



中国研究生创新实践系列大赛
“华为杯”第十六届中国研究生
数学建模竞赛

学 校 上海交通大学

参赛队号 19102480139

- 队员姓名
-
1. 赵秋阳
-
2. 王金枝
-
3. 邓朝霞
-

中国研究生创新实践系列大赛
“华为杯”第十六届中国研究生
数学建模竞赛

题 目 基于贪心算法和禁忌搜索优化的航迹快速规划

摘要：

对飞行器的航迹进行合理的规划可以降低其航行距离，且提高飞行精度。针对飞行器在复杂环境下的航迹路径规划问题，本文采用多目标 0-1 整数规划模型，并结合题目所给定的约束条件，针对不同问题分别建立不同模型对其进行求解。

针对问题一，对于双目标优化问题，本文引入一个适应度将问题简化为单目标问题。可通过调节收益偏好系数来改变适应度，从而调整不同优化目标的权重。之后通过贪心算法依次求解局部最优解，用其近似全局最优解。通过 Matlab 编程求解，仅需约 0.1s 便可求得最优解。通过求解附件 1 中的数据可得到航迹总长度为 106350.06m，校正点个数（包含起始点与终止点）为 10 个的航迹路线。通过求解附件 2 中的数据可得到航迹总长度为 111585.52m，校正点个数为 14 个的航迹路线。后续再结合禁忌搜索法在贪心算法的基础上对路径进行优化。附件 1 的数据可求得航迹总长度为 104527.36m，所经过的校正点个数为 11 个的路径。附件 2 的数据可求得航迹总长度为 109342.52m，所经过的校正点个数为 14 个的路径。

针对问题二，在问题一的约束条件中增加了转弯半径约束。首先采用几何分析法分析转弯半径与转弯时机对航迹距离的影响，通过计算机模拟的方法得知转弯半径为 200m 且在抵达校正点的瞬时开始转弯可以得到最短路径。依旧采用贪心算法求解，得到附件 1 中的航迹距离为 106440.33m，校正点个数为 10 个。附件 2 中的航迹距离为 11780.13m，校正点个数为 14 个。

针对问题三，我们考虑了校正点中可能存在出现校正失败的情况。首先，采用穷举法计算航迹的成功率。之后以问题一中求得的解作为初始解，通过禁忌搜索法对其进行优化，最终计算出成功率较高且满足目标函数的解。附件 1 中所求得的航迹距离为 $110051.66m$ ，校正点个数为 11 个，成功率为 100%。附件 2 中所求得的航迹距离为 $156679.78m$ ，校正点个数为 24 个，成功率为 76.8%。

关键字： 多目标 0-1 整数规划 贪心算法 几何分析 禁忌搜索 穷举法

目录

1. 问题描述	5
2. 问题假设及符号说明	6
2.1 问题假设.....	6
2.2 符号说明.....	7
3. 问题建模与求解	8
3.1 第一问建模与分析.....	8
3.1.1 问题分析	8
3.1.2 模型建立	9
3.1.3 算法求解	10
3.1.4 结果分析	13
3.2 第二问建模与分析.....	16
3.2.1 问题分析	16
3.2.2 模型建立	16
3.2.3 算法求解	18
3.2.4 结果分析	24
3.3 第三问建模与分析.....	27
3.3.1 问题分析	27
3.3.2 模型建立	27
3.3.3 算法求解	28
3.3.4 结果分析	31
4. 算法复杂度与有效性分析	39
4.1 算法复杂度分析.....	39
4.2 算法有效性分析.....	39
5. 对第一问的改进	39
5.1 对附件一数据的优化.....	39
5.2 对附件二数据的优化.....	40
6. 总结	41
参考文献	43
附录 A 问题一的求解代码	1

附录 B 问题二中计算两点间长度的代码 5

附录 C 问题三的求解代码 6

1. 问题描述

控制智能飞行器在复杂环境中的航迹路径对于飞行器的飞行是至关重要的。由于飞行器在飞行过程中其定位系统不能对所处位置进行精准的定位，而定位误差累计到一定程度时会使飞行器不能到达规定地点。所以，飞行器在飞行过程中需要进行定位误差校正，消除某个方向上的误差。

航迹规划是指在特定的约束条件下，寻找运动体从初始点到目标点满足某种性能指标的最优运动轨迹^[1]。本题需要在规定的区域内，在满足给定的约束条件下，对飞行器的航迹进行规划。区域中不同类型的校正点可对不同方向的误差进行校正，通过选取合乎要求的校正点来确定航迹点，使得航迹的距离最短且经过的校正点个数尽可能少。

飞行器的飞行区域如图1所示，起始点为A点，终止点为B点。其航迹约束如下：

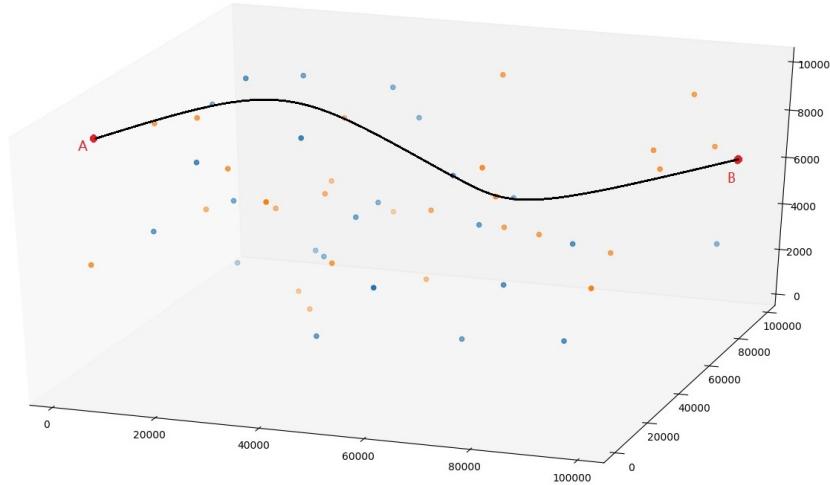


图1 飞行器航迹规划区域示意图

- (1) 飞行器在飞行过程中的定位误差包括垂直误差和水平误差。飞行器每飞行 $1m$ ，垂直误差和水平误差将各增加 δ 个专用单位。到达终点时垂直误差和水平误差均应小于 θ 个单位，且当垂直误差和水平误差均小于 θ 个单位时，飞行器仍能够按照规划路径飞行。
- (2) 飞行器在飞行过程中通过到达不同类型的校正点来校正其定位误差。当飞行器到达校正点时，即可校正垂直或水平方向上的误差。若飞行时飞行器的误差能及时校正，则其可以按照规定路线飞行，经过数次校正后到达目的地。
- (3) 在出发地A点，飞行器的垂直和水平误差均为0。
- (4) 飞行器经过垂直误差校正点后，其垂直误差将变为0，水平误差保持不变。
- (5) 飞行器经过水平误差校正点后，其水平误差将变为0，垂直误差保持不变。

- (6) 当飞行器的垂直误差不大于 α_1 个单位，水平误差不大于 α_2 个单位时才能进行垂直误差校正。
- (7) 当飞行器的垂直误差不大于 β_1 个单位，水平误差不大于 β_2 个单位时才能进行水平误差校正。
- (8) 飞行器无法完成即时转弯 (飞行器前进方向无法突然改变)，假设飞行器的最小转弯半径为 200m。

本题需要求解三个问题，问题如下：

问题 1：根据给定的附件 1 和附件 2 中数据规划满足条件 (1)-(7) 时飞行器的航迹，并考虑两个优化目标，即轨迹长度尽可能小且经过校正区域进行校正的次数尽可能少。讨论算法的有效性和复杂度，绘出相应的航迹规划路径。

附件 1 中的数据参数为： $\alpha_1 = 25, \alpha_2 = 15, \beta_1 = 20, \beta_2 = 25, \theta = 30, \delta = 0.001$
 附件 2 中的数据参数为： $\alpha_1 = 20, \alpha_2 = 10, \beta_1 = 15, \beta_2 = 20, \theta = 20, \delta = 0.001$

问题 2：在问题 1 的基础上将条件 (8) 考虑进去，其余条件和优化目标均与问题 1 保持一致，求解出航迹规划路径。

问题 3：考虑不可控因素情况，假设飞行器在部分校正点能够将某个误差校正为 0 的概率为 80%，若校正失败，则校正后的剩余误差为 $\min(\text{error}, 5)$ 个单位 (error 为校正前误差， \min 为取最小函数)，且假设飞行器到达该校正点时即可知道在该点处是否校正成功，无论校正与否，均不能改变规划路径。在此情况下，求解问题一所要求的轨迹，使得成功到达终点的概率尽可能大。

2. 问题假设及符号说明

2.1 问题假设

根据题目所述，提出如下假设：

- (1) 将飞行器视作一质点，不考虑其形状、大小的影响。
- (2) 在第一、二问中，假定飞行器从当前位置到达下一位置时，其不受突发状况影响，能够抵达所规划的校正点。
- (3) 假定飞行器在进行校正时为瞬时校正，校正前后位置不发生改变。
- (4) 在第一问和第三问中，假定飞行器在两个相邻校正点间为直线飞行。
- (5) 在第一问和第三问中，飞行器转弯是瞬时完成的。
- (6) 在第三问中，假设可能发生校正错误的点中各点成功与失败相互独立

2.2 符号说明

符号	符号说明
n	数据集中点的总个数
h	航迹上点的总个数
P_t	t 时刻飞行器经过的点
T_t	0, 1 变量, t 时刻飞行器经过的校正点的类型, 垂直校正点为 1, 水平校正点为 0
c_{ij}	0, 1 变量, 最优航迹上是否存在从 i 到 j 的路径。若存在为 1, 不存在则为 0
l_{ij}	i 点与 j 点的距离
P_t	飞行器航迹上的第 t 个点
x_t	飞行器在第 t 个航迹点时的水平误差
y_t	飞行器在第 t 个航迹点时的垂直误差
D_t	由 P_t 所确定的下一时刻可行点的集合
d_{it}	点集 D_t 中的第 i 个点
e_{it}	在 t 时刻, 假设飞行器在 $t+1$ 时刻飞行到点 d_{it} , 则飞行器可以在 $t+2$ 时刻飞行到的任意一点
F_t	t 时刻的禁忌点集
γ_{it}	向量 $\overrightarrow{P_t B}$ 与向量 $\overrightarrow{P_t d_{it}}$ 的夹角
ρ	收益偏好系数
Z_{it}	点 d_{it} 对应的紧张度
S_{it}	点 d_{it} 对应的适应度
W	航迹点集
r_t	飞行器从点 P_t 到点 P_{t+1} 的转弯半径
$l'_{P_t, P_{t+1}}$	飞行器从点 P_t 航行到点 P_{t+1} 考虑转弯时的长度

符号	符号说明
φ_{P_t}	飞行器在点 P_t 时的速度方向与向量 $\overrightarrow{P_t P_{t+1}}$ 的夹角
U	航迹中所有可能失败的点的集合
u_i	0-1 变量, 表示 U 中第 i 个点是否校正成功, 校正成功为 0, 失败为 1
Q	一条航迹中所有可能出现的状态集合: 已知 U 中共有 m 点的情况下, U 中所有点的状态(成功或失败)的集合。
q_j	一条航迹中所有可能出现的状态中的第 j 个状态
u_{ij}	航迹状态为 q_j 时, 第 i 个校正点的状态(成功或失败)
g_j	0-1 变量, 航迹状态为 q_j 时, 飞行器是否航行成功, 成功为 1, 失败为 0
a_i	U 中第 i 个点的危险度: 航迹失败 ($g_j = 0$) 的条件下, U 中第 i 个点校失 败 ($u_i = 1$) 的概率
Q'_i	U 中第 i 个点校正失败且飞行器航行失败的所有校正点状态集合
$success$	航迹的成功率
$iter$	禁忌搜索当前循环次数

3. 问题建模与求解

3.1 第一问建模与分析

3.1.1 问题分析

问题一是在规划条件 (1)-(7) 下求解飞行器航迹长度最小且经过校正区域次数最少的航迹。问题一中不考虑飞行器的转弯限制, 即飞行器可以以任意角度转向, 因此本问题主要求解满足约束条件的最优解问题。通过引入航迹约束, 增加启发信息克服搜索的随机性, 使得规划出的可行航迹接近最优航迹^[2]。本次采用 0-1 整数规划模型进行求解, 其所考虑的约束如下:

- (1) 飞行器在正常飞行阶段和抵达终点时, 水平和垂直误差均须小于 θ 个单位。
- (2) 根据所抵达校正点的类型, 可对不同方位的误差进行校正。
- (3) 飞行器的垂直误差不大于 α_1 个单位, 水平误差不大于 α_2 个单位时才能进行垂直误差校正。
- (4) 当飞行器的垂直误差不大于 β_1 个单位, 水平误差不大于 β_2 个单位时才能进

行水平误差校正。

(5) 所选取的校正点能连接成从 A 至 B 的航线。

本题主要在规定航迹规划区域内，找寻一条长度尽可能小且校正次数尽可能少的航迹。问题 1 有两个优化目标，考虑使用 0-1 整数多目标规划模型求解。

3.1.2 模型建立

多约束条件下的路径快速规划问题是求解路径中能够同时满足多个路径属性约束（经过点尽可能少、飞行误差在可允许范围内不改变航迹等），目标函数为航迹最短且经过的校正点尽可能少。下面对该问题进行最短路径建模。

优化目标：

$$\begin{cases} \min\left(\sum_{i=1}^n \sum_{j=1}^n (c_{ij} * l_{ij})\right) \\ \min(h) \end{cases} \quad (1)$$

约束条件：

1) 航迹路线的基本约束：

$$\begin{cases} \sum_{i=1}^n c_{iP_t} = 1, & \forall t \in (1, n) \\ \sum_{j=1}^n c_{P_t j} = 1, & \forall t \in (1, n) \end{cases} \quad (2)$$

$$c_{ii} = 0, \quad \forall i \in (1, h) \quad (3)$$

$$\begin{cases} \sum_{i=1}^n c_{iP_1} = 0 \\ \sum_{j=1}^n c_{P_n j} = 0 \end{cases} \quad (4)$$

$$\begin{cases} P_1 = 0, & \text{起点为 A} \\ P_h = n, & \text{终点为 B} \end{cases} \quad (5)$$

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} = h - 1 \quad (6)$$

2) 保证航线经过点的垂直误差与水平误差均满足校正点约束：

$$\begin{cases} y_t + \delta l_{P_t, P_{t+1}} \leq \alpha_1, & t = 1, 2 \dots h - 2, \quad T_{t+1} = 1 \\ x_t + \delta l_{P_t, P_{t+1}} \leq \alpha_2, & t = 1, 2 \dots h - 2, \quad T_{t+1} = 1 \\ y_t + \delta l_{P_t, P_{t+1}} \leq \beta_1, & t = 1, 2 \dots h - 2, \quad T_{t+1} = 0 \\ x_t + \delta l_{P_t, P_{t+1}} \leq \beta_2, & t = 1, 2 \dots h - 2, \quad T_{t+1} = 0 \end{cases} \quad (7)$$

对于终点 B：

$$\begin{cases} y_{h-1} + \delta l_{P_{h-1}, P_h} < \theta \\ x_{h-1} + \delta l_{P_{h-1}, P_h} < \theta \end{cases} \quad (8)$$

式(1)、(2)为问题1中满足航迹长度尽可能小、经过校正的次数尽可能少的全局目标。式(3)表明取航迹最优点时，无重复取点情况。式(4)表明取航迹最优点时无从自身取自身的情况。式(5)表明A点入度为0，B点出度为0。式(6)表明最优航迹路径的段数为 $h - 1$ 。

