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		杭州电子科技大学
参赛队号		**19103360033**
队员姓名	**1.**	王砚威
	2.	葛照楠
	3.	程科远

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 多约束条件下智能飞行器航迹快速规划

摘 要：

飞行器航迹规划是保证飞行任务圆满完成的重要技术支撑，在民用和军用领域都得到了广泛的应用。由于执行飞行任务时环境的复杂化，飞行器对自身定位误差进行校正显得格外重要，本文主要研究智能飞行器在系统定位精度限制下的航迹快速规划问题。

针对问题一，**首先论证了要达到航迹最短和校正点最少时，飞行器应交替经过水平垂直校正点是最优的飞行策略。**在这个基础上，建立了飞行航迹长度尽可能小和经过校正区域进行校正的次数尽可能少的双目标规划模型。模型求解时，首先，通过各点已知的坐标生成距离矩阵，其次，对于不符合约束的以及同类校正点的距离值将其设为无穷大。将原先的双目标约束求解转换成图结构中从一点出发到达另外一点的最少校正点数与最短路径问题，结合 Dijkstra 算法和广度优先算法进行求解，求得任意校正点个数所对应的最短路径。解得数据集一的最短路径是 **A-->503-->294-->91-->607-->170-->278-->369-->214-->397-->B**，共经过 **9** 个校正点，航迹长度为 **104065.88m**；而最少校正点下的最短路径则是 **A-->503-->69-->237-->155-->338-->457-->555-->436-->B**，经过的校正点个数为 **8**，飞行航迹长度为 **104898.37m**。数据集二的路径为 **A-->163-->114-->8-->309-->121-->123-->45-->160-->92-->93-->61-->292-->B**，经过校正点共 **12** 个，飞行航迹长度为 **110772.81m**，证明了算法得到的解就是全局最优解，同时具有较好的时间复杂度。

针对问题二，添加曲率约束条件，构建新的双目标规划模型。已知平面曲线最优是 Dubins 曲线，对于特定输入航线，计算 Dubins 距离代替欧式距离，进行约束判断以及较优解筛选。通过三次样条插值曲线，进一步进行距离差值判断，挑选最优解。数据集一的最优航迹是 **A-->503-->294-->91-->607-->170-->278-->369-->214-->397-->B**，其校正点数为 **9**，航迹长度为 **104931.98m**；数据集二的最优轨迹是 **A-->163-->114-->234-->222-->227-->309-->121-->123-->45-->160-->92-->93-->61-->292-->B**，其校正点数为 **14**，航迹长度为 **119563.23m**，得到了比较不错的解。

针对问题三，建立成功概率最高、航迹长度最短和经过的校正点个数最少的多目标随机规划模型。论证了对于特定的航迹矩阵，随机的通过期望是离散分布的，其期望为0.8^{a}，a为这个特定的航迹中所有问题校正点都失败时无法通过的环节个数。因此对于a取 0，1，2 是分别得到了通过约束概率不小于 100%，80%和 64%的最优解。其中对于第一组数据集，在 **100%**的通过下限时最优路径的校正点数为 **10**，航迹长度为 **105189.50m**，对于 **80%**的通过概率下限时最优路径的校正点数为**9**，航迹长度为**104239.07m**。对于数据集二，在 **100%**的通过下限时最优路径的校正点数是 **21**，航迹长度为 **165615.21m**；在 **80%**的通过下限时

最优路径的校正点数为 **19**，航迹长度为 **145510.59m**；在 **64%**的通过下限时，最优路径的校正点数为 **16**，航迹长度为 **128739.94m**。

关键词：智能飞行器，航迹规划，多目标优化，距离矩阵，Dijkstra

目录

1. 问题重述

1.1 问题背景

航迹规划是指在一定的约束条件下，寻找从起始点到目标点满足某种性能指标最优，或者是满意的运动轨迹[3-5]。由于飞行器在执行任务时的环境的复杂化以及不可控因素的干扰，复杂环境下的航迹快速规划成为智能飞行器的一个重要课题[6-7]。飞行器自身系统结构的限制给精准定位带来困难，定位误差积累到一定程度可能会导致本次飞行任务的失败。因此，在飞行过程中对智能飞行器的定位误差进行校正是航迹规划中的一项重要任务。

