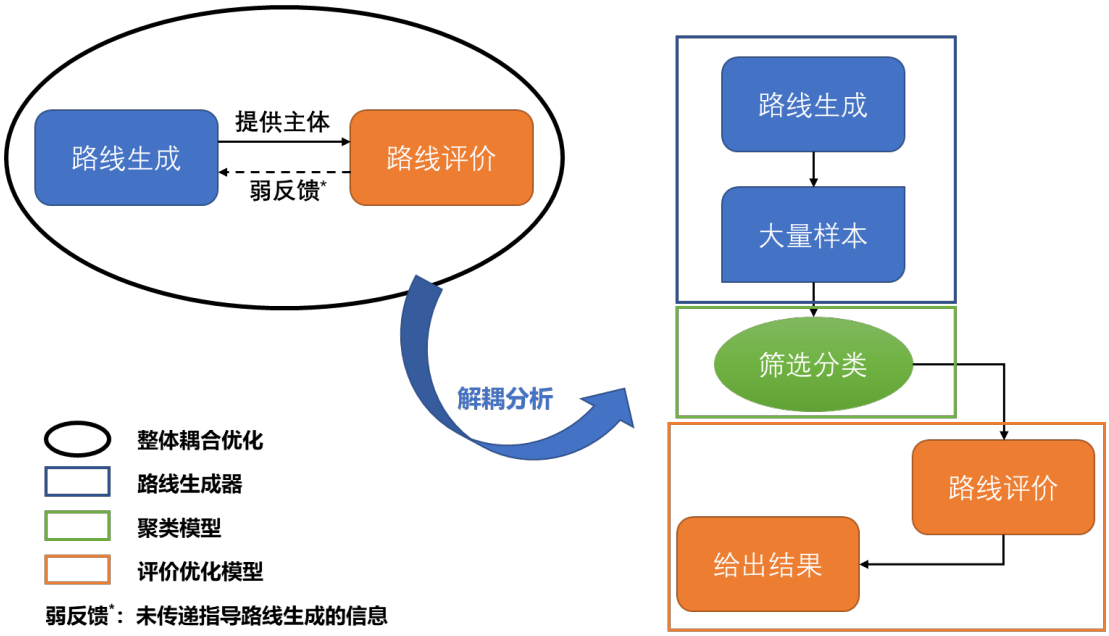


图 3 解耦分析图

因此采用整体式的优化方案十分复杂且可以预计其计算成本是较高的，故本文采用了分步解耦的思路。先一步地，可以通过基于马尔科夫链的状态演化方法生成一系列初步满足约束的可行路线$[R_1, R_2, R_3, \cdots, R_n]$，采用向量相似方法对该部分数据进行无监督的聚类与筛选，海量的待选路线经过本次筛选后就可以送入竞赛总容量的优化评价模型中进行优化，进而获得满足所有约束的最优路线规划。下面将具体描述这三部分的建模。

5.2 路线生成模型

理论上讲一个确定起始点并且待选样本点有限可列的问题，其所有可能的路线是可以穷举的，由于假设 1 的限制，待选样本集容量是递减的。假设路线有

20 个点时并且取问题 1 所给的检查点集合时，可以估计此种列举规模为 $\frac{99!}{79!}\approx 1\times 10^{39}$，这显然是无法处理的。进一步的当路线所需检查点的数量也不确定时，列举将更加复杂，并且这些庞杂的结果满足约束的仍然是小部分。进一步分析可以发现，与路线生成主要相关的约束有三条即：路线完成时间不少于 120 分钟；路线检查点数量不少于 20 个以及相邻检查点尽可能连续、顺序打卡路径尽可能最短。凭借这三条约束我们建立了基于马尔科夫链的状态演化模型用以生成大量的待选路径。

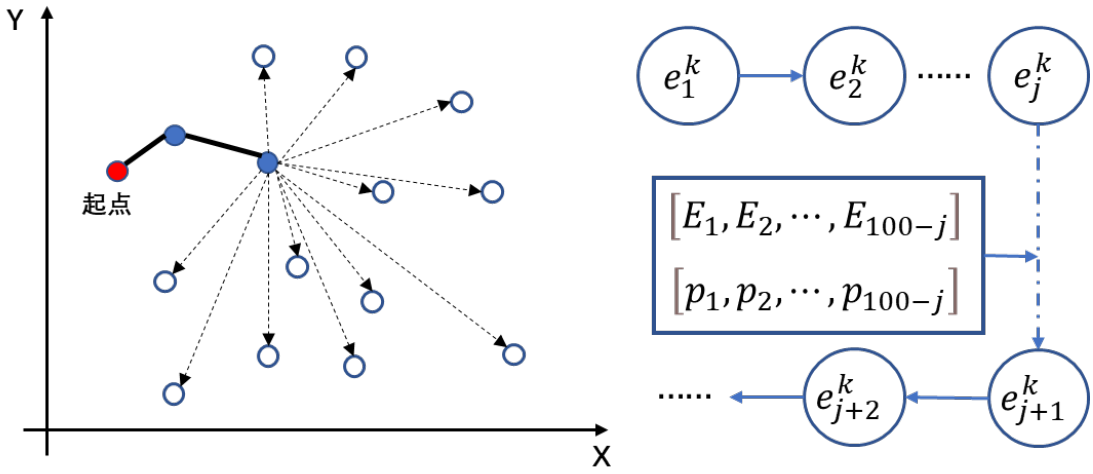


图 4 马尔科夫链示意图

假设 2 做出了当前点的下一步连接选择与历史状态无关，这样一来第 k 条路线第 j 个状态的下一步检查点的选择就可以通过待选状态向量 E^k 和状态转移概率向量 p^k 来决定。

$$E=\left[E_1,E_2,\cdots,E_{100-j}\right] \tag{1}$$

$$p=\left[p_1,p_2,\cdots,p_{100-j}\right] \tag{2}$$

此时已确定的路线向量为：

$$R=[e_1,e_2,\ldots e_j] \tag{3}$$

从待选状态向量中随机选出的状态即为路线的下一步状态，并且新的状态记录进路线向量 R 的同时，待选向量中需要删除已选出的检查点：

$$\begin{aligned} e_{j+1}&=E_s \\ E&=\left[E_1,E_2,\cdots,E_{100-j}\right]-\{E_s\} \\ R&=[e_1,e_2,\ldots e_j]+\{e_{j+1}\} \end{aligned} \tag{4}$$