式(7)表明对于下一误差校正点 T_{t+1} ，当校正点不为B点时，判断该点校正点类型不同，需要满足一定条件才能进行校准。当校正点为终点B时，只需满足垂直水平误差小于 θ ，如式(8)所示。

3.1.3 算法求解

在实际问题中，通过引入0-1变量，可以把有各种情况需要分别讨论的线性规划问题归纳在一个问题中讨论。在一组等式或不等式的约束下，求一个函数的最大值（或最小值）问题，其中目标函数或约束条件中至少有一个非线性约束，这类问题称之为非线性规划问题，简记为NP问题^[3]。本问题是一个典型的NP-Hard问题，当问题规模增加时，在可接受的时间内难以求得最优解。该问题中可供选择的校正点在300个以上，且约束条件很多，变量和约束条件规模过大。所以我们考虑使用贪心算法求解，在不回溯的前提下寻找问题每一步的最优解，从而得到一个满意解。在复杂的实际问题中，贪心算法不一定能得到全局最优解，但可在限定的时间内以局部最优来近似全局最优^[4]。

本题可根据当前飞行器的位置，采用贪心算法求解下一时刻的飞行器位置，得到当前状态下的最优节点。之后根据所得到的最优节点再继续依照所设定的求解方法求解下一个节点，最终得到最优航迹。具体算法思路如下：

- 1) 首先初始化飞行器的位置 P_0 和当前的水平误差 x_0 与垂直误差 y_0 。飞行器位置 P_0 为A点位置，误差均为0。
- 2) 之后再根据当前校正点的位置和误差，找出所有满足如下要求的校正点，并将其定义为 D_t 。

$$\begin{cases} \delta \cdot l_{P_t, d_{it}} + y_t \leq \alpha_1, \delta \cdot l_{P_t, d_{it}} + x_t \leq \alpha_2, \text{该点为垂直误差校正点} \\ \delta \cdot l_{P_t, d_{it}} + y_t \leq \beta_1, \delta \cdot l_{P_t, d_{it}} + x_t \leq \beta_2, \text{该点为水平误差校正点} \end{cases}$$

由于校正点只能校正水平或者垂直误差中的一项，为了避免下一时刻的校正点 e_{it} 与当前点 p_t 所选择的校正点 d_t 的距离过大，导致飞行器在该校正点未经校正方向的误差过大导致飞行器无法进行下一次的校正，我们对 D_t 中的点进行了第二次筛选。对于 D_t 中的每一点 d_{it} 来说，若存在一点 e_{it} ，满足下列要求，

则保留 d_{it} , 否则将该点从 D_t 中剔除。

$$\begin{cases} \delta(l_{P_t, d_{it}} + l_{d_{it}, e_{it}}) + x_t \leq \beta_2, \delta \cdot l_{d_{it}, e_{it}} \leq \beta_1 & , d_{it} \text{ 为垂直校点, } e_{it} \text{ 为水平校点} \\ \delta(l_{P_t, d_{it}} + l_{d_{it}, e_{it}}) + y_t \leq \alpha_1, \delta \cdot l_{d_{it}, e_{it}} \leq \alpha_2 & , d_{it} \text{ 为水平校点, } e_{it} \text{ 为垂直校点} \end{cases}$$

3) 随后判断 D_t 是否为空集, 若为空集, 则将当前点 p_t 添加到禁忌点集 F_t 中, F_t 中的点将无法被加入到 D_{t-1} 中。随后回退到上一时刻的位置, 误差重置为上一时刻的误差, 返回步骤 2)。若不为空集, 则进行下一步。

4) 计算求解 D_t 中所有校正点 d_{it} 中的适应度, 适应度最大的校正点即为当前最优解。在定义适应度前, 首先确定一下所需考虑的目标参数。本题的最优目标为航迹路线最短且所经过的校正点最少。为了使每一步确定的一个最优节点为最优解, 则需要对所选范围内各点的各项指标进行对比分析, 并选取适应度最优的点。

衡量校正点的优劣主要涉及到的指标有: 当前校正点的垂直和水平误差与题目中设定的运行误差范围限值之差、前进方向与线段 AB 之间的夹角以及点 P_t 与点 d_{it} 的距离。

首先, 定义一个紧张度为 Z_{it} , 如下所示:

$$Z_{it} = \begin{cases} \alpha_1 - y_t, & d_{it} \text{ 为垂直校正点} \\ \beta_2 - x_t, & d_{it} \text{ 为水平校正点} \end{cases}$$

紧张度用以表示当前点的水平和垂直误差与题目所规定的水平、垂直误差范围的差值。其值越小, 则表明该点的误差越接近校正限值, 下一时刻所能够校正的误差越多, 校正点的功效越大。

之后需要考虑前进方向与向量 \overrightarrow{AB} 之间的夹角, 图2为向量夹角的示意图。为了便于计算, 本次选取 $\cos \gamma_{it}$ 作为一个衡量指标。在这里我们不考虑 γ_{it} 为钝角的情况, 因为在当前步骤视角下这种路线使飞行器远离 B 点。即我们令 $\cos \gamma_{it}$ 恒大于零。当 γ_{it} 越小, 即 $\cos \gamma_{it}$ 越大时, 其航迹越靠近线段 AB 的连线, 所航行的距离越小。

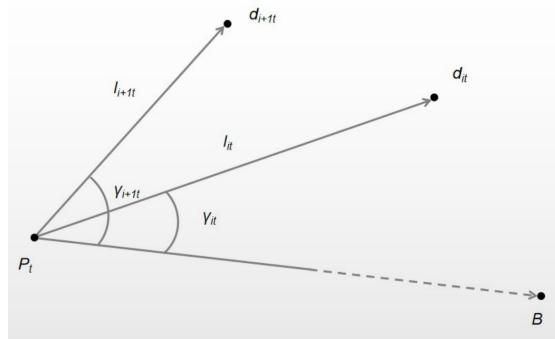


图 2 向量夹角示意图

最后需要考虑的因素是点 P_t 与点 d_{it} 之间的距离，即 $l_{P_t, d_{it}}$ 。其越大，则每走一步之后距离 B 点越近，所需的总的校正点数目越少。

综上所述，可定义适应度如下：

$$S_{it} = \frac{K\rho \cos \gamma_{it} + (1 - \rho)l_{P_t, d_{it}}}{Z_{it}}$$

其中 ρ 为收益偏好系数，通过调整该系数可以得到不同权重下，适应度最优的结果。由于该问题较为复杂，无法判定角度和距离的占比，我们尝试调整不同的收益偏好系数，从而得到最优解^[5]。 K 为比例系数，其可将角度的余弦值和距离的数量级调整至相同。经过初步计算，可确定 K 的值为 10^4 。

经过计算可以取得适应度最大的点 d_{it} 作为当前最优解，即下一时刻的初始点 p_{t+1} ，之后再计算误差。

$$p_{t+1} \text{ 为垂直校正点时，误差为 } \begin{cases} x_{t+1} = 0 \\ y_{t+1} = y_t + l_{it}\delta \end{cases}$$

$$p_{t+1} \text{ 为水平校正点时，误差为 } \begin{cases} x_{t+1} = x_t + l_{it}\delta \\ y_{t+1} = 0 \end{cases}$$

5) 计算点 p_{t+1} 是否可以直接到达最终点 B ，若满足下列条件，则已抵达最终点 B ，输出点集 $W = (A, P_1, P_2, \dots, P_{h-1}, B)$ 的坐标依次连线作为航迹。否则，返回 2)。

$$\begin{cases} x_{t+1} + \delta l(p_{t+1}, B) < \theta \\ y_{t+1} + \delta l(p_{t+1}, B) < \theta \end{cases}$$

算法的流程图如图3所示：

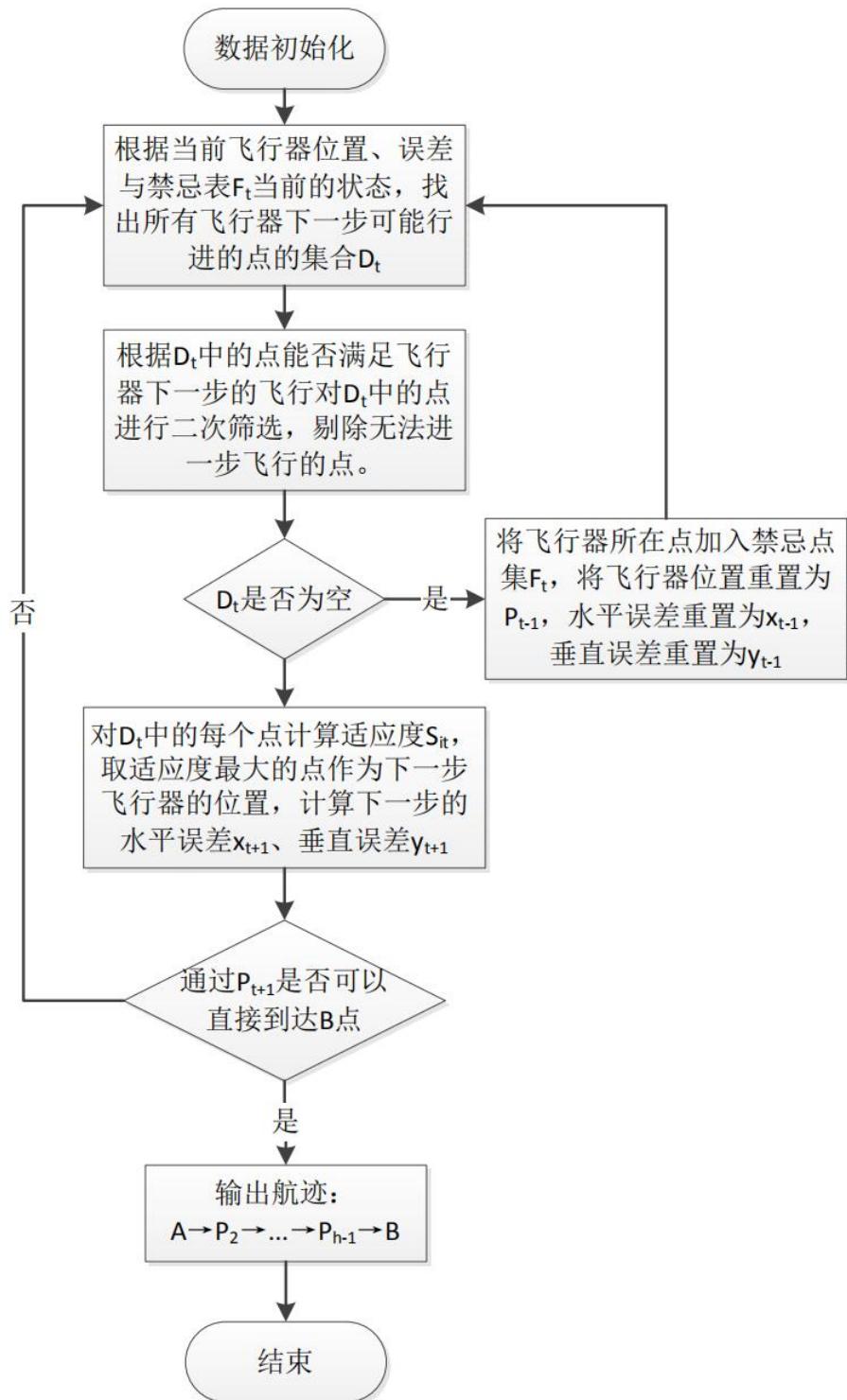


图 3 算法流程图

3.1.4 结果分析

以附件 1 中的数据作为初始条件，可以求得如图4所示的航迹图。从图中可以看出，所求出的航迹接近 A 与 B 两点的最短连线，经过计算可得其距离为 $106350.06m$ 。

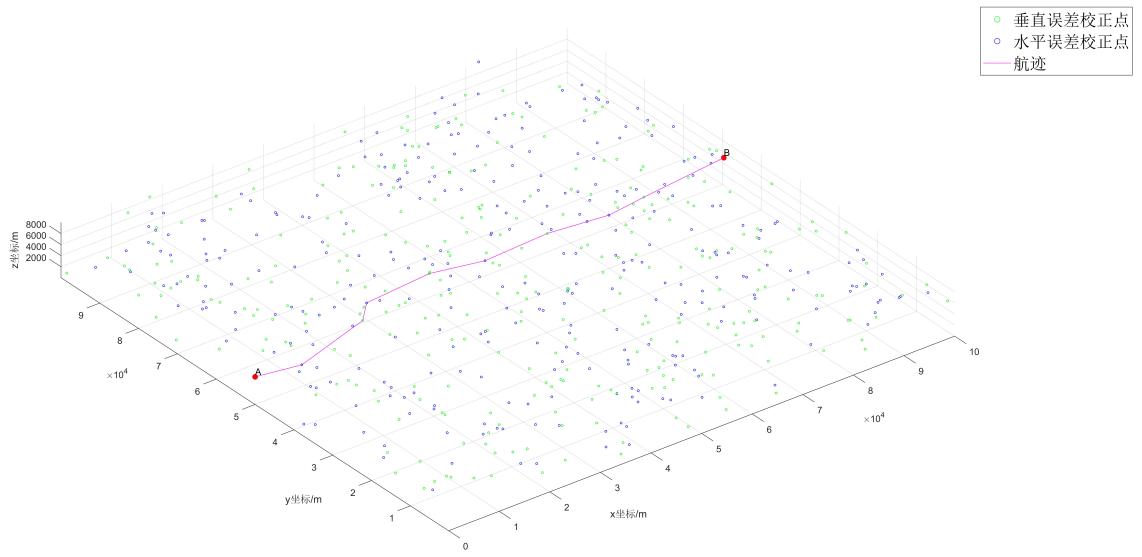


图4 附件1所求出的航迹图

经计算可得其航迹规划结果如表1所示。从表中可以看出所求得的所有水平校正点，即校正点类型为0，其均满足校正前的垂直误差不大于 $\beta_1 = 20$ 个单位，水平误差不大于 $\beta_2 = 25$ 个单位。所求得的所有垂直校正点，即校正点类型为1，其均满足校正前的垂直误差不大于 $\alpha_1 = 25$ 个单位，水平误差不大于 $\alpha_2 = 15$ 个单位。所需要的校正点(包含起始点与终止点)个数为10个，抵达终点时的水平误差和垂直误差均小于 $\theta = 30$ 个单位，满足所设定的终点约束条件。

表1 附件1航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
521	9.63	9.63	0
64	21.76	12.13	1
80	11.42	23.55	0
170	23.4	11.98	1
278	10.46	22.43	0
369	21.89	11.44	1
214	13.31	24.75	0
397	22.33	9.02	1
612	16.97	25.99	终点 B

以附件 2 中的数据作为初始条件，可以求得如图5所示的航迹图。从图中可以看出，所求出的航迹接近 A 与 B 两点的最短连线，经过计算可得其距离为 $111585.52m$ 。