飞行器在执行飞行任务时，其航迹有如下约束规则：

a)飞行过程中的定位误差包含垂直误差和水平误差，飞行轨迹每增加 1m，垂直误差和水平误差各相应增加δ个单位，且到达终点时垂直误差和水平误差都要小于θ个单位。

b)飞行区域中存在校正点用于飞行器定位误差的校正，到达校正点时，飞行器能够对该位置的误差校正类型进行校正，垂直和水平误差都及时得到校正则飞行器可按照预定的航迹顺利到达目的地。

c)在起始点处，飞行器的垂直和水平误差均为 0。

d)飞行器在垂直误差校正点进行垂直误差校正后，其垂直误差将变为 0，水平误差保持不变。

e)飞行器在水平误差校正点进行水平误差校正后，其水平误差将变为 0，垂直误差保持不变。

f)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。

g)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

h)飞行器转弯时前进方向不能突然改变，假设最小转弯半径是 200m。

1.2 待解决的问题

针对智能飞行器在系统定位精度限制下的航迹快速规划问题问题，本文需要解决的问题如下：

问题一：使用附件 1 和附件 2 中给出的飞行区域坐标和对应点的校正类型，分别绘制满足航迹约束规则 a)~g)时的规划路径(给出飞行器从起点出发经过误差校正点编号和校正前误差结果表)，以保证航迹的长度尽可能小以及经过校正区域进行校正的次数尽可能的少，并完成所用算法的有效性和复杂度分析。

问题二：在满足约束规则 a)~g)前提下，考虑飞行器的最小转弯半径为 200m，针对附件 1 和附件 2 的数据集，分别绘制满足条件 a)~h)时的飞行器航迹(给出飞行器从起点出发经过误差校正点编号和校正前误差结果表)，保证航迹的长度尽可能小以及经过校正区域进行校正的次数尽可能的少，并讨论算法的有效性和复杂度。

问题三：由于天气等不可控因素的影响，飞行器在部分校正点进行误差校正时，可能存在无法某个误差精确校正为 0 的情况。假设飞行器在部分校正点能成功将某个误差校正为 0 的概率为 80%，且不论校正是否成功均不改变规划路径。要求在此情况下对问题一所要求的航迹进行重新规划，使得成功到达终点的概率尽可能大，绘制在两个数据集下的航迹规划路径(给出飞行器从起点出发经过误差校正点编号和校正前误差结果表)。

2. 问题假设

(1) 假设当垂直误差和水平误差均小于θ个单位时，飞行器能按照规划路径飞行。
(2) 假设飞行器到达某校正点时即可知道在该点处能否校正成功。
(3) 假设飞行器在部分校正点能成功将某个误差校正为 0 的概率为 80%。
(4) 假设每个问题校正点的失败概率互相独立。
(5) 假设在校正点处校正失败后的剩余误差认为是 5 个单位(对于较优解几乎没有影响，因为较优解不会发生连续两段飞行距离之和还小于 5000 的可能)。

3. 符号说明

符号	说明
$d(i,j)$	校正点i到校正点j间的欧式距离
S_f	飞行器在校正点处的校正位移量
L_f	飞行航迹长度
$N(i,j)$	为 1 时表示选取从i点出发到j点的路径
m_f	有概率校正失败的问题点个数
p_s	成功到达终点的概率下限值
$\tilde{d}(i,j)$	满足最小半径约束校正点i到校正点j间的曲线距离
T_R	飞行器的最小转弯半径
R_c	经过的校正点的个数
D_w	可能校正失败的校正点