路线经历历程以及经历时间、经历难度均可累加记录：

$$Sum\_dis = Sum\_dis + D(e_j, e_{j+1}) \tag{5}$$

$$Sum\_t = Sum\_t + \frac{D(e_j, e_{j+1})}{\overline{u}} + T\_hard(e_{j+1}) \tag{6}$$

$$Sum\_hard = Sum\_hard + T\_hard(e_{j+1}) \tag{7}$$

这里矩阵 D 是对角线为 0 的方阵，它存储了检查点之间的距离信息。路线生成的终止条件为：

$$\begin{aligned} j &\geq 20 \\ Sum\_t &\geq 120 \end{aligned} \tag{8}$$

采用随机的方法虽然避免了穷举的海量计算与存储，但在生成基本可行的路线方面，随机方法仍需选择合适的选取方式以及额外的约束以此来提高生成结果的有效性。

最简单的可以考虑状态转移概率为均匀分布，即在所有待选检查点中任意选择下一步的状态。可以预见的是这样选择的结果将存在很大的不确定性，图 5 给出了采用均匀随机下的一个结果。

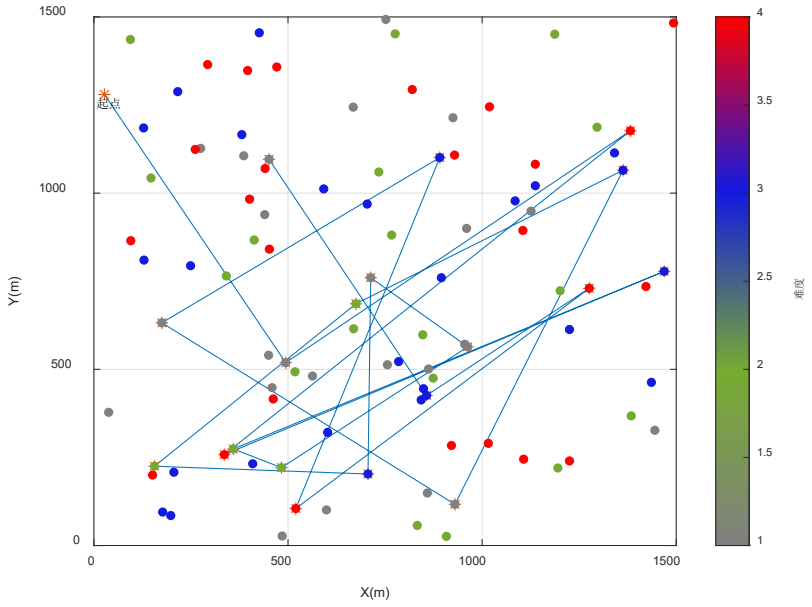


图 5 均匀随机下的结果路线图

5.2.1 概率生成方式

可以看出这样的结果是明显不可行的，在一些距离相近的点上本该就近连接的，反而随机到了较远的点。完全随机是没有利用到检查点之间的距离信息的，为利用距离信息，这里进一步加入了阈值 d 来限制每一步的距离，即在随机挑选之前先用距离信息排除一些待选检查点，使得最终确定的点均满足：

$$D(e_j, e_{j+1}) \le d \tag{9}$$

图 6 给出了采用不同阈值的一些结果，可以看出采用距离信息先一步筛选后，生成的路线确有所改进。

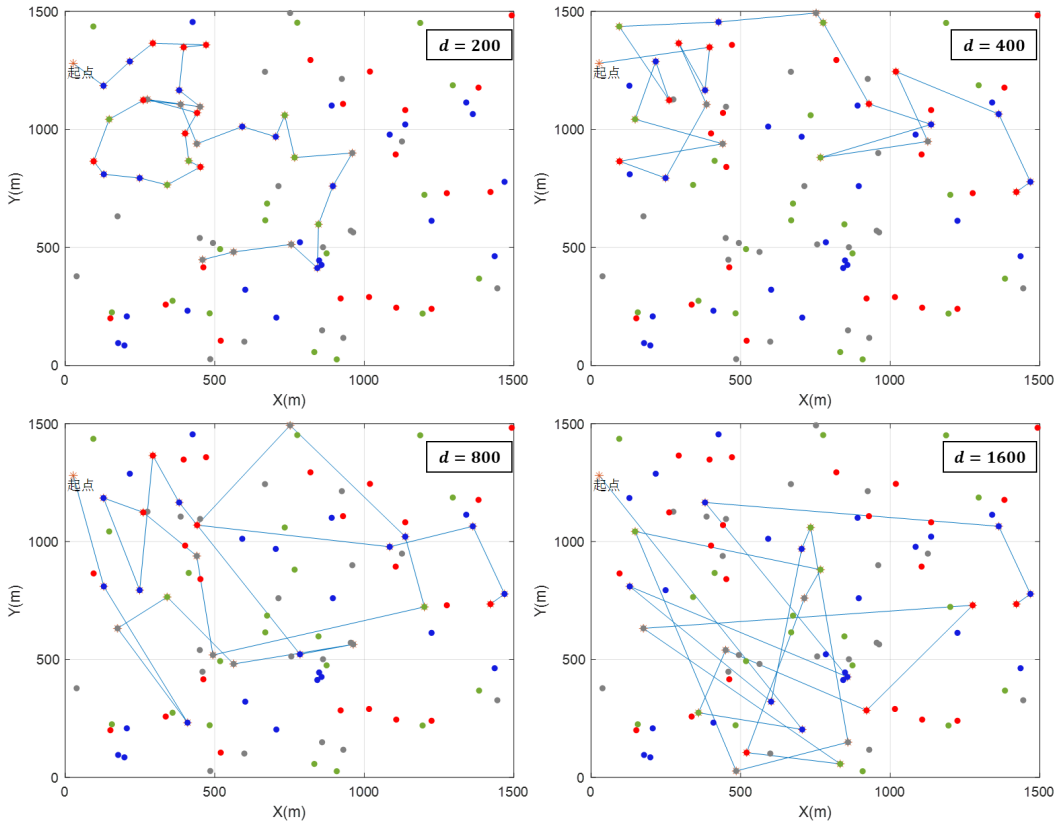


图 6 路线筛选效果图

但这里仍需注意的是，采用固定的阈值 d 是不好确定的，过大的 d 值会导致生成的数据中有效路线占比过低，而太小的 d 值又会导致结果向“最短路线”收敛（贪心下的最短）从而失去大数据的多样性。同时还会存在阈值筛选过后，待选检查点中距离仍然相差较大的情况，例如采用 d=400 筛选后待选检查点距离为 [100,125,134,399,399]，此时再次取随机后又有可能跳过较近的点。