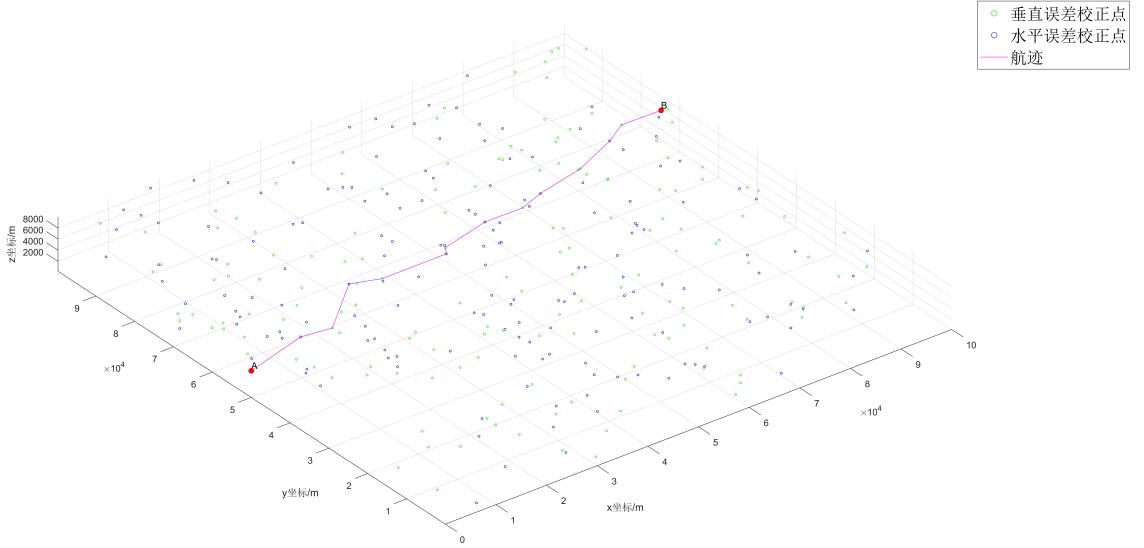


图 5 附件 2 所求出的航迹图

经计算可得其航迹规划结果如表3所示。从表中可以看出所求得的所有水平校正点，即校正点类型为 0，其均满足校正前的垂直误差不大于 $\beta_1 = 15$ 个单位，水平误差不大于 $\beta_2 = 20$ 个单位。所求得的所有垂直校正点，即校正点类型为 1，其均满足校正前的垂直误差不大于 $\alpha_1 = 20$ 个单位，水平误差不大于 $\alpha_2 = 10$ 个单位。所需要的校正点(包含起始点与终止点)个数为 14 个，抵达终点时的水平误差和垂直误差均小于 $\theta = 20$ 个单位，满足所设定的终点约束条件。

表 3 附件 2 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
163	13.29	13.29	0
114	18.62	5.33	1
8	13.92	19.26	0
309	19.45	5.52	1
121	11.25	16.78	0
123	16.6	5.35	1
49	11.79	17.14	0
160	18.3	6.51	1
92	5.78	12.29	0
93	15.26	9.48	1
61	9.83	19.32	0
292	16.39	6.55	1
326	6.96	13.51	终点 B

3.2 第二问建模与分析

3.2.1 问题分析

问题二是在问题一的基础上，增加了一个约束条件，即飞行器的最小转弯半径为 200m。为了寻求航迹长度最小的路径，需要考虑在两点间弧线与直线之间不同组合方式对距离大小的影响情况，从而得到一种最优的组合方式。

可沿用第一问的思想，对每一步决策寻找最优解，此时两个校正点间的距离不能简单地用欧式距离计算，需要考虑转向圆弧的影响。

3.2.2 模型建立

现定义如下参数：

建立飞行器从航迹中一点 P_t 到达另一点 P_{t+1} 的曲线最短路径的数学模型，全局目标函数与问题 1 中的目标函数相同，即

$$\begin{cases} \min\left(\sum_{t=1}^h l'_{P_t, P_{t+1}}\right) \\ \min(h) \end{cases}$$

约束条件为：

1) 航迹路线的基本约束:

$$\begin{cases} \sum_{i=1}^n c_{iP_t} = 1, & \forall t \in (1, n) \\ \sum_{j=1}^n c_{P_t j} = 1, & \forall t \in (1, n) \end{cases} \quad (9)$$

$$c_{ii} = 0, \quad \forall i \in (1, h) \quad (10)$$

$$\begin{cases} \sum_{i=1}^n c_{iP_1} = 0 \\ \sum_{j=1}^n c_{P_n j} = 0 \end{cases} \quad (11)$$

$$\begin{cases} P_1 = 0, & \text{起点为 A} \\ P_h = n, & \text{终点为 B} \end{cases} \quad (12)$$

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} = h - 1 \quad (13)$$

2) 保证航线经过点的垂直误差与水平误差均满足校正点约束:

$$\begin{cases} y_t + \delta l'_{P_t, P_{t+1}} \leq \alpha_1, & t = 1, 2 \dots h-2, \quad T_{t+1} = 1 \\ x_t + \delta l'_{P_t, P_{t+1}} \leq \alpha_2, & t = 1, 2 \dots h-2, \quad T_{t+1} = 1 \\ y_t + \delta l'_{P_t, P_{t+1}} \leq \beta_1, & t = 1, 2 \dots h-2, \quad T_{t+1} = 0 \\ x_t + \delta l'_{P_t, P_{t+1}} \leq \beta_2, & t = 1, 2 \dots h-2, \quad T_{t+1} = 0 \end{cases} \quad (14)$$

对于终点 B:

$$\begin{cases} y_{h-1} + \delta l'_{P_{h-1}, P_h} < \theta \\ x_{h-1} + \delta l'_{P_{h-1}, P_h} < \theta \end{cases} \quad (15)$$

3) 保证转弯半径最小值受到约束:

$$r_t \geq 200 \quad t = 1, 2 \dots h-1 \quad (16)$$

对于 P_t 到 P_{t+1} 的曲线路径规划问题, 关键在于如何确定转弯半径和转弯时机。因此我们需要讨论下列几种因素对路径的影响:

- 1) 转弯半径的大小: r_t 的大小对 $l'_{P_t, P_{t+1}}$ 的影响;
- 2) 转弯时机: 转弯时机不同, 航迹的形状也不同, 则需要考虑直线 + 曲线 + 直线和曲线 + 直线两种形式对 $l'_{P_t, P_{t+1}}$ 的影响。

3.2.3 算法求解

由于在初始点 A 可从任意方向前进，不受转向的影响，因此从初始点 A 到达第二个校正点时，选择走直线可以保证距离最小。之后再考虑转弯情况对路径长度的影响，下面讨论一下两定点之间带转弯弧度与直线之间距离最短的组合模型。

图6为定点 A 与定点 B 之间圆弧与直线的组合路径示意图。假设飞行器将从点 A 抵达点 B ，经过点 A 时的瞬时速度方向与直线 AB 的夹角为 φ ，若采用圆弧加直线的飞行方式，则飞行器飞行的距离可以视作圆弧 AC 与线段 CB 的长度之和。

由于存在多种影响距离的因素，因此需要分情况进行讨论。设飞行器到达点 A 时的速度矢量为 \vec{v} ，由立体几何知识可知，飞行器的运动轨迹一定在 \vec{v} 和 \overrightarrow{AB} 所构成的二维平面内。本文主要考虑两种决策变量：飞行器的转弯半径和飞行器的转弯时机。

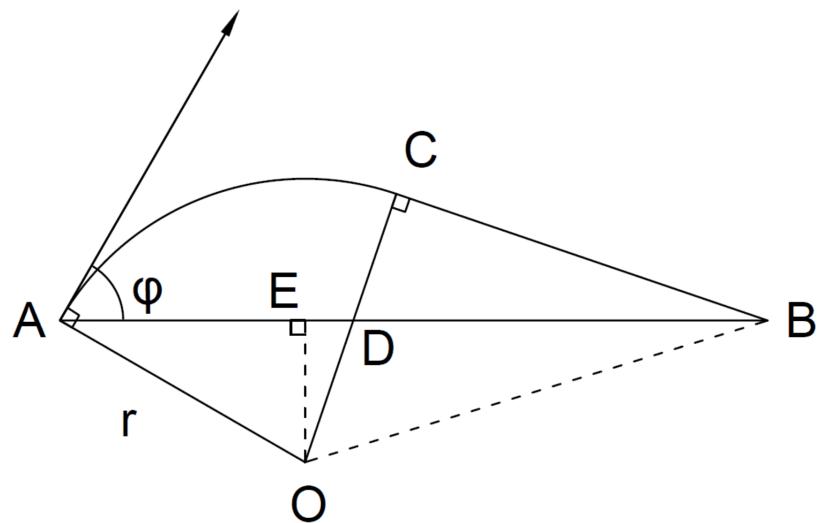


图 6 圆弧与直线组合路径示意图

首先讨论一种较为简单的情况，即飞行器到达点 A 时立即以半径 r 转弯，转弯至该处的切线经过点 B 时以直线路径达到点 B 。设 AB 的直线距离为 l ，则下列参数可通过公式求得：

$$AE = r \sin \varphi \quad (17)$$

$$OE = r \cos \varphi \quad (18)$$

$$BE = l - r \sin \varphi \quad (19)$$

$$OB = \sqrt{OE^2 + BE^2} = \sqrt{l^2 + r^2 - 2lr \sin \varphi} \quad (20)$$

$$BC = \sqrt{OB^2 - r^2} = \sqrt{l^2 - 2lr \sin \varphi} \quad (21)$$

$$\angle OBC = \arccos \sqrt{\frac{l^2 - 2lr \sin \varphi}{l^2 + r^2 - 2lr \sin \varphi}} \quad (22)$$

$$\angle OBE = \arccos \frac{l - r \sin \varphi}{\sqrt{l^2 + r^2 - 2lr \sin \varphi}} \quad (23)$$

$$\angle AOC = \varphi + \angle ABC$$

$$= \varphi + \arccos \sqrt{\frac{l^2 - 2lr \sin \varphi}{l^2 + r^2 - 2lr \sin \varphi}} - \arccos \frac{l - r \sin \varphi}{\sqrt{l^2 + r^2 - 2lr \sin \varphi}} \quad (24)$$

$$\begin{aligned} \widehat{AC} + BC &= r(\varphi + \arccos \sqrt{\frac{l^2 - 2lr \sin \varphi}{l^2 + r^2 - 2lr \sin \varphi}} - \arccos \frac{l - r \sin \varphi}{\sqrt{l^2 + r^2 - 2lr \sin \varphi}}) \\ &\quad + \sqrt{l^2 - 2lr \sin \varphi} \end{aligned} \quad (25)$$

由于式(25)中，所涉及的参数表达式过于复杂，难以求导。故我们希望采用计算机模拟的方法来对比不同转弯方式的优劣。在第一问中，通过对最优解的计算，可得到附件1对应的航迹各校正点间的平均距离为11816.67m，飞行器每次转弯的平均角度为27.89°。附件2对应的各校正点间的平均距离为8583.50m，飞行器每次转弯的平均角度为36.67°。在接下来中，为了简便起见，预设飞行器在相邻两个航迹点间的飞行距离为10000m，转弯角度为30°，可认为其比较符合实际情况。

首先分析转弯半径的影响，如图7所示。假设飞行器同时在到达点A的瞬间开始转弯，但转弯半径r不同。设l=10000, φ=30°，从200至5000等间隔地取值，将其带入式(25)中进行计算，得到不同r值下的 $\widehat{AC} + BC$ 值，如图8所示。从图中可以看出，随着转弯半径不断从200增大至5000，所经过的航迹长度越长，飞行距离也越远。因此，若在A点瞬时转弯，且以转弯圆的切线形式抵达B点时，转弯半径越小，则航迹路线越短。在此类情况下，转弯半径应取最小值200m。

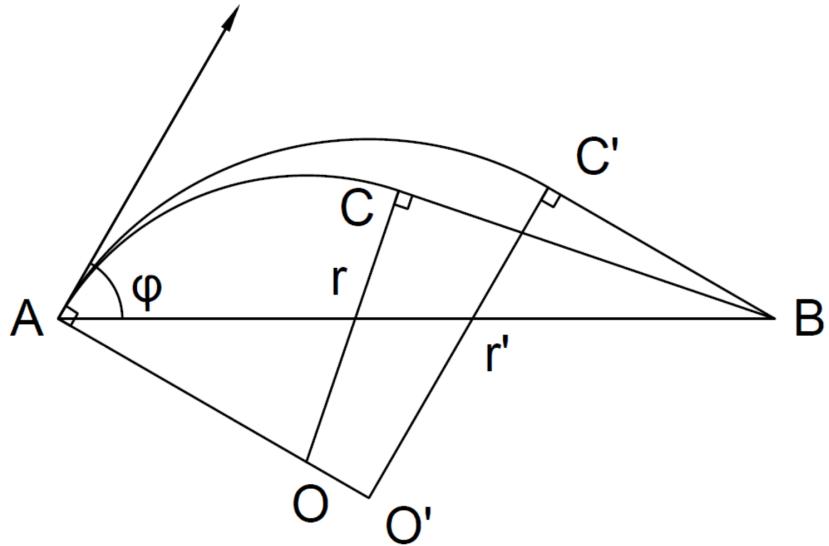


图 7 圆弧与直线组合路径示意图

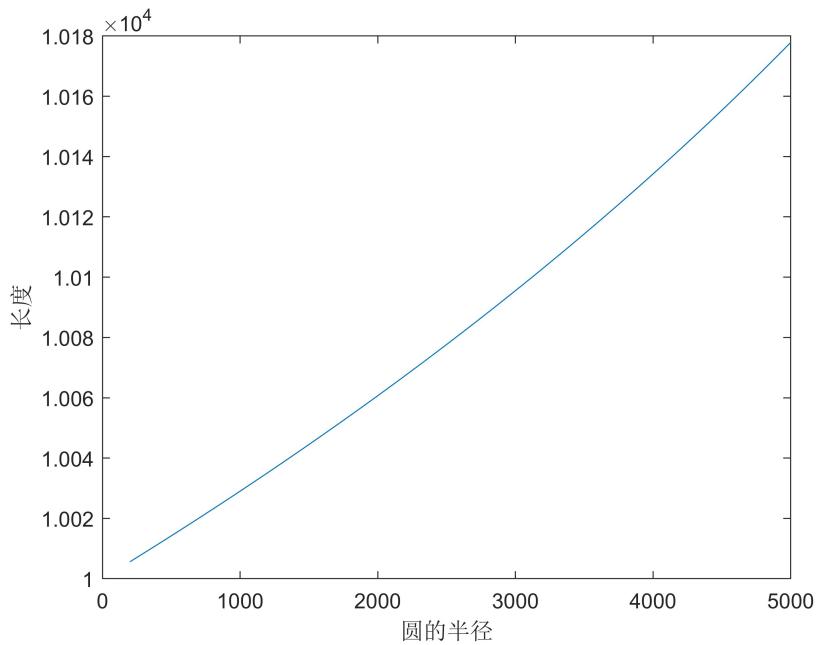


图 8 路径长度随转弯半径的变化图

其次讨论转弯点的位置，即飞行器该在从出发点直线飞行多远处开始转弯。如图9所示，路径 ABC 代表飞行器在点 A 立刻转弯， $AMNB$ 代表飞行器在飞行一段距离之后再转弯，其中 $r = r'$ 。延长 BC 与点 A 的速度矢量的延长线交于点 G ，延长 BN 与点 A 的速度矢量的延长线交于点 G' 。设 $\angle AGB = \omega$, $\angle AG'B = \omega'$ ，飞行器飞行路径长度为 $AM + \widehat{MN} + NB$ 。

$$\widehat{MN} = r(\pi - \omega') \quad (26)$$

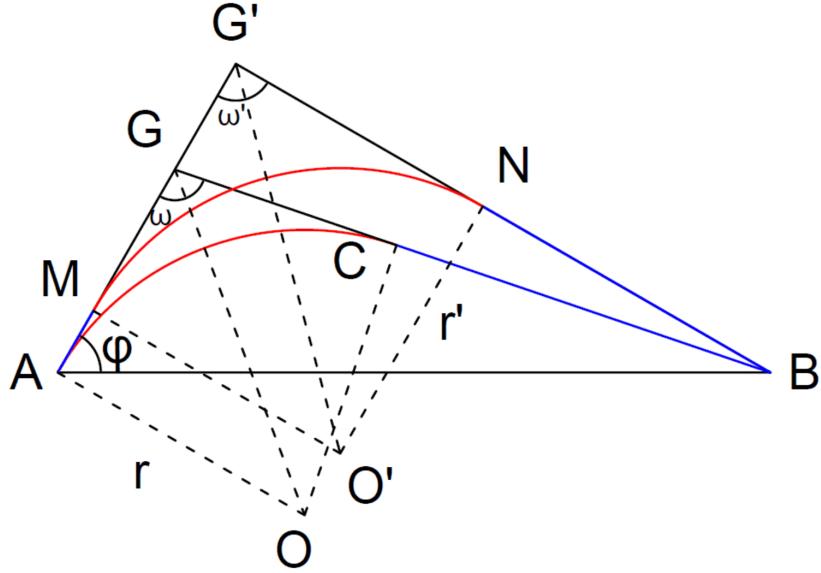


图 9 圆弧与直线组合路径示意图

由正弦定理有:

$$\frac{l}{\sin \omega'} = \frac{AG'}{\sin(\pi - \omega' - \varphi)} = \frac{BG'}{\sin \varphi} \quad (27)$$