4. 问题一模型建立与求解

4.1 问题一分析

问题一要求根据附件中的两个数据集规划智能飞行器的航迹路径，以使航迹长度尽可能小和经过校正区域进行校正的次数尽可能的少，显然这是一个约束多目标优化问题。

首先，我们对约束条件进行分析，得到了两个定理：

定理一：当垂直-水平校正交替的情况下能够获得最优航迹路径；

定理二：在校正点处的位移变化不影响最终的航迹长度的排名。

对于定理(1)的证明，如图 4-1 所示，假设垂直-垂直-水平校正(两垂直一水平交替校正)也可以获得最优航迹路径，则在第i个校正点处进行水平校正的水平误差与前两次垂直校正时积累的水平误差和应不大于β_2，即

$$\varepsilon(i-2)+\varepsilon(i-1)+\varepsilon(i)\leqslant\beta_2=\alpha_1 \tag{4-1}$$

由(4-1)得

$$\varepsilon(i-2)+\varepsilon(i-1)<\alpha_1 \tag{4-2}$$

由(4-2)可知前两次垂直误差之和小于α_1，可以合并进行校正

$$\delta \cdot [d(i-1,i-2)+d(i-2,i-3)] < \alpha_1 \tag{4-3}$$

三角形两边之和小于第三边，则也满足

$$\delta \cdot d(i-1,i-3) < \alpha_1 \tag{4-4}$$

同样也有，能够飞入第二个垂直校正点，则必然满足

$$\delta \cdot d(i-1,i-3) < \varepsilon(i-2)+\varepsilon(i-1) < \alpha_2 \tag{4-5}$$

因此可以得到，对于飞行器而言，可以直接飞入第二个垂直校正点而不必经过第一个垂直校正点，使得经过的距离更短同时转折点数量更少。对于两水平一垂直交替校正的情况同理，因此垂直和水平交替进行校正是最合适的序列方案。

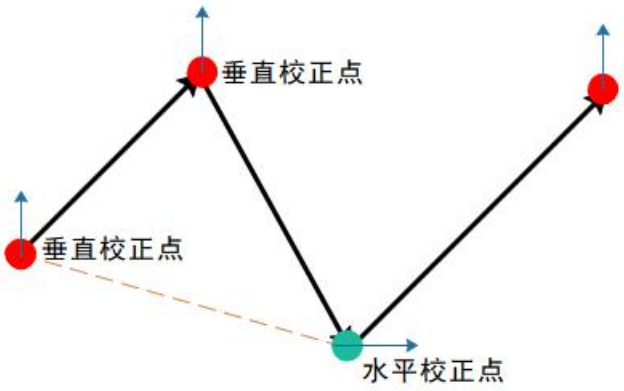

图 4-1 定理一证明示意图

对于定理(2)的证明，飞行器在校正点处的校正位移量S_f(包括垂直位移量和水平位移量，两者大小相同)与航迹长度L_f成正相关，且常系数为λ，如式(4-6)，我们可以得到在校正点处的位移变化不影响最终的航迹长度排名。

$$S_f = \lambda \cdot L_f \tag{4-6}$$

进一步的，根据问题一的约束规则建立多约束的双目标优化模型，采用带约束的最短路径算法并在其中结合定理一的结论大大简化了模型的求解过程，最终得到满足条件的最优解。

4.2 问题一模型的建立

(1) 确定目标函数：智能飞行器从起始点飞往目的地的过程中，以飞行航迹长度尽可能小和经过校正区域进行校正的次数尽可能少为目标进行建模，目标函数如下：

目标函数 1：最小化飞行航迹长度

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij} \cdot d(i,j) \tag{4-7}$$

目标函数 2：最小化经过校正区域进行校正的次数

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij} \tag{4-8}$$

其中，N_{ij}表示是否选取从i点出发到j点的路径，如式(4-9)所示；$d(i,j)$表示从i点出发到j点的欧式距离。

$$N_{ij}=\begin{cases}1, & \text{选取从}i\text{点出发到}j\text{点的路径}\\ 0, & \text{不选取从}i\text{点出发到}j\text{点的路径}\end{cases} \tag{4-9}$$

(2) 确定模型的约束条件，此问题的约束条件如下：

智能飞行器飞入某个校正点和飞出某校正点的次数最多为 1 次，该约束条件为：

$$\begin{cases}\sum_{i=1}^{n} N_{ij}\leqslant 1\\ \sum_{j=1}^{n} N_{ij}\leqslant 1\end{cases} \tag{4-10}$$