图 7 即为距离阈值取 200 时的两次结果，明显看出其检查点的选取是趋于一致的。

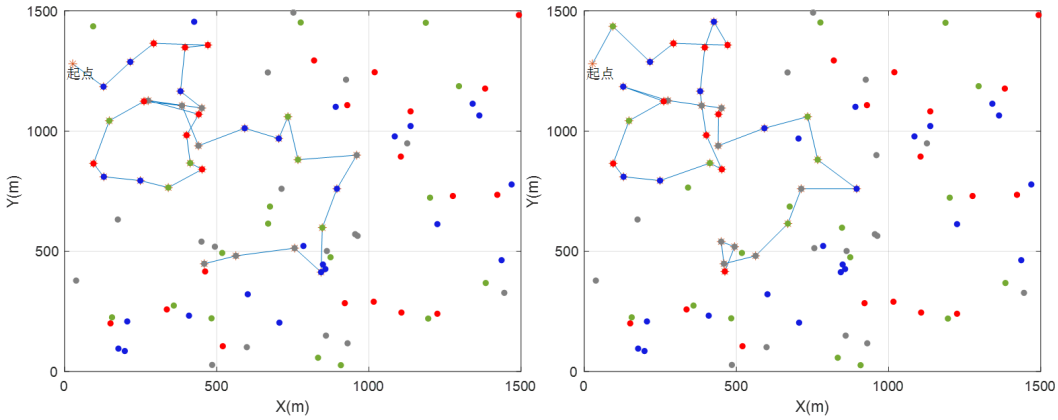


图 7 距离阈值取 200

为了同时保证数据的多样性和结果的有效性，进一步可采取正态分布的概率形式依距离信息来选择下一步的检查点。具体而言，第一步需要对待选状态向量 E 按照距当前检查点的距离进行排序：

$$E\_new=[E_1,E_2,\cdots,E_{100-j}]$$
$$D(E_1)\le D(E_2)\le\cdots\le D(E_{100-j}) \quad (10)$$

排序后的待选检查点从左至右距当前点的距离依次增大，为使随机选择尽可能落在距离较近的点上，这里采用近似正态分布的概率。下图给出了百万次选择下的频数分布。

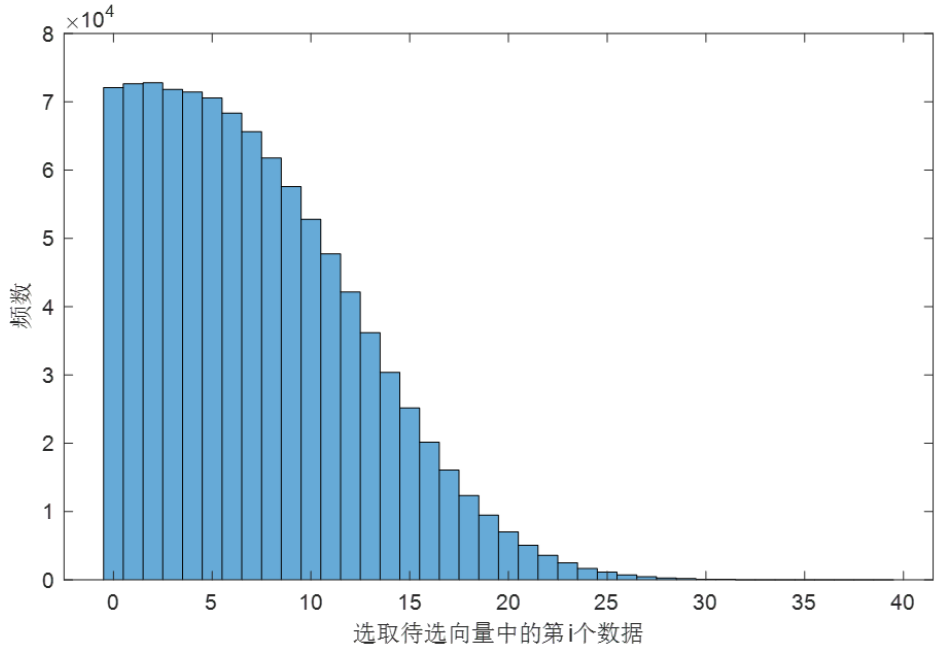


图 8 选取待选向量中的第 i 个数据频数图

采用上述分布概率选取检查点后，路线多样性的确有所保证，下图给出了一

条考虑距离信息的生成路线。这里需要注意的是红圈标注的区域里明显出现了相邻点不连续的情形，这与约束(6)相悖。因此为考虑顺序打卡的连续性，这里进一步引入“最短路线”修正。