经计算可得:

$$AG' = l \frac{\sin(\omega' + \varphi)}{\sin \omega'} \quad (28)$$

$$BG' = l \frac{\sin \varphi}{\sin \omega'} \quad (29)$$

$$MG' = NG' = r \tan \frac{\pi - \omega'}{2} \quad (30)$$

$$AM = AG' - MG' = l \frac{\sin(\omega' + \varphi)}{\sin \omega'} - r \tan \frac{\pi - \omega'}{2} \quad (31)$$

$$AM + \widehat{MN} + NB = r(\pi - \omega') + l \frac{\sin(\omega' + \varphi) + \sin \varphi}{\sin \omega'} - 2r \tan \frac{\pi - \omega'}{2} \quad (32)$$

由于式(32)中, 所涉及的参数表达式过于复杂, 难以求导。故依旧采用计算机模拟的方法来求解其最优飞行路线。由于在分析转弯半径时, 已经求解出转弯半径取最小值 200 时可以求得最小路径。因此, 本次模拟时预设 AB 间的距离为 10000m, 转弯角度为 30° , 转弯半径设定为最小半径 200m。

本次模拟的变量为 ω' , 其可以用来判定转弯的位置。通过几何关系易知, 当飞行器越晚开始转弯, 则其所对应的 ω' 值便越小。设 $l = 10000$, $\varphi = 30^\circ$, $r = 200m$, ω' 从 0 至 $\frac{5\pi}{6}$ 等间隔地取值, 将其带入式(32)中进行计算, 可得到航迹总长度。航迹总长度随 ω' 的值的变化情况, 如图10所示。从图中可以看出, 随着 ω' 的增大, 其所经过的航迹长度逐渐变小。因此, 若从点 A 以固定的转弯半径转弯, 且

以转弯圆的切线形式抵达 B 点时，越早开始转弯，其航迹路线越短。故在此类情况下，转弯点应取飞行器刚好抵达的点 A 。

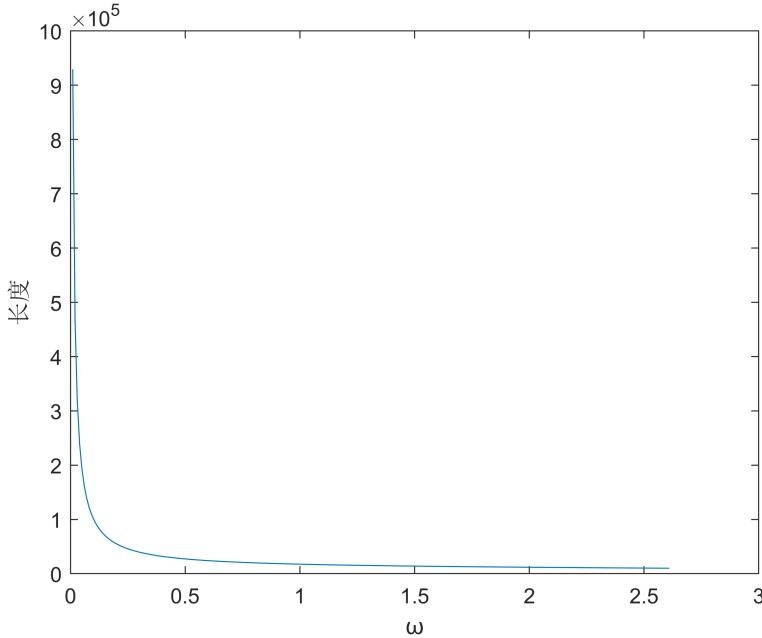


图 10 路径长度随 ω' 的变化图

之后再采用贪心算法对其进行求解，即依据当前飞行器的位置求解出下一时刻飞行器的位置，得到当前时刻的最优节点。具体的算法思路如下：

1) 首先初始化飞行器的位置 P_0 和当前的水平误差 x_0 与垂直误差 y_0 。飞行器位置 P_0 为 A 点位置，误差均为 0。由于飞行器在初始时刻速度矢量可以任意选择，规定飞行器从 A 点飞往其他点时，速度矢量为 A 点到该点的矢量方向，即飞行器从 A 点到其他点的路径必然是直线。

2) 之后再根据当前校正点的位置、误差与速度矢量，计算当前点到其他所有点计算转弯距离后的距离 $l_{P_t, d_{it}}$ ，找出所有满足如下要求的校正点，并将其定义为 D_t 。

$$\begin{cases} \delta \cdot l_{P_t, d_{it}} + y_t \leq \alpha_1, \delta \cdot l_{P_t, d_{it}} + x_t \leq \alpha_2 & , \text{该点为垂直误差校正点} \\ \delta \cdot l_{P_t, d_{it}} + y_t \leq \beta_1, \delta \cdot l_{P_t, d_{it}} + x_t \leq \beta_2 & , \text{该点为水平误差校正点} \end{cases}$$

并根据问题 1 中的方法，对 D_t 进行二次筛选。

3) 根据问题 1 中的方法，计算禁忌点集 F_t 。

4) 根据问题 1 中的方法，计算求解 D_t 中所有校正点 d_{it} 中的适应度：

$$S_{it} = \frac{K\rho \cos \gamma_{it} + (1 - \rho)l'_{P_t, d_{it}}}{Z_{it}}$$

其中 l'_{P_t, d_t} 为根据公式25计算的由点 P_t 到点 D_t 之间考虑转弯得到的最短距离。适应度最大的校正点即为当前最优解。

5) 计算点 p_{t+1} 是否可以直接到达最终点 B , 若能, 则已抵达最终点 B , 输出点集 $W = (A, P_1, P_2, \dots, P_{h-1}, B)$ 的坐标依次连线作为航迹。否则, 返回 2)。算法的流程图如图11所示:

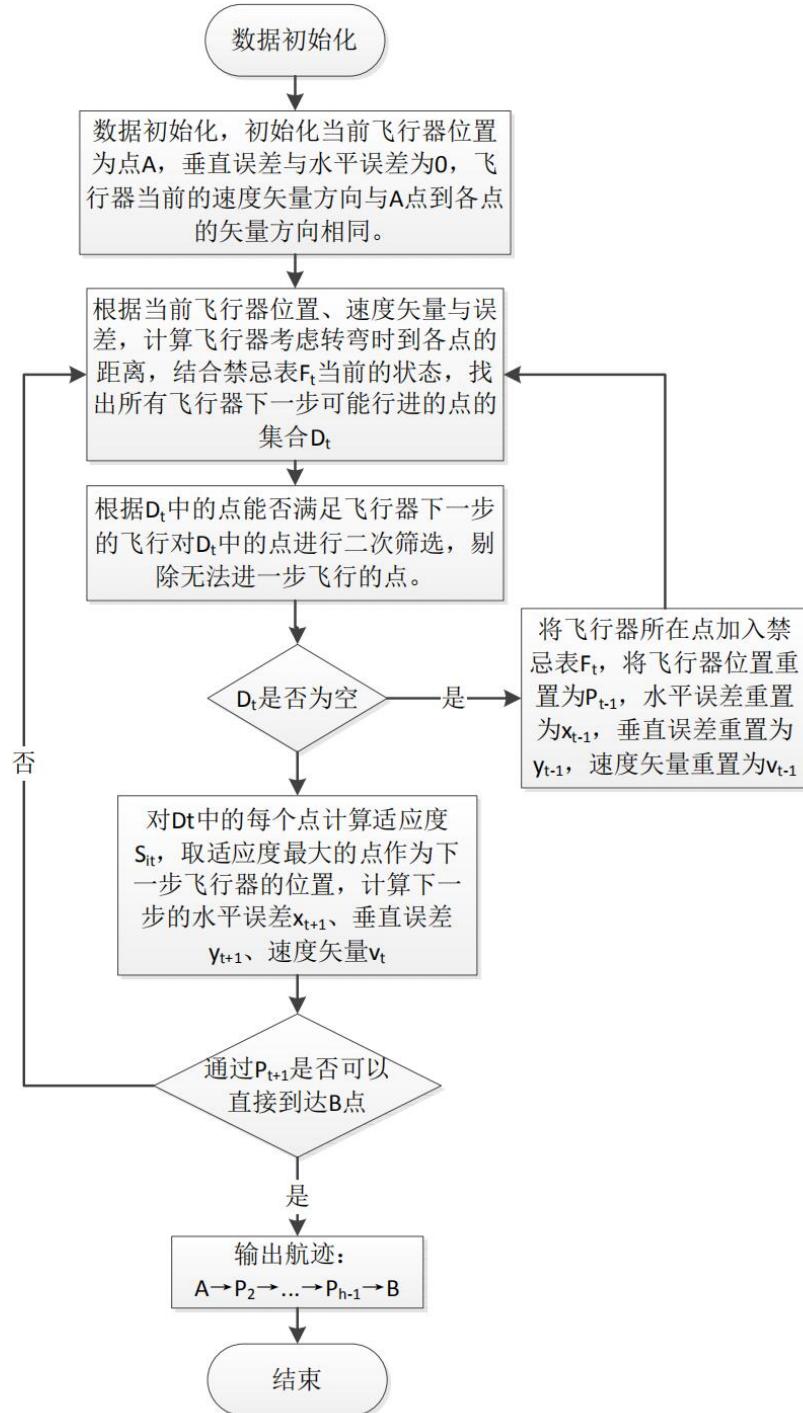


图 11 算法流程图

3.2.4 结果分析

以附件 1 中的数据作为初始条件，可得到航迹图如图12所示。从图中可以看出，所求航迹接近 A 与 B 两点的最短连线，经过计算可得其距离为 106440.33m。

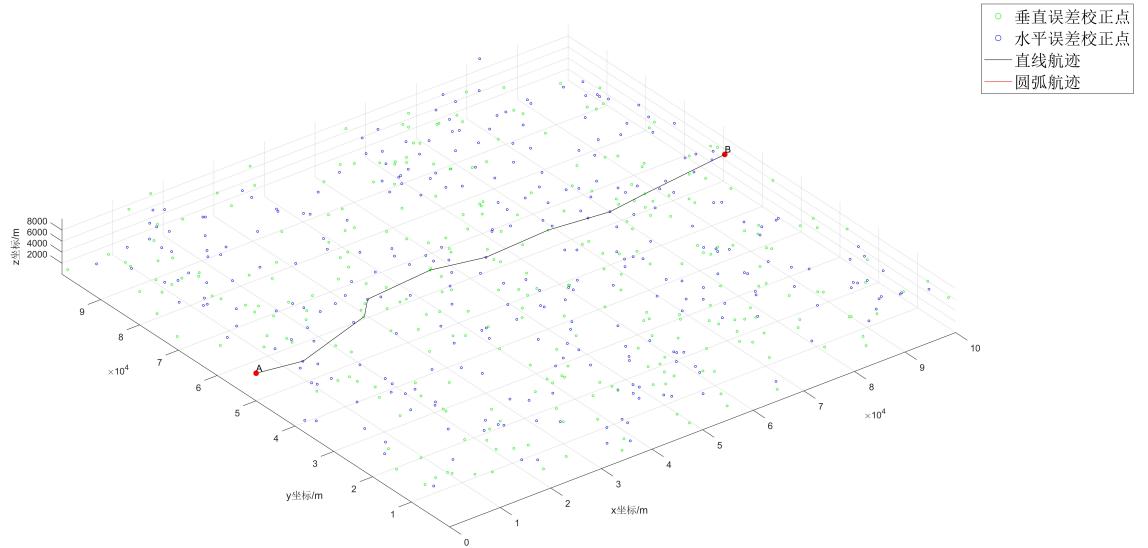


图 12 附件 1 所求出的航迹图

由于圆弧路线相较于直线路线而言，其距离过于短，因此在图12中几乎观测不到红色的圆弧路线。为了能够观测到圆弧路径，我们对局部进行了放大处理，如图13所示。从图中可看出红色圆弧路径相较于直线路线而言，长度较短。通过计算可知，所求得的两个相邻校正点间的平均距离约为 10000m。转弯半径为 200m，是平均距离的 $\frac{1}{50}$ ，故而圆弧距离占两点之间距离的比例较小，在宏观图上难以观测到。

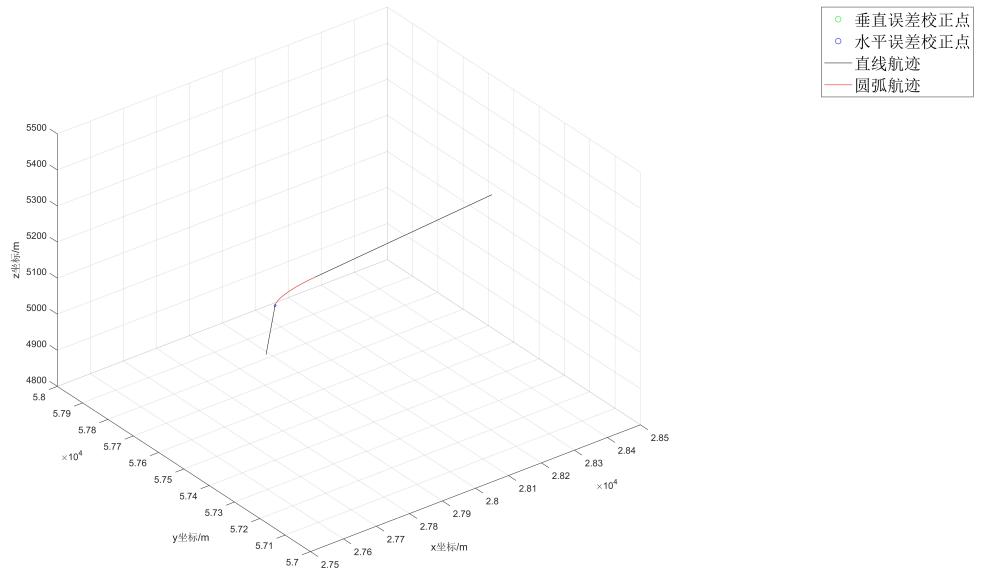


图 13 附件 1 所求出的航迹局部细节图

经计算可得其航迹规划结果如表5所示，所需要的校正点(包含起始点与终止点)个数为 10 个。从表中可以看出所求得的所有水平校正点，即校正点类型为 0 的点，其均满足校正前的垂直误差不大于 $\beta_1 = 20$ 个单位，水平误差不大于 $\beta_2 = 25$ 个单位。所求得的所有垂直校正点，即校正点类型为 1，其均满足校正前的垂直误差不大于 $\alpha_1 = 25$ 个单位，水平误差不大于 $\alpha_2 = 15$ 个单位。抵达终点 B 时的水平误差和垂直误差均小于 $\theta = 30$ 个单位，满足所设定的终点约束条件。