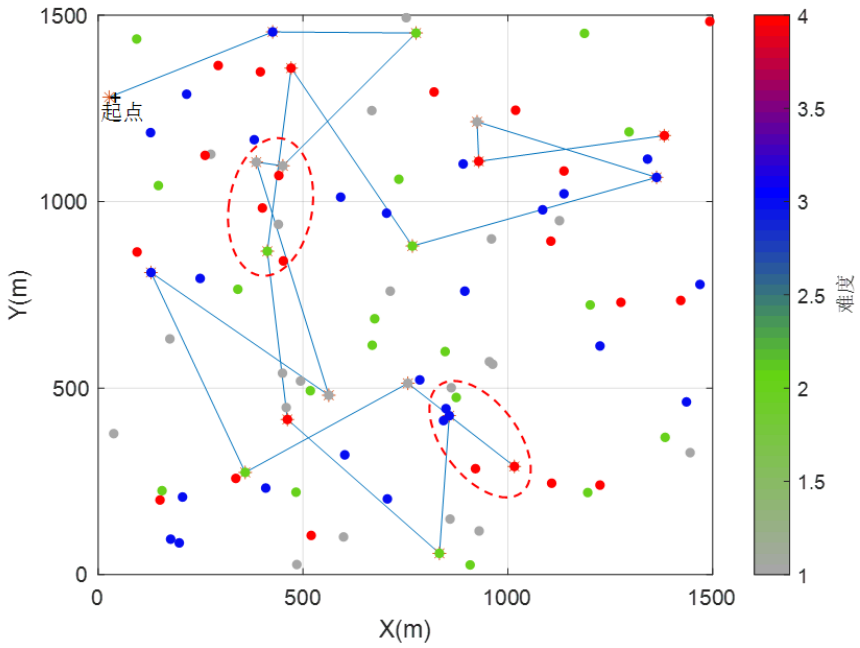


图 9 最短路线修正示意图

所谓最短路线就是对上述随机生成的路线点进行顺序重排，重新排列的顺序要保证相邻点尽可能连续以及路径最短。

同时为防止出现修正后的新路线不再满足总时间约束大于 120 分钟的约束，修正后的新路线要重新进行判断若不满足时间长度则继续随机生成，若满足则输出修正后的路线。

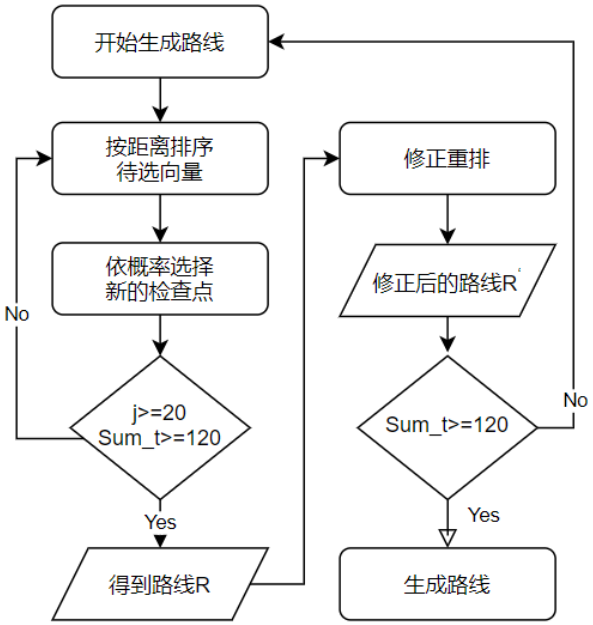


图 10 路线生成部分流程图

图 10 给出了路线生成部分的完整计算流程，其中修正重排部分目前采用了贪心的方法，即对所取得的检查点序列按照每次距离最短的局部最优解进行顺序重排。

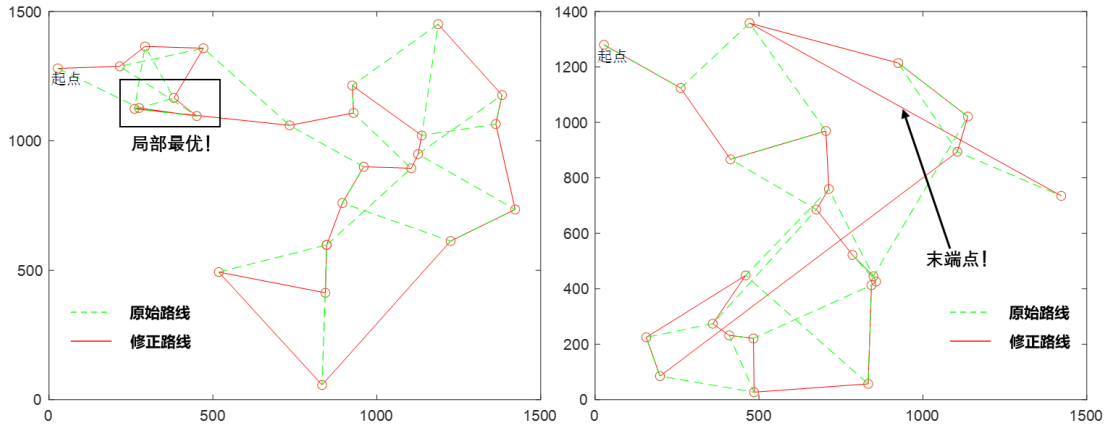


图 11 重排后结果示意图

从上面的结果来看，修正重排后结果较原始路线连续性优异许多，作为定向越野比赛的路线而言也更为合理。不过这里仍需注意的是，采用贪心算法会带来局部最优的问题。如图中四个相邻点之间的连接由于采用了局部最优的思路从而出现了折返的现象。同时排序的过程中最后两个点也可能出现相距较远的情况。从定向越野的现实要求来考虑，实际上这两种情况都是可以接受的。

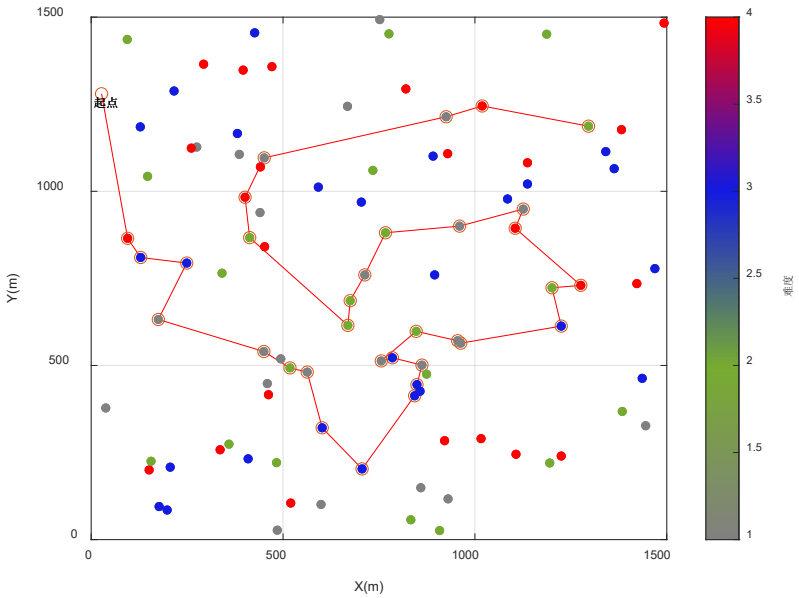


图 12 完整路线生成模型

5.3 路线筛选与分类

5.3.1 路线筛选策略

上一小节生成的 10 万条路线是基于概率随机生成的，这些路线满足了平均时间 120 分钟的时间约束，最小 20 个检查点的检查点数量约束，以及路线的连续性约束。但是这些路线还无法直接作为定向越野的设计路线，需要考量题目给出的路线设计原则，对生成的路线进行筛选，剔除不满足基于生成的大量路线，更进一步的修正我们设计的路线。

路线筛选的具体思路如下：

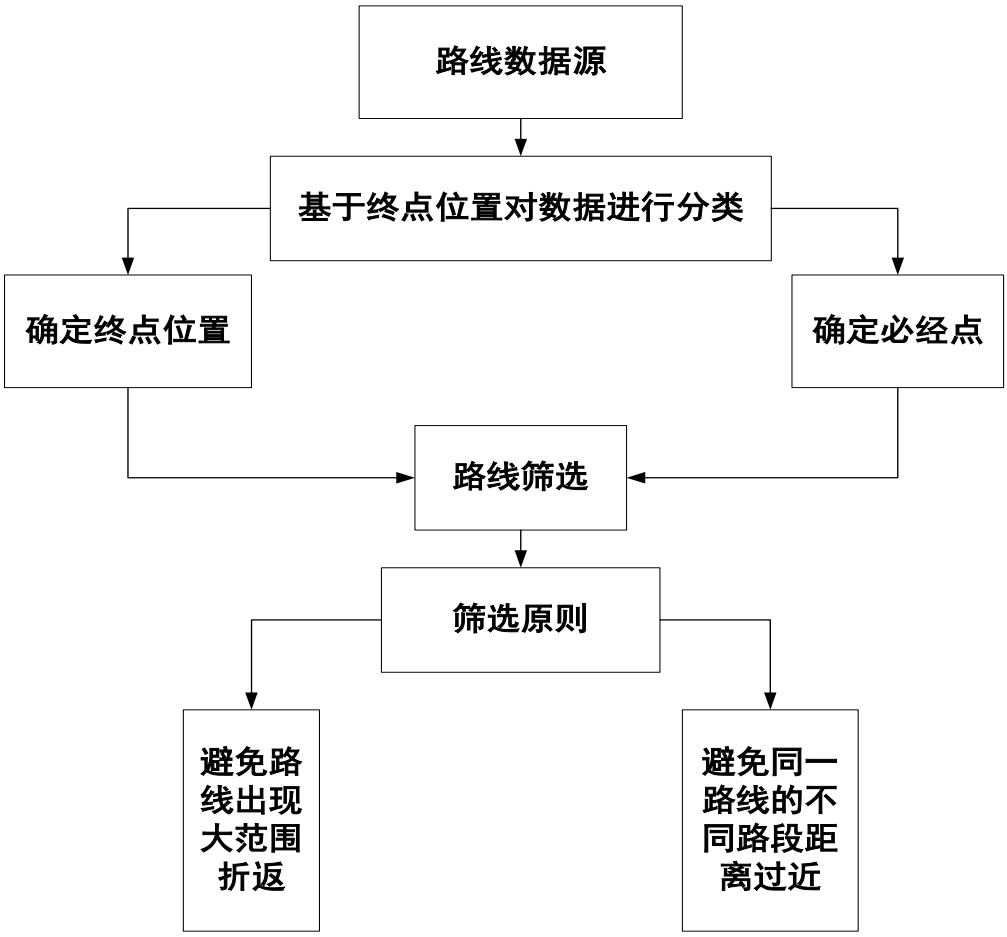
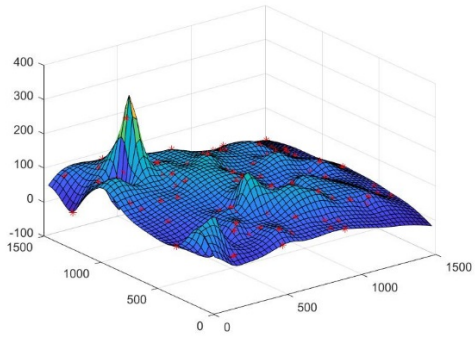
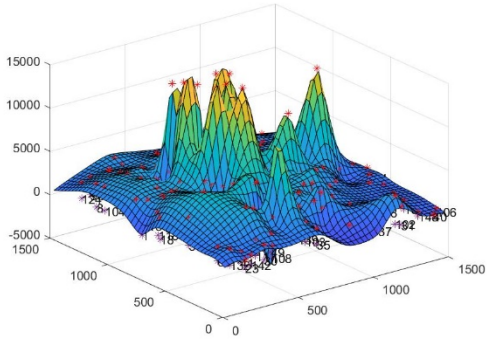


图 13 数据筛选流程图

基于不同终点路线数据源，依照不同终点对应的路线数据量大小和终点的难度作为原则，将数量较多且难度较低的点筛选出来作为我们的终点。

获得终点后，基于该终点对应的路线数据类，对检查点进行筛选。不同路线上有不同的检查点，获取不同检查点出现的频率。将高频检查点筛选出来，作为备选。下一步对备选检查点进行两两组合作为必经打卡点，通过不同组合的待选检查点获得每种组合下的路线。将对应路线数量最多的必经点组合作为我们的必经打卡点。检查点的频率图如下所示：





问题 1 检查点频率图　　问题 2 检查点频率图

图 14 检查点频率图

确定终点和必经打卡点后，对路线进行筛选。参考路线设计的原则（避免跟跑和互相交流），我们以减少大幅度折返和不同路段距离过近为约束，对路线进行筛选，最终获得优化部分所需要的待选设计路线。

5.3.2 路线筛选方法

为了避免大幅度折返对不同路段之间的夹角进行限制，认为如果相邻的两个路段之间的夹角小于 10°，那么就属于大幅度折返，导致相邻路段之间距离过近，会发生跟跑和选手交流的情况。因此我们对待选路线进行筛选，去除夹角过小的路线。

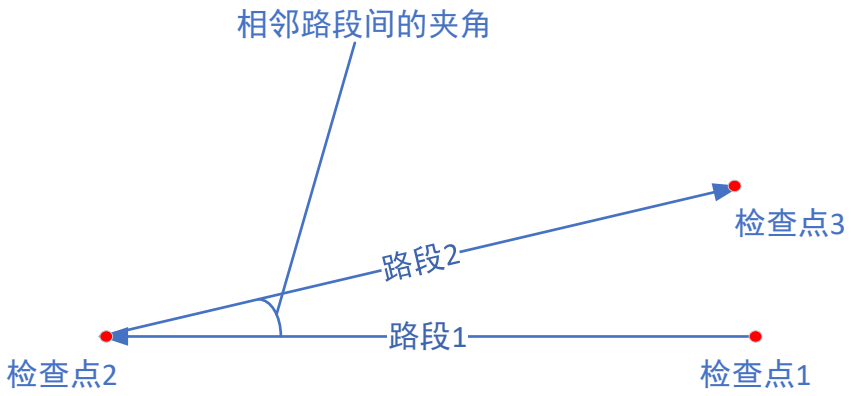


图 15 筛选示意图

另一方面为了限制非相邻路段的长时间的并行导致的选手交流现象，对不同路段的相似程度进行判断，包括余弦相似和距离相似。余弦相似通过不同路段之间的斜率进行限制，认为当其余路段 nk 和路段 n1 的余弦角小于 10°时，我们认为满足预余弦相似；另一方面由于一直不同路段的起始点和结束点，所以我们可

以通过 y=kx+b，来描述线段。通过分析根据点到直线的距离大小来确定距离相似程度。这里我们认为不同路线之间的距离差在 30m 以内，就已经会出现选手交流的现象。