表 5 附件 1 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
521	9.63	9.63	0
64	21.76	12.13	1
80	11.47	23.6	0
170	23.48	12.01	1
278	10.46	22.47	0
369	21.9	11.44	1
214	13.32	24.76	0
397	22.34	9.02	1
612	16.97	25.99	终点 B

以附件 2 中的数据作为初始条件，可以求得如图14所示的航迹图。从图中可以看出，所求出的航迹接近 A 与 B 两点的最短连线，经过计算可得其距离为 $111780.13m$ 。

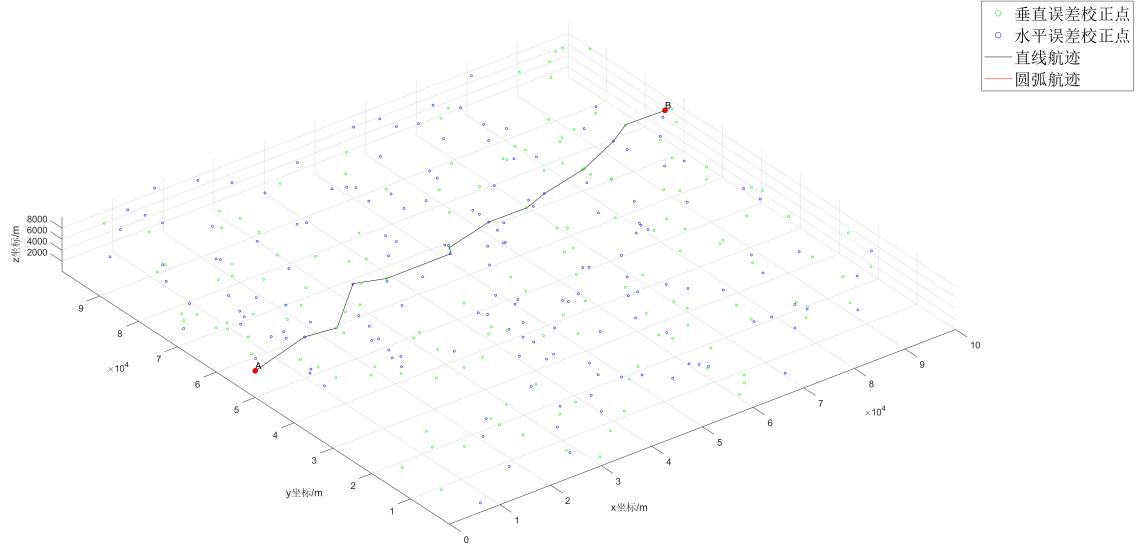


图 14 附件 2 所求出的航迹图

经计算可得其航迹规划结果如表7所示，所需要的校正点(包含起始点与终止点)个数为 14 个。从表中可以看出所求得的所有水平校正点，即校正点类型为的点，其均满足校正前的垂直误差不大于 $\beta_1 = 15$ 个单位，水平误差不大于 $\beta_2 = 20$ 个单位。所求得的所有垂直校正点，即校正点类型为 1，其均满足校正前的垂直误差不大于 $\alpha_1 = 20$ 个单位，水平误差不大于 $\alpha_2 = 10$ 个单位。抵达终点 B 时的水平误差和垂直误差均小于 $\theta = 20$ 个单位，满足所设定的终点约束条件。

表 7 附件 2 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
163	13.29	13.29	0
114	18.63	5.34	1
8	13.96	19.3	0
309	19.52	5.56	1
121	11.25	16.81	0
123	16.66	5.41	1
49	11.8	17.21	0
160	18.32	6.52	1
92	5.79	12.31	0
93	15.28	9.49	1
61	9.83	19.32	0
292	16.38	6.55	1
326	6.96	13.51	终点 B

3.3 第三问建模与分析

3.3.1 问题分析

在第三问中，加入了不确定性因素，即部分校正点有一定概率出现校正失败的情况。由于校正点可能出现失败的情况，若继续采用贪心算法求解，则无法求得飞行器在校正点处的准确误差。因此无法利用当前校正点的误差，求解下一时刻最优校正点。针对此种情况，我们采用问题 1 中的解作为初始解，通过禁忌搜索法对初始解进行优化，最终得到满意解。

3.3.2 模型建立

目标函数：

$$\begin{cases} \max(\text{success}) \\ \min\left(\sum_{i=1}^n \sum_{j=1}^n (c_{ij} * l_{ij})\right) \\ \min(h) \end{cases} \quad (33)$$

约束条件：

1) 航迹路线的基本约束：

$$\begin{cases} \sum_{i=1}^n c_{iP_t} = 1, & \forall t \in (1, n) \\ \sum_{j=1}^n c_{P_t j} = 1, & \forall t \in (1, n) \end{cases} \quad (34)$$

$$c_{ii} = 0, \quad \forall i \in (1, h) \quad (35)$$

$$\begin{cases} \sum_{i=1}^n c_{iP_1} = 0 \\ \sum_{j=1}^n c_{P_n j} = 0 \end{cases} \quad (36)$$

$$\begin{cases} P_1 = 0, & \text{起点为 A} \\ P_h = n, & \text{终点为 B} \end{cases} \quad (37)$$

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} = h - 1 \quad (38)$$

2) 保证航线经过点的垂直误差与水平误差均满足校正点约束:

$$\begin{cases} y_t + \delta l_{P_t, P_{t+1}} \leq \alpha_1 & t = 1, 2, \dots, h-2 \quad T_{t+1} = 1 \\ x_t + \delta l_{P_t, P_{t+1}} \leq \alpha_2 & t = 1, 2, \dots, h-2 \quad T_{t+1} = 1 \\ y_t + \delta l_{P_t, P_{t+1}} \leq \beta_1 & t = 1, 2, \dots, h-2 \quad T_{t+1} = 0 \\ x_t + \delta l_{P_t, P_{t+1}} \leq \beta_2 & t = 1, 2, \dots, h-2 \quad T_{t+1} = 0 \end{cases} \quad (39)$$

对于终点 B:

$$\begin{cases} y_{h-1} + \delta l_{P_{h-1}, P_h} < \theta \\ x_{h-1} + \delta l_{P_{h-1}, P_h} < \theta \end{cases} \quad (40)$$

3) 所有状态的概率之和为 1

$$\sum_{j=1}^{2^m} P(q_j) = 1 \quad (41)$$

4) 成功率的约束

$$success = \sum_{j=1}^{2^m} (g_j P(q_j)) \quad (42)$$

3.3.3 算法求解

首先我们采用穷举法计算所得解的成功率。对于任意一个解的点集 $W = (A, P_1, P_2, \dots, P_{h-1}, B)$ 而言，可找到其中所有可能出现校正失败的点，记为集合 U ，集合 U 所包含的元素个数为 m 。之后以 0-1 变量 u_i 来表示第 i 点是否校正失败，如下所示：

$$u_i = \begin{cases} 1, & \text{该点校正失败} \\ 0, & \text{该点校正成功} \end{cases}, i = (1, 2, \dots, m)$$

由题意可知：

$$P(u_i) = \begin{cases} 0.2, & u_i = 1 \\ 0.8, & u_i = 0 \end{cases}$$

对于点集 U 中每一个校正点而言，其最终只有两种情况，校正成功或校正失败。由于校正点的排列位置在点集 U 中是固定的，因此包含 m 个点的点集 U 共有 2^m 种校正情况。现假定每种情况的集合为 Q ， q_j 表示其中第 j 种状态，($j = 1, 2, \dots, 2^m$)。则 q_j 中包含 i 个点，其状态定义为 $u_{1j}, \dots, u_{ij}, \dots, u_{mj}$ 。根据题意，每一点每种状态的概率如下：

$$P(u_{ij}) = \begin{cases} 0.2, & u_{ij} = 1 \\ 0.8, & u_{ij} = 0 \end{cases}$$

则有每一种情况的概率为 $P(q_j) = \prod_{i=1}^m P(u_{ij})$ 。设置 0-1 变量 g_j ，其用以表示在集合 Q 中的航迹是否能成功航行。当 $g_j = 1$ 时，表示该航迹能成功航行，当 $g_j = 0$ 时，则表明其无法成功航行。故对于一条航迹而言，总的成功航行概率为 $success = \sum_{j=1}^{2^m} P(q_j)g_j$ 。

计算每个解的成功率后，对航迹中所经过的每个可能失败的点定义危险度 a_i 。危险度为航迹失败情况下， u_i 失败的概率。 a_i 越大，则该点校正失败越容易引发航迹失败的情况，该点越危险。

定义事件 A 为点 u_i 失败，事件 B 为航迹路线运行失败。由条件概率公式有： $a_i = P(A|B) = \frac{P(A \cap B)}{P(B)}$ 。定义状态集合 Q'_i 为点 u_i 校正失败且航迹失败的情况集合，则可以得到：

$$P(A \cap B) = \sum_{q \in Q'_i} P(q)$$

$$P(B) = 1 - success$$

则

$$a_i = \frac{P(A \cap B)}{P(B)} = \frac{\sum_{q \in Q'_i} P(q)}{1 - success}$$

将 U 中元素按照危险度大小排序，根据初始解和排序后的集合 U' 进行禁忌搜索。

禁忌搜索算法的原则为：

1) 决策原则: 代价减少大且非禁忌的移动优先采用。若最优候选解对应的移动不在禁忌范围内, 则当前的选择为采纳该移动^[6]。

2) 蔑视原则: 若禁忌对象移动对应的代价减少值大于“best so far”状态, 则无视其禁忌属性而仍将其作为为当前选择^[7]。这样可以不失掉问题的优化解。

选择某个移动后, 应将禁忌表中所有移动的非 0 禁忌值均减少 1, 并将当前选择的移动所对应的禁忌表元素设定为禁忌长度值。

禁忌搜索算法沿用局部领域搜索的思想, 具有灵活的记忆功能和藐视准则。其在搜索过程中可以接受劣解, 所以具有较强的“爬山”能力, 其搜索时能够突跳出局部最优解, 从而增强获得更好的全局最优解的概率^[8]。

(1) 初始解的生成:

初始解选用问题 1 得到的最优解, 对解中的危险点计算危险度并排序。

(2) 搜索规则:

从危险度最大的点开始, 在所有点中寻找可替换点。若替换后, 在所有点不失败的情况下航迹可行, 可将该点与危险度最大点交换。计算交换前后解的成功率 $success_{\text{前}}$ 和 $success_{\text{后}}$ 以及交换前后航迹的总长度 $length_{\text{前}}$ 和 $length_{\text{后}}$, 以 $\frac{K}{success} + length$ 为指标, 若 $\frac{K}{success_{\text{前}}} + length_{\text{前}} > \frac{K}{success_{\text{后}}} + length_{\text{后}}$ 则接受新解。在本题中, 根据现实情况, 我们认为航迹失败一次的代价大致为飞行器飞行一次的路程代价。为了消除数量级的影响, 我们设置比例系数 $K = 10^5$ 。若没有点可以置换危险度最大的点, 可依次排序替换危险度次大的点。

(3) 禁忌表和藐视规则:

当算法进行一次交换操作, 即以校正点 i 替换校正点 j 。禁忌表长度为 100, 即在 100 次迭代以内不能进行相同的操作。当处于禁忌表中的操作执行时, 可以获得比历史最优解更好的解, 该操作不受禁忌表限制。每次迭代后, 每项操作的禁忌表长度减一。

(4) 结束条件:

当迭代达到最大迭代次数 1000 或没有可以替换的点后, 则结束搜索, 接受历史最优解。

(5) 模拟退火思想与解的松弛:

实际利用应用问题 1 所得的解进行搜索时, 发现解的成功率过低。分析可以发现, 问题 1 中的解中所用校正点太少, 导致解过于紧张, 可供替换点少, 所以我们对问题 1 中的解创新性的进行“松弛”操作, 随机选取可行的点插入到解中, 使每个危险点上可供替换的点增多, 加大搜索空间。另外, 针对搜索空间过小问题, 我们采用模拟退火思想, 即当新解差于当前解时, 以 $P = 2 \times 10^{-4}(1000 - iter)$ 的概率接受新解。在算法运行的前期接受较差解的概率大, 拓宽搜索空间; 在算法运行的后期接受较差解的概率小, 使解优化更完善。

算法的流程图如图15所示：

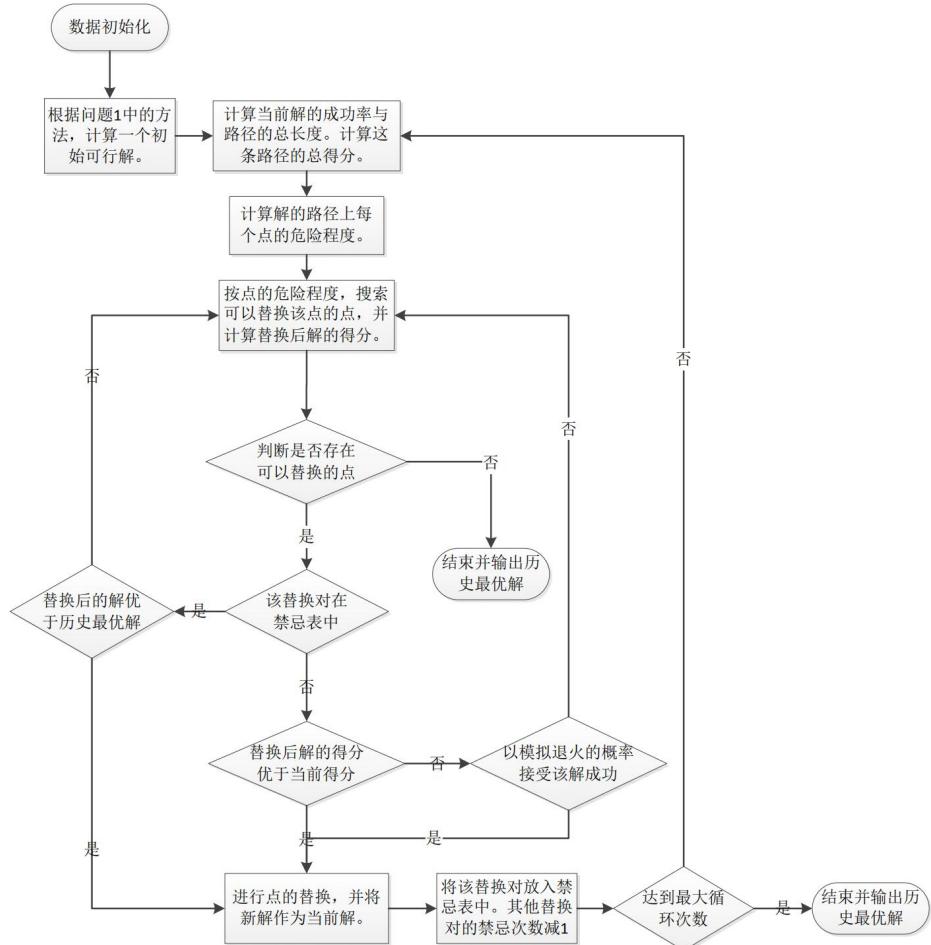


图 15 算法流程图

3.3.4 结果分析

为了能够更加清晰直观地观测航迹规划区中可能发生校正失败的点，即异常点的分布情况，我们在问题 1 中求解出来的航迹图里增添了异常点的标记，并以小三角形标识符来表示。如图16和图17所示。从图中可以看出在问题 1 中所求得附件 1 的航迹路线中，包含 4 个可能失效的异常点。

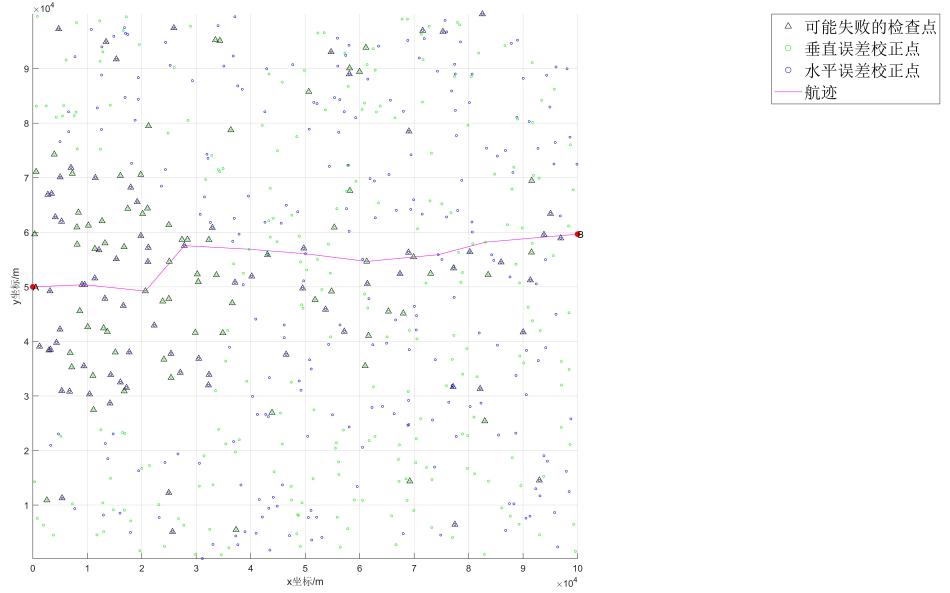


图 16 附件 1 所求出的第一问中航迹俯视图

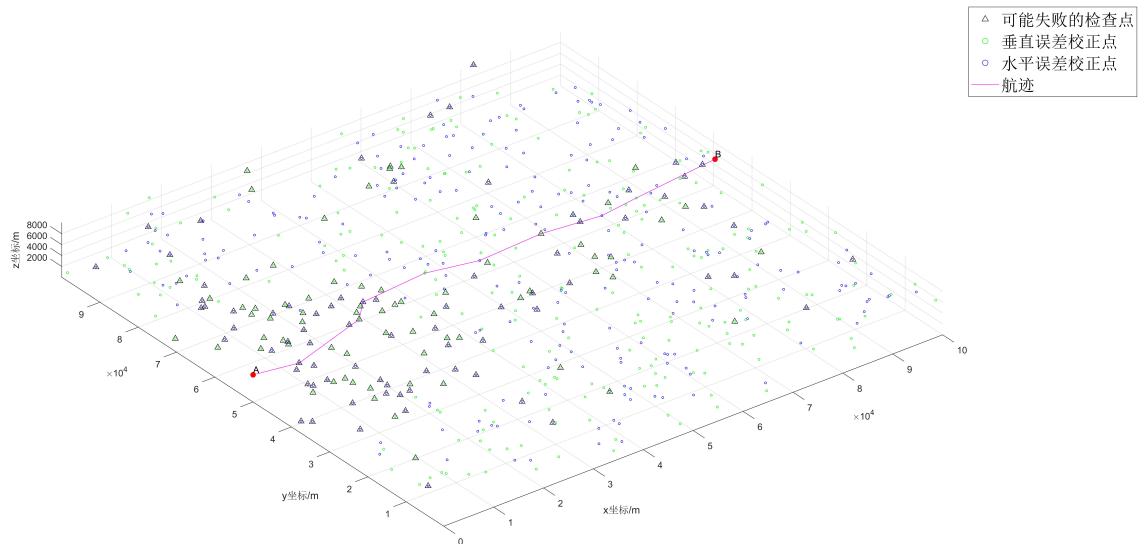


图 17 附件 1 所求出的第一问中的航迹侧视图

求解前, 问题 1 中所求得的路径的成功率为 40.96%, 经过禁忌搜索算法求解后, 可得到成功率 100% 的航迹路线。航迹图如图 18 和图 19 所示。从图中可以看出, 所求航迹接近 A 与 B 两点的最短连线, 经过计算可得其距离为 110051.66m。

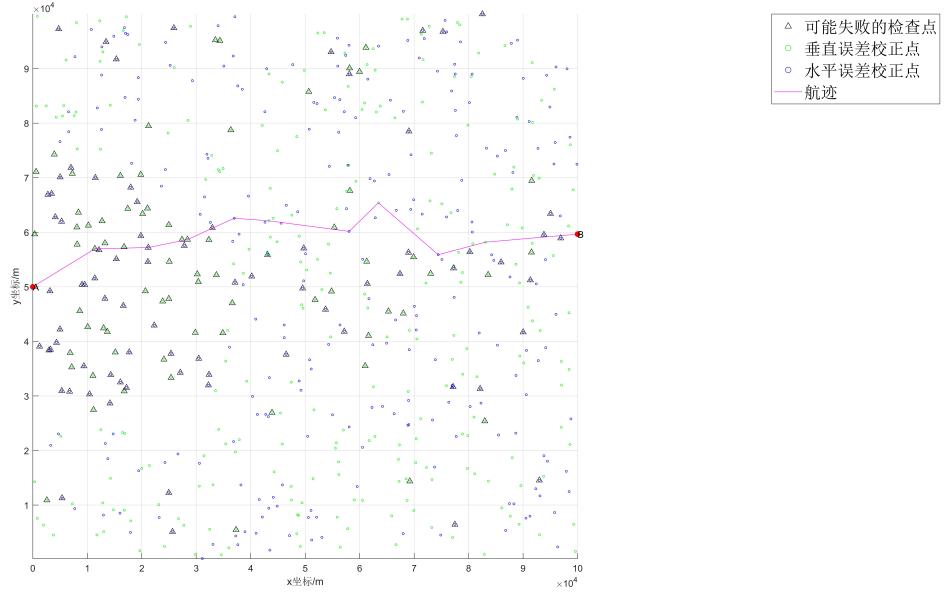


图 18 附件 1 所求出的第三问的航迹俯视图

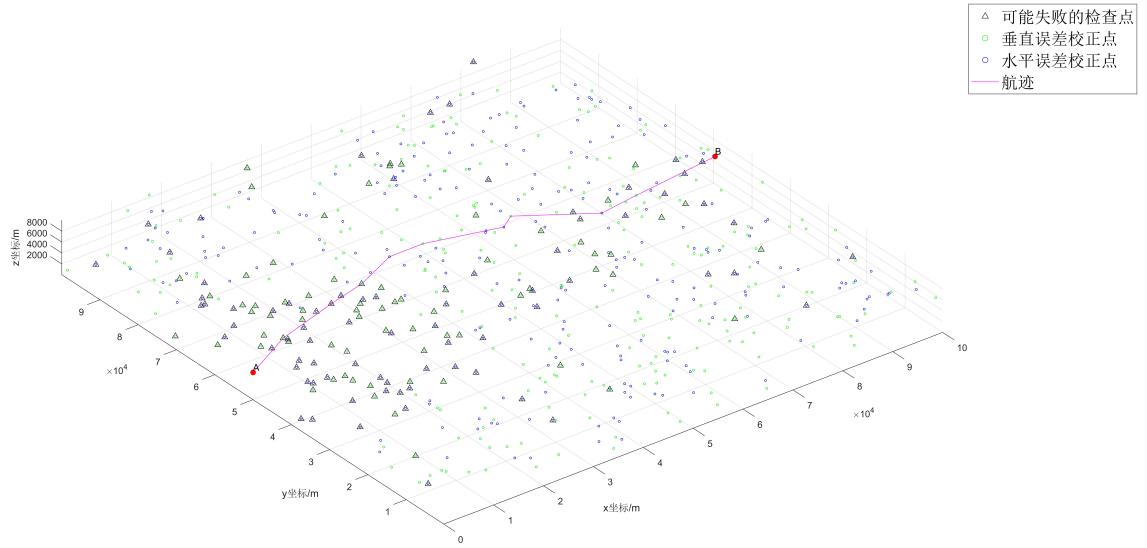


图 19 附件 1 所求出的第三问的航迹侧视图

经计算可得其航迹规划结果如表9所示，所需要的校正点(包含起始点与终止点)个数为 11 个，异常点为 3 个。从表中可以看出所求得的所有水平校正点，即校正点类型为 01 和 02 的点，其均满足校正前的垂直误差不大于 $\beta_1 = 20$ 个单位，水平误差不大于 $\beta_2 = 25$ 个单位。所求得的所有垂直校正点，即校正点类型为 11 和 12 的点，其均满足校正前的垂直误差不大于 $\alpha_1 = 25$ 个单位，水平误差

不大于 $\alpha_2 = 15$ 个单位。抵达终点 B 时的水平误差和垂直误差均小于 $\theta = 30$ 个单位，满足所设定的终点约束条件。

表 9 附件 1 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
503	13.39	13.39	12
294	15.18	23.57	01
91	22.54	7.35	12
75	14.44	16.8	01
524	20.81	6.37	11
11	14.92	21.29	01
450	22.77	7.85	11
214	14.56	22.4	01
397	23.57	9.02	11
612	16.97	25.99	终点 B

同样，我们先观测附件 2 中校正点中包含异常点的分布情况，在第一问求出的附件 2 航迹图上增加了异常点。航迹图如图20和图21所示。从图中可以看出在问题 1 中所求得的附件 2 航迹路线中，包含 10 个可能失效的异常点。侧视图中可看出，在航迹路线周围存在较多的异常点。

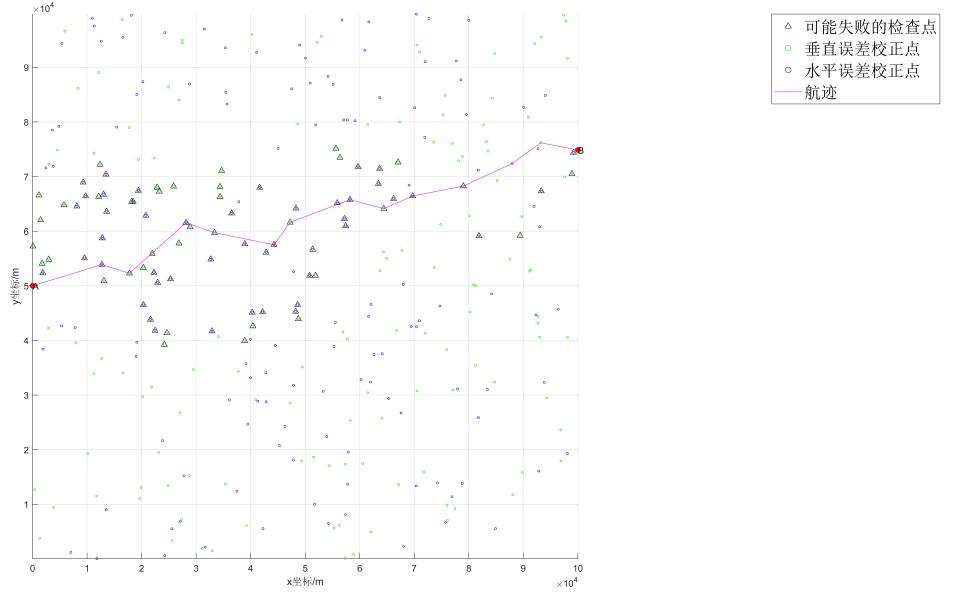


图 20 附件 2 所求出的第一问中航迹俯视图

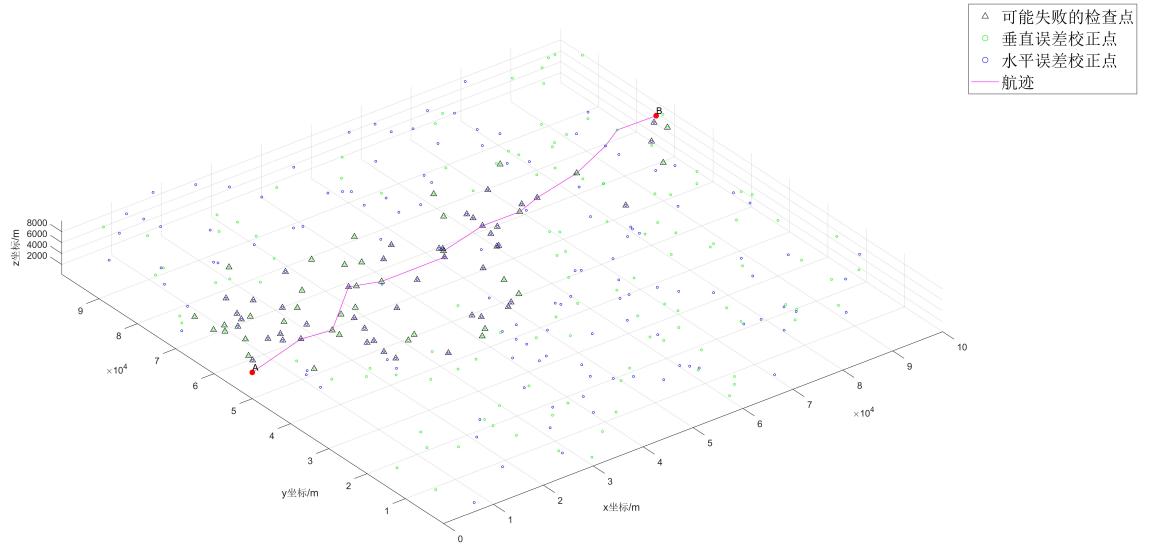


图 21 附件 2 所求出的第一问中的航迹侧视图

首先，我们采用问题 1 中所求得的附件 2 的解计算成功率，发现其十分低，仅为 10.74%。经过分析，我们发现是由于航迹经过了一段异常点密度较高的区域，导致航迹成功率一直较低，难以跳出局部最优解。因此我们在贪心算法的基础上，将异常点的适应度调低，使得航迹绕过异常点密集的区域，重新得到一组成功率较高的点集作为改进初始解。求得改进过后的初始解成功率为 51.20%，所经过