点(x_0, y_0)到线 $Ax+By+C=0$ 的距离公式如下：

$$\frac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}} \tag{11}$$

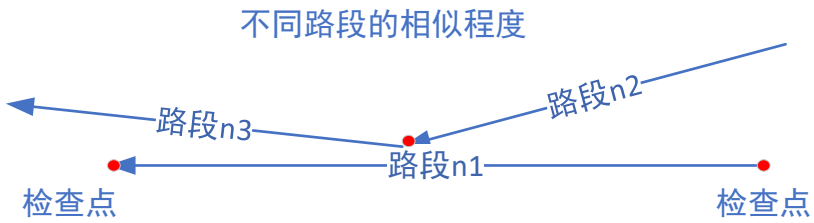
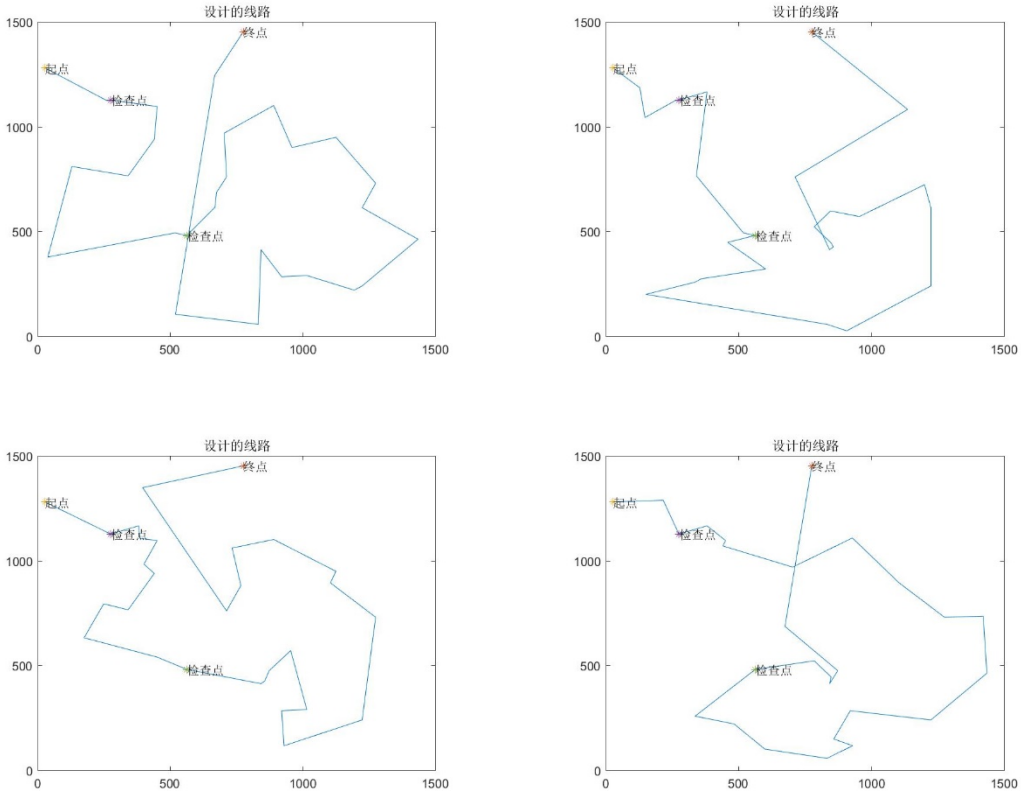
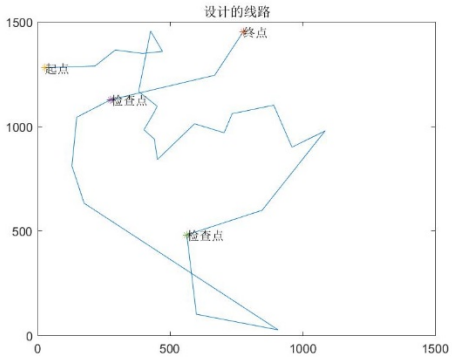
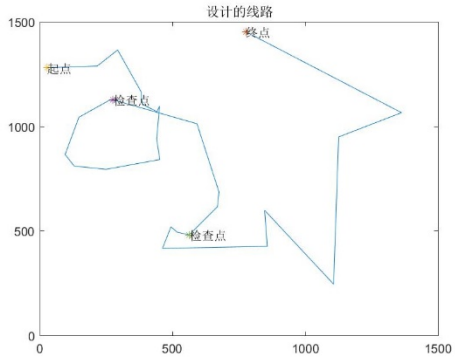


图 16 筛选示意图

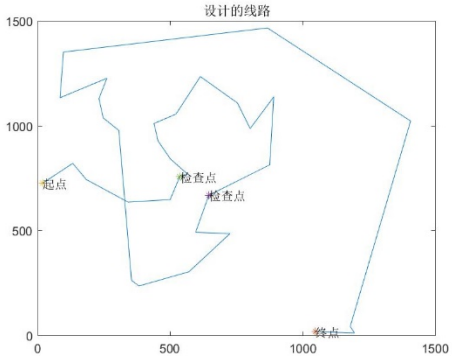
基于上述的筛选和分类方法，我们得到的部分典型路线如下图所示：

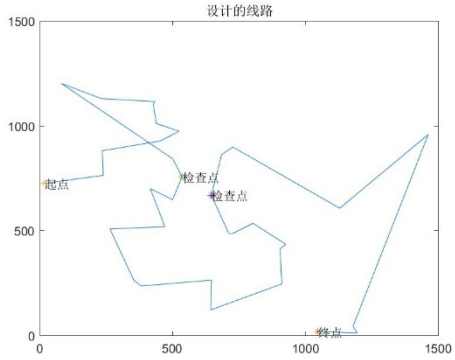


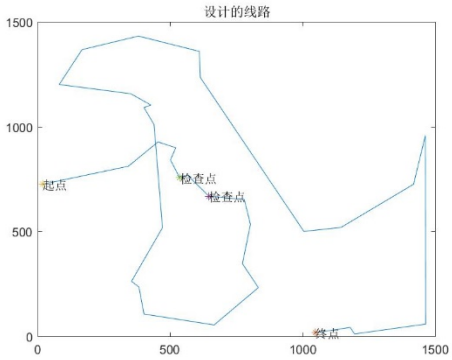

设计的线路
终点
起点
检查点
检查点
0
500
1000
1500

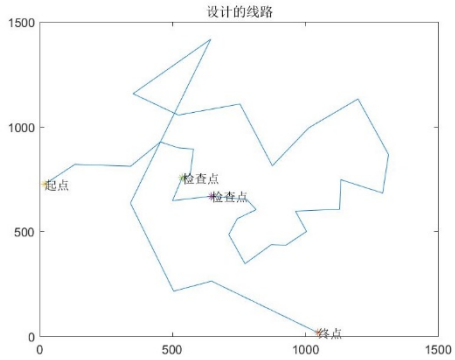

设计的线路
终点
起点
检查点
检查点
0
500
1000
1500

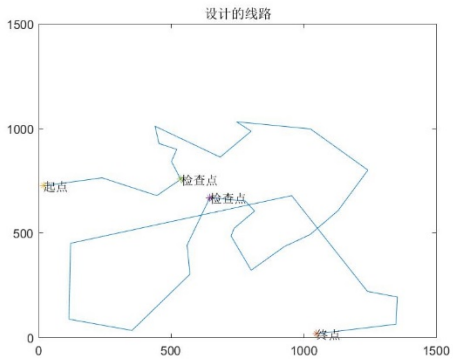
问题 1 典型路线

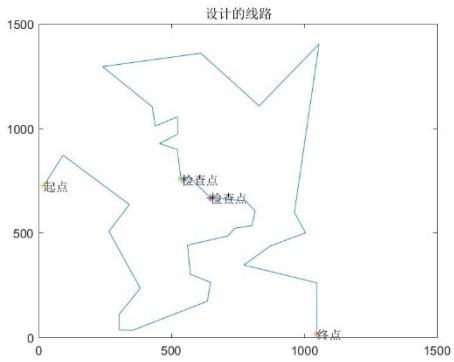

设计的线路
起点
检查点
检查点
终点
0
500
1000
1500


设计的线路
起点
检查点
检查点
终点
0
500
1000
1500


设计的线路
起点
检查点
检查点
终点
0
500
1000
1500


设计的线路
起点
检查点
检查点
终点
0
500
1000
1500





问题 2 典型路线

图 17 部分筛选结果图

5.4 优化模型

这里采用三层嵌套方法同时使用 MATLAB 内置遗传算法进行优化。第一层优化目标为路线数目 k，第二层为 k 种最佳路线，第三层为每条路线的出发时间间隔，假设 b 条路线 $S_i(i=1,2\cdots b)$ 上有 $R_i(i=1,2\cdots b)$ 个参赛选手，线路 S_i 出发间隔为 d_i，从上述路线任取 k 条线路 $S_1,S_2,S_3,S_4,S_5\cdots S_k$，k 条线路中的共同的检查点集为 $A=\{a_1,a_2,a_3,a_4\cdots a_m\}$，对于其中每一共有检查点 a_m 分别计算每个参赛者到达该检查点的时刻，那么对于线路 $S_i(i=1,2\cdots k)$ 上所有参赛者的到达检查点 a_m 的时刻为 $T^m=\{t_i^m \quad t_i^m+d_i \quad t_i^m+2d_i \quad \cdots \quad t_i^m+(R_i-1)d_i | i=1,2,3\cdots k\}$，将所有时刻标在时间轴上，如下图所示

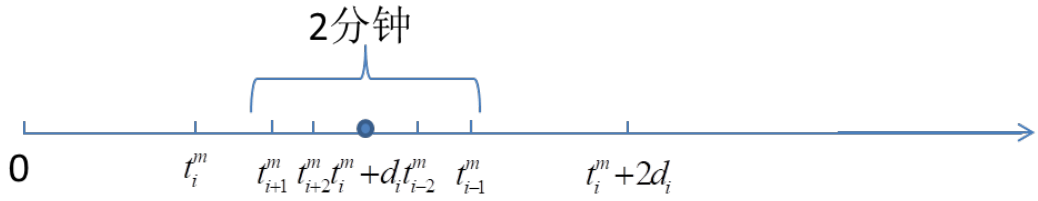


图 18 约束示意图

以每一时刻为圆心，一分钟为半径，建立集合，每个集合中包含 T_m 中元素个数不超过五个，以此为约束，若超过五个则罚函数极大；目标函数即为参赛者总人数最多即 $R'=\sum_{i=1}^{k}R_i$ 最大。

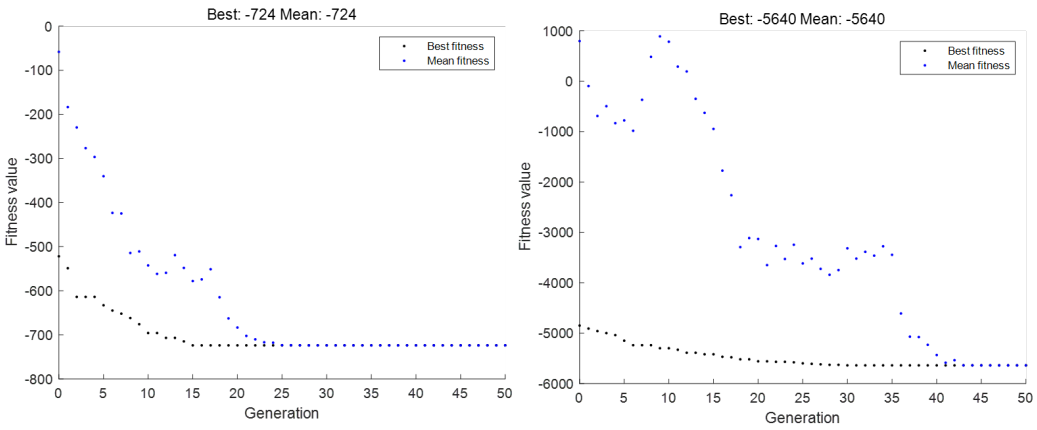


图 19 遗传算法优化示意图

如图 19 所示，分别为路线数目取 8 和 25 时对时间间隔进行遗传算法优化示意图，这里时间间隔取值范围为 2-8 分钟，显然在路线数较少时重叠点上的“同时到达”约束并不苛刻，而当路线数目进一步增多时，均采用最短时间间隔可能无法再满足“同时到达”约束，此时通过调整不同路线间的时间间隔是完全可行的。

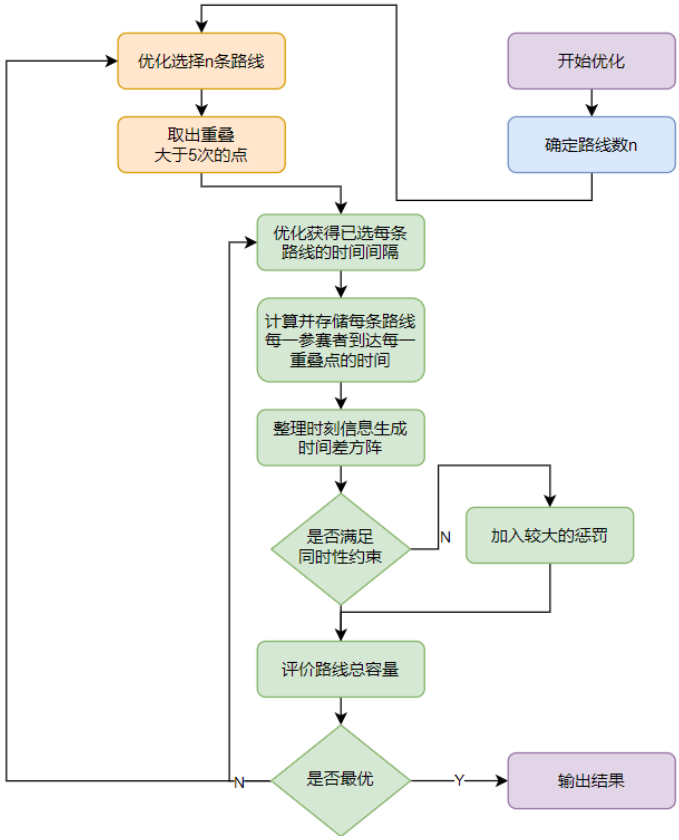


图 20 优化部分路程图

6 模型的评价

6.1 模型的优点：

（1） 本模型首先从概率论的角度，建立马尔科夫链模型，采用贪心算法生成了多条符合其中规则的路径，在对其进行聚类，筛选得到具有至少两个共同检查点且终点相同的路线；以此为输入，三层嵌套和遗传算法优化了路线数目，路线和每条路线的出发间隔。

（2） 本文充分考虑问题 1,2 需要优化的变量，采用三层嵌套和遗传算法按照一定的优先级顺序进行优化，效果非常明显。

6.2 模型的缺点：

（1） 采用贪心算法生成的最短路线容易产生缺陷，尤其在最后只剩两个检查点时通常跨度大。

（2） 针对问题 3 的算法计算时间太长，需要加速算法。

7 参考文献

[1] 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，2018，027(004):105-111.