的校正点(包含起始点与终止点)的个数为 22 个, 航迹路线长度为 165026.79m。其航迹图如图22和图23所示。

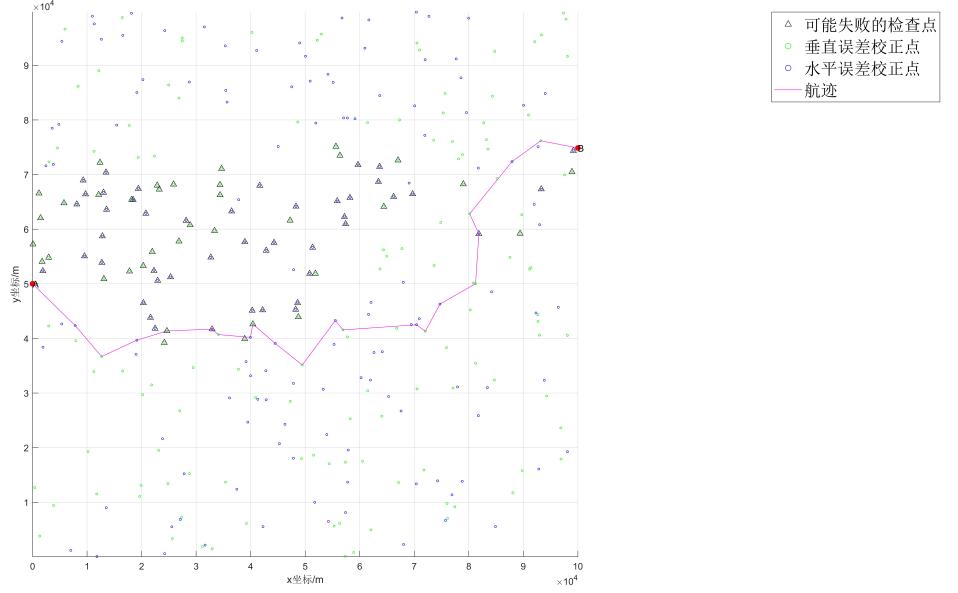


图 22 附件 2 所求出的改进初始解航迹俯视图

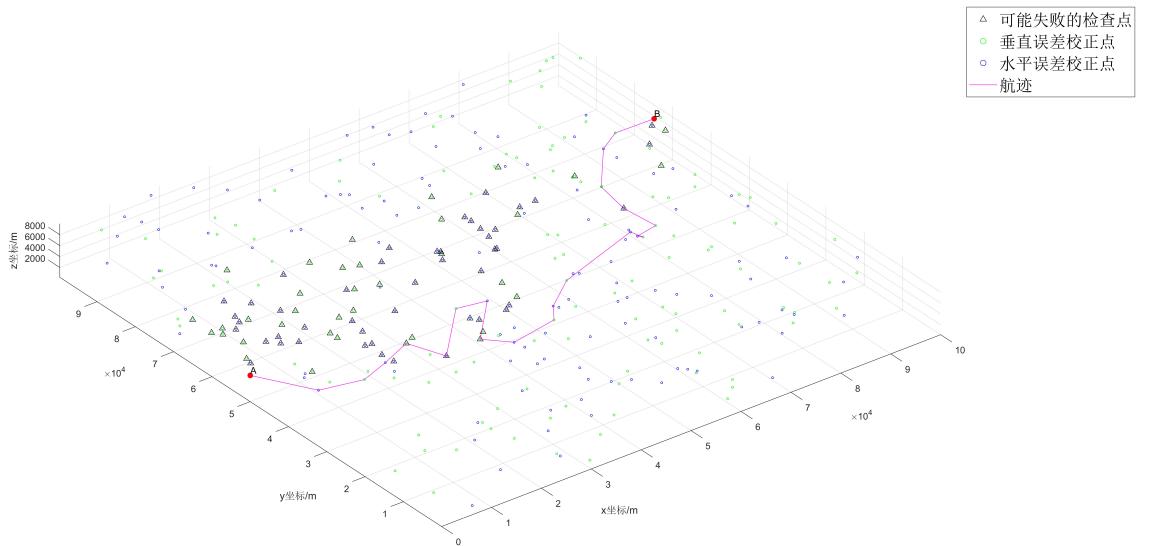


图 23 附件 2 所求出的改进初始解航迹侧视图

将上述改进过后的解作为初始解, 采用禁忌搜索法可求得一条新的航迹。该航迹的成功率为 76.8%, 所经过的校正点(包含起始点与终止点)的个数为 24 个, 航迹路线长度为 156679.78m。所得航迹图如图24和图25所示。

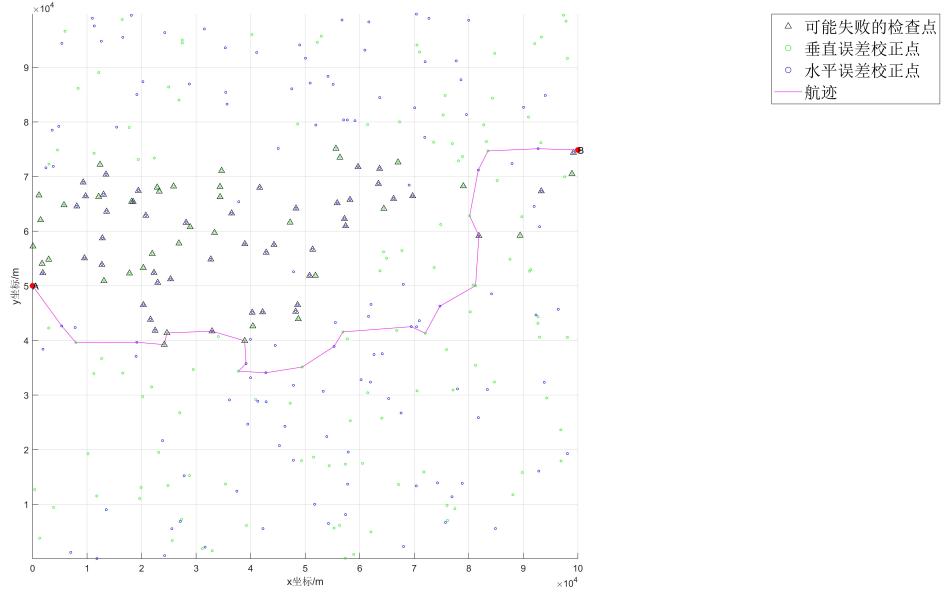


图 24 附件 2 所求出的第三问的航迹俯视图

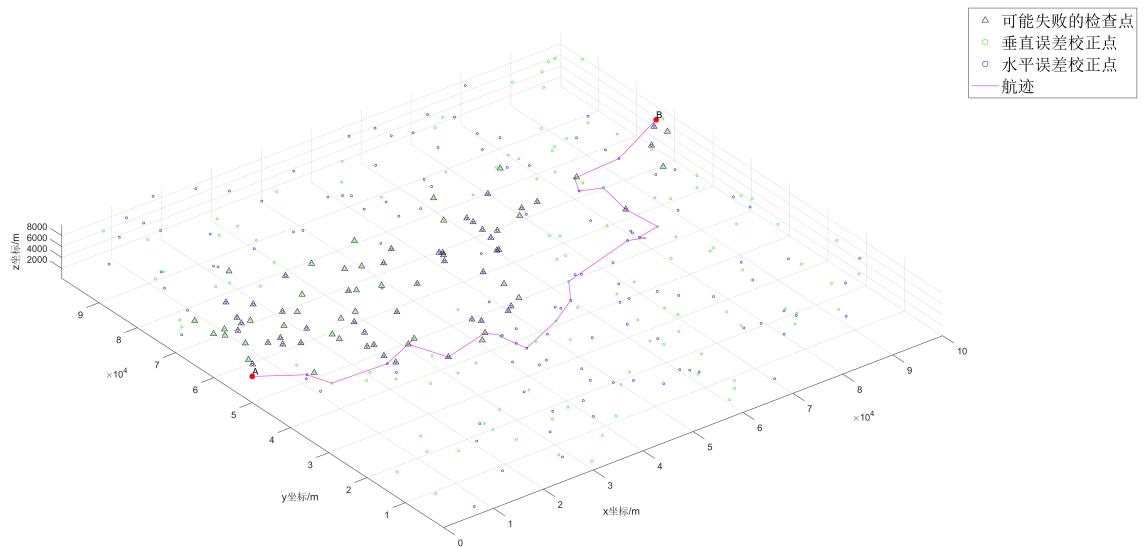


图 25 附件 2 所求出的第三问的航迹侧视图

经计算可得其航迹规划结果如表11所示，所需要的校正点(包含起始点与终止点)个数为 24 个，异常点为 5 个。从表中可以看出所求得的所有水平校正点，即校正点类型为 01 和 02 的点，其均满足校正前的垂直误差不大于 $\beta_1 = 15$ 个单位，水平误差不大于 $\beta_2 = 20$ 个单位。所求得的所有垂直校正点，即校正点类型为 11 和 12 的点，其均满足校正前的垂直误差不大于 $\alpha_1 = 20$ 个单位，水平误差

不大于 $\alpha_2 = 10$ 个单位。抵达终点 B 时的水平误差和垂直误差均小于 $\theta = 20$ 个单位，满足所设定的终点约束条件。

表 11 附件 2 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
169	9.27	9.27	01
322	13.42	4.15	11
100	11.18	15.33	01
60	17.09	5.9	12
137	8.11	9.02	11
194	9.88	18.9	01
190	16.86	6.98	12
151	9.4	11.37	01
36	11.58	2.18	11
296	6.29	8.47	01
250	13.38	7.09	11
243	6.96	14.05	01
73	10.5	3.54	11
249	12.85	16.39	01
274	15.69	2.84	11
12	6.44	9.28	01
216	14.24	7.8	11
279	9.22	17.02	01
301	14.17	4.95	11
38	9.87	14.83	01
110	13.8	3.93	11
99	9.2	13.13	01
326	17.85	8.65	终点 B

4. 算法复杂度与有效性分析

4.1 算法复杂度分析

问题 1：该算法的时间复杂度主要来源于两个方面，第一方面每个校正点可以到达的下一个点的搜索时长，该步骤的时间复杂度为 $O(n)$ ；第二方面是判断每个校正点下一步可以到达的点的有效性所需时长，这一步的时间复杂度对于每一个候选点来说为 $O(n)$ 。所以理论上时间复杂度为 $O(n^2)$ ，但由于点分布的密度是固定的，随着 n 的增加，每一步的候选点的数目可以近似认为是常数。所以该算法的时间复杂度为 $O(n)$ 。又因为算法仅需存储每一个步骤的相关信息，所以其空间复杂度为 $O(n)$ 。

问题 2：问题 2 仅将直线距离替换为转弯距离和直线距离之和，其复杂度相比于问题 1 无变化，时间和空间复杂度均为 $O(n)$ 。

4.2 算法有效性分析

本算法利用贪心算法计算每一步的最优决策，可以有效的找到问题的满意解，虽然该解可能不是全局最优，但该算法具有优越的时间与空间复杂度。在 i7-8750H, 2.2GHz 主频的笔记本电脑上，使用 MATLAB R2019a 求解第一问附件 1 数据仅需 0.103120s，附件 2 数据仅需 0.109964s。求解第二问附件 1 数据需 0.174160s，附件 2 数据 0.177354s，可见算法的高效性。

5. 对第一问的改进

5.1 对附件一数据的优化

由于第一问采用了贪心算法，其以局部最优来代替全局最优，也许会存在所求解并非最优解的情况。而在第三文中所采用的禁忌搜索法可在局部最优的基础上寻求更加优越的点，从而可能得到最佳解。因此，我们在第一问的贪心算法基础上结合了禁忌搜索，并得到了一个航迹长度更小的路径，但校正次数比之前多一次。

所求得的优化的航迹校正点参数如表 13 所示，其航迹总长度为 104527.36m，所经过的校正点（包含起始点与终止点）个数为 11 个。由于本题为双目标优化问题，优化目标的优先级无法确定。本次以校正区域进行的校正次数尽可能少为最优目标，采用经过校正点次数较少的航迹为最优路径。

表 13 附件 1 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
200	13.95	13.95	0
354	16.18	2.23	1
80	14.84	17.07	0
237	19.47	4.63	1
170	7.69	12.32	1
278	10.46	22.77	0
369	21.89	11.44	1
214	13.31	24.75	0
397	22.33	9.02	1
612	16.97	25.99	终点 B

5.2 对附件二数据的优化

采用贪心算法结合禁忌搜索对附件二中的数据进行的优化求解，得到的校正点参数如表 15 所示。其航迹总长度为 $109342.52m$ ，所经过的校正点(包含起始点与终止点)个数为 14 个。所得结果比之前仅采用贪心算法求得的航迹总长度 $111585.52m$ 要短 $2243m$ ，经过的校正点个数两者都为 14 个。故而可将优化的结果作为最终优化的航迹。

表 15 附件 2 航迹规划结果

校正点编号	校正前垂直误差	校正前水平误差	校正点类型
0	0	0	出发点 A
163	13.29	13.29	0
114	18.62	5.33	1
8	13.92	19.26	0
309	19.44	5.52	1
305	5.97	11.49	0
123	15.17	9.2	1
45	10.01	19.21	0
160	17.49	7.49	1
92	5.78	13.26	0
93	15.26	9.48	1
93	9.83	19.32	0
292	16.39	6.55	1
326	6.69	13.51	终点 B

6. 总结

结合了禁忌搜索的贪心算法的时间复杂度比传统贪心算法大，其求解问题所花费的时间比传统贪心算法多。两种方法均有各自的优势，若在不要求高精度解的情况下，贪心算法可以快速求解一个近似全局最优的解。若对解的精度要求较高而无对求解速度的要求，便可采用禁忌搜索对贪心算法所求结果进行优化求解，得到更为精确的最优解。我们建立了多目标 0 – 1 整数规划模型。针对多目标、决策变量与约束条件规模很大的复杂规划问题，我们采用贪心算法求解，实现了题干中快速规划的要求。对于校正可能失败的情况，我们添加航迹的成功概率为优化目标，使用禁忌搜索的方法，从初始解出发搜索尽可能同时满足多优化目标的解。

对于问题 1，优化目标为航迹的总长度与使用的校正点的个数。我们使用贪心算法求解，把全局优化目标转化为每一步决策的优化目标，将优化航迹的总长度转化为优化每一步行进的方向矢量与当前位置与终点的连线矢量的夹角，并且把优化使用的校正点的个数转化为最大化每一步行进的长度，创新性地提出了适应度的概念。通过选取每一步决策中适应度最大的点作为下一步的目标点，

利用贪心算法在短时间内求得了满意解。算法在 $0.1s$ 左右即可求得满意解。对于附件 1，使用 10 个校正点（包括起点与终点），路径长度为 $106350.06m$ ；对于附件 2，使用 14 个校正点，路径长度为 $111585.52m$ 。

对于问题 2，我们在问题 1 的方法基础上通过几何分析与计算机模拟，分析出了飞行器每一步的最佳转弯策略，并给出详细的两点间最短距离计算方法。在距离计算方法的基础上，继续使用贪心算法求解，在短时间内求得了满意解。算法在 $0.17s$ 左右求得满意解，对于附件 1，使用 10 个校正点，路径长度为 $106440.33m$ ；对于附件 2，使用 14 个校正点，路径长度为 $111780.13m$ 。

对于问题 3，由于校正点以一定概率校正失败所带来的不确定性，无法继续沿用问题 1、2 的思路使用贪心算法进行求解，而校正点概率失败也增加了约束条件与问题的复杂度。于是我们考虑采用半启发式方法——禁忌搜索来求解。由于问题 1 中的解即为可行解，并且其具有较短的总路径长度与较少的航迹点。因此我们使用问题 1 中的解作为初始解，通过穷举法计算每个解的成功率，综合路径长度、航迹点个数与成功率作为优化目标，使用禁忌搜索的方法优化初始解。我们同时优化了禁忌搜索算法，使用向航迹中插入新点的方法与模拟退火的思想加大搜索空间，增加了搜索的成功率。在计算附件 2 时，由于初始解穿过异常点密集的区域导致禁忌搜索无法跳出局部最优，于是我们通过改进问题 1 中得到初始解的方法，适当调低异常点的适应度，最终得到成功率较高的满意解。对于附件 1，使用 11 个校正点，路径长度为 $110051.66m$ ，成功率为 100%；对于附件 2，使用 24 个校正点，路径长度为 $165026.79m$ ；成功率为 76.8%。