[2] 张铁良，邢启明，张洁．定向越野路线设计原则与实践[J]．国防科技，2018(2):123-127.

[3] 张弛，丁轶．基于遗传算法的无人机航路规划研究[J]．信息化研究，2018.

[4] 魏照坤．基于 AIS 的船舶轨迹聚类与应用[D]．2015.

[5] 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，2018，027(004):105-111.

8 附录

由于程序较多，这里展示主程序：

```
Clear all;
clc;
close all;

load('data.mat');

global input1;
global input2;
global Select;
global Dei;
```

```
global Sum_dis;
global Sum_hard;

%input1 = importfile1("D:\程序\定向越野_容纳路线\question_1_road.csv", [1,
Inf]);
load('data3.mat');
input1=XUHAO;
%input2 = importfile1("D:\程序\定向越野_容纳路线\question_1_sum.csv", [1,
Inf]);
%input2(:,input2(1,:)==0)=[];


for i=1:size(input1,1)
    temp_road=input1(i,:);
    temp_road(temp_road==0)=[];   %去零
    temp_road=unique(temp_road,'stable');  %防止数据可能出现的重复
    temp=0;
    temp2=0;
    temp3=0;
    for j=1:size(temp_road,2)-1
        temp=temp+dis(temp_road(j),temp_road(j+1));
        temp2=temp2+dis(temp_road(j),temp_road(j+1))./100+hard(j+1);
        temp3=temp3+hard(j+1);
    end
    input2(i,1)=temp;
    input2(i,2)=temp2;
    input2(i,3)=temp3;
end

u=100;

%%
Sum_pop=0;  %评价值，最大容量人数
best=0;
p=0;
n=20;

while p<10
row_num=size(input1,1);

Num=1:row_num;  %待选路线序号

Select=[];   %已选路线序号

Dei=[];   %存下所有大于 5 的点认为是重叠点
```

```
%先随机取出 n 条基准路线
for i=1:n
    a=size(Num,2);
    temp=randi([1,a]);
    Select(i)=Num(temp);
    %删除已选的
    Num(Num(:)==Select(i))=[];
end

%%%%%%%%%%%%%检测已选路线中的重复点%%%%%%%%%%%%%
a=size(Select,2);

Sum_road=1:100;   %所有序号作为背景板
Sum=[];           %存所有已选路线的序号拼接
Sum_f=[];         %存下每个点出现的频数

%先把已选路线中的第一条放进 Sum 中
temp1=input1(Select(1),:);
temp1(temp1==0)=[];   %去零
temp1=unique(temp1,'stable');  %防止数据可能出现的重复
Sum=[Sum_road,temp1];        %重叠到大背景上
Sum_f=histc(Sum,Sum_road);  %统计出频数


for i=2:a
    temp2=input1(Select(i),:);  %取出下一条路径
    temp2(temp2==0)=[];   %去零
    temp2=temp2(2:end-1);  %掐头去尾
    temp2=unique(temp2,'stable');  %防止数据可能出现的重复

    %与 Sum 比较
    Sum=[Sum,temp2];  %再往后拼成一段
    Sum_f=histc(Sum,Sum_road);  %统计出频数
end

Sum_f=Sum_f-1;  %删去背景板即可获得所有重叠点信息

Dei=find(Sum_f>4);

%%%%%%%%%%判断重叠点是否会出现同时堵塞情况%%%%%%%%%%%
 a=size(Select,2);
```

```
 b=size(Dei,2);
 flag=0;    %堵塞标识

 %先获得每条路线到重叠点之前的里程和难度信息

 Sum_dis=zeros(a,b);
 Sum_hard=zeros(a,b);

for i=1:a
    %老样子还是要先处理一下原始路径信息
    temp_road=input1(Select(i),:);
    temp_road(temp_road==0)=[];   %去零
    temp_road=unique(temp_road,'stable');  %防止数据可能出现的重复

    for j=1:b
        temp=find(temp_road==Dei(b));
        if isempty(temp)>0
            Sum_dis(i,j)=0;
            Sum_hard(i,j)=0;
        else
            for k=1:temp-1

Sum_dis(i,j)=Sum_dis(i,j)+dis(temp_road(k),temp_road(k+1));
                Sum_hard(i,j)=Sum_hard(i,j)+hard(temp_road(k));
            end
        end
    end
end

%挨个检验重叠点是否合格,考虑定时间间隔

a=size(Select,2);


lb=zeros(a,1)+2;
ub=zeros(a,1)+5;

A=[]; b=[]; Aeq=[]; beq=[];
nonlcon = [];
IntCon=[];
%'PlotFcn', @gaplotbestf,
options = optimoptions('ga','PlotFcn', @gaplotbestf,'MaxGenerations',25);
[deta,fval] =
ga(@Ga_deta_time,a,A,b,Aeq,beq,lb,ub,nonlcon,IntCon,options);
deta=round(deta);
```

```
% a=size(Select,2);
%
%deta=3+zeros(a,1);    %每条已选路线都有一个发车间隔

Sum_pop=fitness(deta);

b=size(Dei,2);
Sum_t=[];

for i=1:b
    Sum_t=Sum_dis(:,i)./u+Sum_hard(:,i); %基准时间
    Time_deta=[];    %时间差矩阵

    for j=1:a
        %每条路线有多少人,就跑了多少个时间片段
        if Sum_t(j)>0
        temp_sum=Sum_t(j):deta(j):(Sum_t(j)+deta(j)*(Sum_pop(j)-1));
        Time_deta=[temp_sum';Time_deta];
        %Time_deta=unique(Time_deta);
        end


    end

    num=size(Time_deta,1);

    Time_dif=zeros(num);

 %生成时间差矩阵
    for j=1:num
        Time_dif(:,j)=Time_deta(j)-Time_deta;
        Time_dif(:,j)=abs(Time_dif(:,j));
        %判断时间差在 1 分钟以内的数量
        jce=length(find(Time_dif(:,j)<=1));
        if jce>5
            flag=1;

            break;
        end
    end
```

```
    if flag>0
%        disp("堵塞");

        break;
    end

end

    if flag>0
        disp("堵塞");
        continue;
    end

last=sum(Sum_pop);

if (last>best)
    best=last
    result = table(deta,Sum_pop',Select, 'VariableNames',
{'deta','Sum_pop','Select'});
    writetable(result,'best.csv');
    p=0;
else
    p=p+1
end

end
```