最后，我们使用问题 3 中禁忌搜索的思想，对问题 1 中的解进行了优化，使问题 1 的答案部分得到优化。对于附件 1，得到路径更短但使用校正点更多的解，该解使用 11 个校正点，路径长度为 $104526.36m$ ；对于附件 2，得到路径更短且校正点个数相同的解，该解使用 14 个校正点，路径长度为 $109342.28m$ 。

本模型也存在一些缺点：一是无法保证求得的解是全局最优解。二是禁忌搜索的参数难以调整，并且当问题规模加大时，搜索时间难以保证。在要求更精确，规模更大，随机性更强的问题的求解算法上，该模型还有待优化与改进。

参考文献

- [1] 郭国林, 基于 Dijkstra 算法的无人机航迹规划应用研究[J], 中国高新技术企业, 2011 年 (2) : 24-26, 2011 年。
- [2] 尹高扬, 周绍磊, 吴青坡, 无人机快速三维航迹规划算法[J], 西北工业大学学报, 2016 年第 34 卷第 4 期: 564-570, 2016 年。
- [3] 卓金武, MATLAB 在数学建模中的应用[M], 北京: 北京航空航天大学出版社, 17-28, 2011 年。
- [4] 陆国峰, 基于多约束多目标的旅游路线推荐及关键算法研究[D], 湖南: 国防科学技术大学, 16-18, 2013 年
- [5] 姜启源 谢金星 叶俊, 数学模型 (第五版) [M], 北京: 高等教育出版社, 96-130, 2019 年。
- [6] 夏 洁, 高金源, 余舟毅, 基于禁忌搜索的启发式任务路径规划算法[D], 控制与决策, 2002 年 17(z1): 773-776, 2002 年。
- [7] Stepanenko S Engels B, New Tabu Search based global optimization methods outline of algorithms and study of efficiency [J], Journal of computational chemistry, 2008, 29(5): 768-780, 2008.
- [8] 潘郁, 达庆利, $1|fuzzy|\min \sum_{i=1}^n C_i$ 模型的禁忌搜索算法[J], 东南大学学报 (自然科学版), 2006 年第 36 卷第 5 期: 852-856, 2006 年。

附录 A 问题一的求解代码

```
% 问题1的求解程序

clc;
clear all;
data1 = xlsread('append1.xlsx');
a1 = 25;
a2 = 15; % 垂直误差校正需要垂直误差<a1, 水平误差<a2
b1 = 20;
b2 = 25; % 水平误差校正需要垂直误差<b1, 水平误差<b2
theta = 30;
delta = 0.001;
p = 0.9869; % 收益偏好系数

% data1 = xlsread('append2.xlsx');
% % data1 = data1(~(data1(:, 6) == 1), :);
% a1 = 20;
% a2 = 10; % 垂直误差校正需要垂直误差<a1, 水平误差<a2
% b1 = 15;
% b2 = 20; % 水平误差校正需要垂直误差<b1, 水平误差<b2
% theta = 20;
% delta = 0.001;
% p = 0.5707; % 收益偏好系数

max_error = [a1 a2];
pointCounts = size(data1, 1);
pointA = data1(1, 2:4);
pointB = data1(pointCounts, 2:4);
% 所有校正点与校正点类型
error_points = data1(2:pointCounts-1, 2:4);
error_types = data1(2:pointCounts-1, 5);
pointCounts = pointCounts - 2;

now_error = zeros(1, 2);
now_point = pointA;
step = 0;
road_point = [];
road_point = [road_point; pointA];
road_index = [];
road_index = [road_index; 0];
```

```

history_forbidden = [];
while(norm(now_point - pointB) * delta + now_error(1) > theta ||
      norm(now_point - pointB) * delta + now_error(2) > theta)
%    寻找可用点, 计算参数
available_points = [];
available_types = [];
available_lengths = [];
available_anglescos = [];
for i = 1:pointCounts
flag = true;
for m = 1:size(history_forbidden, 1)
if (all(error_points(i, :) == history_forbidden(m, :)))
flag = false;
end
end
if (~all(now_point == error_points(i, :)) && flag)
if (error_types(i) == 1)
if (norm(now_point - error_points(i, :)) * delta + now_error(1)
< a1 && norm(now_point - error_points(i, :)) * delta +
now_error(2) < a2)
available_points = [available_points; error_points(i, :)];
available_types = [available_types 1];
available_lengths = [available_lengths norm(now_point -
error_points(i, :))];
available_anglescos = [available_anglescos dot(pointB -
now_point, error_points(i, :) - now_point) / norm(pointB -
now_point) / norm(error_points(i, :) - now_point)];
end
else
if (norm(now_point - error_points(i, :)) * delta + now_error(1)
< b1 && norm(now_point - error_points(i, :)) * delta +
now_error(2) < b2)
available_points = [available_points; error_points(i, :)];
available_types = [available_types 0];
available_lengths = [available_lengths norm(now_point -
error_points(i, :))];
available_anglescos = [available_anglescos dot(pointB -
now_point, error_points(i, :) - now_point) / norm(pointB -
now_point) / norm(error_points(i, :) - now_point)];
end

```

```

end
end
end

%    计算点的可行性
a_cnt = size(available_points, 1);
available = zeros(1, a_cnt);
for i = 1:a_cnt
for j = 1:pointCounts
if (~all(available_points(i, :) == error_points(j, :)))
if (available_types(i) == 0)
if (error_types(j) == 1 && (norm(available_points(i, :) -
error_points(j, :)) + available_lengths(i)) * delta +
now_error(1) < a1)
available(i) = 1;
break;
end
else
if (error_types(j) == 0 && (norm(available_points(i, :) -
error_points(j, :)) + available_lengths(i)) * delta +
now_error(2) < b2)
available(i) = 1;
break;
end
end
end
end
end

available(available_anglescos < 0) = 0;
available_points(available == 0, :) = [];
available_types(available == 0) = [];
available_lengths(available == 0) = [];
available_anglescos(available == 0) = [];

if(size(available_points, 1) == 0)
disp('回溯');
history_forbidden = [history_forbidden; now_point];
now_error = zeros(1, 2);
now_point = pointA;
step = 0;
road_point = [];

```

```

road_point = [road_point; pointA];
continue;
end

a_cnt = size(available_points, 1);
suits = zeros(1, a_cnt);
for i = 1:a_cnt
%     计算适应度
%     计算紧张度
if (available_types(i) == 0)
tense = b2 - now_error(2);
else
tense = a1 - now_error(1);
end
suits(i) = (p * available_anglescos(i) * 10000 + (1 - p) *
available_lengths(i)) + 1e5 / tense;
%     suits(i) = available_anglescos(i) * available_lengths(i)
% / tense;
end
[M, I] = max(suits);

now_point = available_points(I, :);
before = [0 0];
if (available_types(I) == 0)
before(1) = now_error(1) + available_lengths(I) * delta;
before(2) = now_error(2) + available_lengths(I) * delta;
now_error(2) = 0;
now_error(1) = before(1);
else
before(1) = now_error(1) + available_lengths(I) * delta;
before(2) = now_error(2) + available_lengths(I) * delta;
now_error(1) = 0;
now_error(2) = before(2);
end
step = step + 1;
road_point = [road_point; now_point];
fprintf('第%d步, 当前点坐标: %f, %f, %f
, 当前垂直误差: %f, 当前水平误差: %f\n', step, now_point(1),
now_point(2), now_point(3), before(1), before(2));
if step > 100

```

```

break;
end
end

now_error(1) = now_error(1) + norm(pointB - now_point) * delta;
now_error(2) = now_error(2) + norm(pointB - now_point) * delta;
step = step + 1;
now_point = pointB;
road_point = [road_point; now_point];
fprintf('第%d步, 当前点坐标: %f, %f, %f
        ,当前垂直误差: %f, 当前水平误差: %f\n', step, now_point(1),
        now_point(2), now_point(3), now_error(1), now_error(2));

% 计算长度
length = 0;
for i = 1:size(road_point, 1) - 1
length = length + norm(road_point(i, :) - road_point(i + 1, :));
end
fprintf('总长度: %f\n', length);

% 保存数据
% save('solv1_1.mat', 'best_road');
% save('solv1_2.mat', 'best_road');

```

附录 B 问题二中计算两点间长度的代码

```

% 用于第二问, 根据入射角、起点、终点和转弯半径计算两点间的最短距离、转弯圆心O
% 与直线与圆的切点C

function [length, O, C] = calculateLength2(point1, point2,
    inVector, r)
% 起点为A的情况
if (all(inVector == [0 0 0]))
length = norm(point2 - point1);
O = (point2 - point1) / length * r;
C = point1;
else
% 根据几何公式计算相应的数据
lineVector = point2 - point1;
l = norm(lineVector);

```

```

varphi = acos(dot(inVector, lineVector) / norm(inVector) / 1);
l1 = r * sin(varphi);
l2 = l - l1;
l3 = sqrt(l2 ^ 2 + r ^ 2 - l1 ^ 2);
l4 = sqrt(l3 ^ 2 - r ^ 2);
a3 = acos(l4 / l3);
a4 = acos(l2 / l3);
a2 = a3 - a4;
a1 = varphi + a2;
length = r * a1 + l4;
l5 = l1 + r * cos(varphi) * tan(a2);
AD = lineVector / l * l5;
n = cross(inVector, lineVector);
AO = cross(n, inVector);
AO = AO / norm(AO) * r;
O = point1 + AO;
OD = -AO + AD;
OC = OD / norm(OD) * r;
C = O + OC;
end
end

```

附录 C 问题三的求解代码

```

% 问题3的求解程序

clc;
clear all;
data3 = xlsread('append1.xlsx');
a1 = 25;
a2 = 15; % 垂直误差校正需要垂直误差<a1, 水平误差<a2
b1 = 20;
b2 = 25; % 水平误差校正需要垂直误差<b1, 水平误差<b2
theta = 30;
delta = 0.001;

% data3 = xlsread('append2.xlsx');
% a1 = 20;
% a2 = 10; % 垂直误差校正需要垂直误差<a1, 水平误差<a2

```

```

% b1 = 15;
% b2 = 20; % 水平误差校正需要垂直误差<b1, 水平误差<b2
% theta = 20;
% delta = 0.001;

original_solution = [0 521 64 80 170 278 369 214 397 612];
% original_solution = [0 252 266 100 137 194 126 47 205 258
250 86 73 39 274 12 216 279 301 61 292 326];

% 禁忌搜索
solution = original_solution;
point_counts = size(solution, 2);
max_iter = 1000;
jiejin = 100; % 禁忌表长度
insert_number = 0; % 松弛点个数
scores = [];
forbidden = zeros(size(data3, 1) - 2); % 禁忌表
solution_detail = calculate_solution_detail(data3, solution,
a1, a2, b1, b2, theta, delta);
[success_rate, false_index_sort] =
calculate_success_rate(data3, solution, a1, a2, b1, b2,
theta, delta);
score = solution_detail.length + 1e5 / success_rate;
scores = [scores score];
for it = 1:max_iter
cant_find_better_solution = true;

to_change_order = false_index_sort;
for i = 2:size(solution, 2) - 1
if (~ismember(solution(i), false_index_sort))
to_change_order = [to_change_order; solution(i)];
end
end

% 进行点的插入
if (insert_number > 0)
insert_number = insert_number - 1;
tc_index = ceil((size(solution, 2) - 1) * rand());
after_index = tc_index + 1;

```

```

min_sum_length = inf;
for i = 2:size(data3, 1) - 1
n_point = data3(i, 2:4);
sum_length = norm(solution_detail.points(tc_index, 1:3) -
n_point) + norm(solution_detail.points(after_index, 1:3) -
n_point);
if (sum_length < min_sum_length && ~ismember(i - 1,
solution))
min_sum_length = sum_length;
best_insert = i - 1;
end
end

solution = [solution(1:tc_index) best_insert
solution(tc_index + 1:size(solution, 2))];
solution_detail = calculate_solution_detail(data3, solution,
a1, a2, b1, b2, theta, delta);
[success_rate, false_index_sort] =
calculate_success_rate(data3, solution, a1, a2, b1, b2,
theta, delta);
score = solution_detail.length + 1 / success_rate * 1e5;
scores = [scores score];
continue;
end

for idx = 1:size(to_change_order, 1)
to_be_changed = to_change_order(idx);
tc_index = find(solution == to_be_changed);
before_index = tc_index-1;
before_error = solution_detail.errors(before_index, :);
after_index = tc_index+1;
% 寻找邻域
changeable = [];
for i = 2:size(data3, 1) - 1
index = i - 1;
n_point = data3(i, 2:4);
if (data3(i, 5) == 1)
flag1 = before_error(1) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta < a1;
flag2 = before_error(2) + norm(n_point -

```

```

    solution_detail.points(before_index, 1:3)) * delta < a2;
if (solution_detail.points(after_index, 4) == 1)
flag3 = norm(n_point - solution_detail.points(after_index,
1:3)) * delta < a1;
flag4 = before_error(2) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta +
norm(n_point - solution_detail.points(after_index, 1:3))
* delta < a2;
else
flag3 = norm(n_point - solution_detail.points(after_index,
1:3)) * delta < b1;
flag4 = before_error(2) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta +
norm(n_point - solution_detail.points(after_index, 1:3))
* delta < b2;
end
else
flag1 = before_error(1) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta < b1;
flag2 = before_error(2) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta < b2;
if (solution_detail.points(after_index, 4) == 1)
flag3 = before_error(1) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta +
norm(n_point - solution_detail.points(after_index, 1:3))
* delta < a1;
flag4 = norm(n_point - solution_detail.points(after_index,
1:3)) * delta < a2;
else
flag3 = before_error(1) + norm(n_point -
solution_detail.points(before_index, 1:3)) * delta +
norm(n_point - solution_detail.points(after_index, 1:3))
* delta < b1;
flag4 = norm(n_point - solution_detail.points(after_index,
1:3)) * delta < b2;
end
end
flag6 = ~ismember(index, solution));
if (flag1 && flag2 && flag3 && flag4 && flag6)
changeable = [changeable index];

```

```

end
end

% 替换并评价解的好坏

best_score = score;
best_solution = solution;
best_index = 0;
for i = 1:size(changeable, 2)
temp_solution = solution;
temp_solution(tc_index) = changeable(i);
[s_rate, temp_false_index_sort] =
    calculate_success_rate(data3, temp_solution, a1, a2, b1,
    b2, theta, delta);
temp_detail = calculate_solution_detail(data3,
    temp_solution, a1, a2, b1, b2, theta, delta);
temp_score = temp_detail.length + 1e5 / s_rate;
p = rand();
%
% 如果解更好，则接受
if (temp_score < best_score || (p < (max_iter - it) * 2e-4))
%
% 如果在禁忌表中，但比历史最优解好，则破禁
if (forbidden(to_be_changed, changeable(i)) == 0 ||
    temp_score < all_best_score)
best_score = temp_score;
best_solution = temp_solution;
best_index = changeable(i);
best_detail = temp_detail;
best_false_index_sort = temp_false_index_sort;
end
if (temp_score < all_best_score)
all_best_score = temp_score;
all_best_solution = temp_solution;
end
end
end
if (best_index > 0)
forbidden(forbidden > 0) = forbidden(forbidden > 0) - 1;
forbidden(to_be_changed, best_index) = jiejin;
forbidden(best_index, to_be_changed) = jiejin;
solution = best_solution;
solution_detail = best_detail;

```

```
false_index_sort = best_false_index_sort;
score = best_score;
scores = [scores best_score];
cant_find_better_solution = false;
break;
else
continue;
end
end
if (cant_find_better_solution)
break;
end
end
```