除了起始点和终点，飞行器在其他校正点上飞入和飞出的次数相同，有如下约束：

$$\sum_{j=1}^{n}(N_{ji}-N_{ij})=\begin{cases}-1, & i=1\\ 0, & i\in(1,n)\\ 1, & i=n\end{cases} \tag{4-11}$$

到达终点时飞行器的垂直误差和水平误差都要小于θ个单位，则：

$$\delta\cdot[d(k,i)+d(i,n)]<\theta \tag{4-12}$$

式(4-12)中，有$N_{ki}=1,N_{in}=1$。

每次经过垂直校正点，飞行器的垂直误差变为 0，水平误差则为过去一段飞行的误差积累。在飞行器到达下一个水平校正点时，其垂直误差仅仅等于该段飞行距离积累的垂直误差，水平误差则是包括上一段飞行距离在内的两次飞行航迹的误差，而在飞出这个水平校正点后水平误差清零，无人机只存在垂直误差，如此反复。要求无人机能够顺利进入校正点进行，水平和垂直误差需要各自满足对应校正点的进入要求。综上，该约束条件如下：

$$\delta\cdot[d(k,i)+d(i,j)]\leqslant\alpha_1,\ j\text{为垂直校正点} \tag{4-13}$$

$$\delta\cdot d(i,j)\leqslant\alpha_2,\ j\text{为垂直校正点} \tag{4-14}$$

$$\delta\cdot d(j,l)\leqslant\beta_1,\ l\text{为水平校正点} \tag{4-15}$$

$$\delta\cdot[d(i,j)+d(j,l)]\leqslant\beta_2,\ l\text{为水平校正点} \tag{4-16}$$

式(4-13)、(4-14)、(4-15)、(4-16)中，有$N_{ki}=1,N_{ij}=1,N_{jl}=1$。

综上所述，可以建立问题一的数学模型为：

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}N_{ij}\cdot d(i,j)$$

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}N_{ij}$$

$$
s.t.\begin{cases}
\sum_{i=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} (N_{ji} - N_{ij}) = \begin{cases} -1, & i=1 \\ 0, & i \in (1,n) \\ 1, & i=n \end{cases} \\
\delta \cdot [d(k,i) + d(i,n)] < \theta \\
\delta \cdot [d(k,i) + d(i,j)] \leqslant \alpha_1 \\
\delta \cdot d(i,j) \leqslant \alpha_2 \\
\delta \cdot d(j,l) \leqslant \beta_2 \\
\delta \cdot [d(i,j) + d(j,l)] \leqslant \beta_2
\end{cases}
$$

4.3 模型的求解及分析

4.3.1 算法设计

为了求解飞行器在飞行区域中从出发点到终点之间的最优航迹，我们使用优化的最短路径算法，该算法收敛速度快且稳定，算法的具体求解步骤如下：

Step1：读取数据集中的 X、Y、Z 坐标以及校正点类型作为输入；
Step2：生成各校正点间的距离矩阵，将水平与水平(垂直与垂直)校正点之间的距离设为无穷大(INF)，不满足约束规则 a)的设为 INF，两校正点间水平(垂直)误差大于$\alpha_2(\beta_1)$的距离设为 INF；
Step3：使用带约束的广度优先算法获取校正点的最少个数 N；
Step4：使用带约束的 Dijkstra 算法得到最短路径相应的校正点个数 M(M≤N)；
Step5：比较 N 与 M 的大小，若 N=M，转到 Step7;
Step6：取正整数$n \in [M,N]$，将 n 作为约束条件，使用带约束的 Dijkstra 算法得到校正点个数为 n 所对应的最短路径；
Step7：得到校正点的个数和所对应的最优路径。

在 Dijkstra 算法的基础上，加入约束条件，并且融合广度搜索算法里的深度思想，形成带约束优化 Dijkstra 算法，算法流程图如图 4-2 所示。

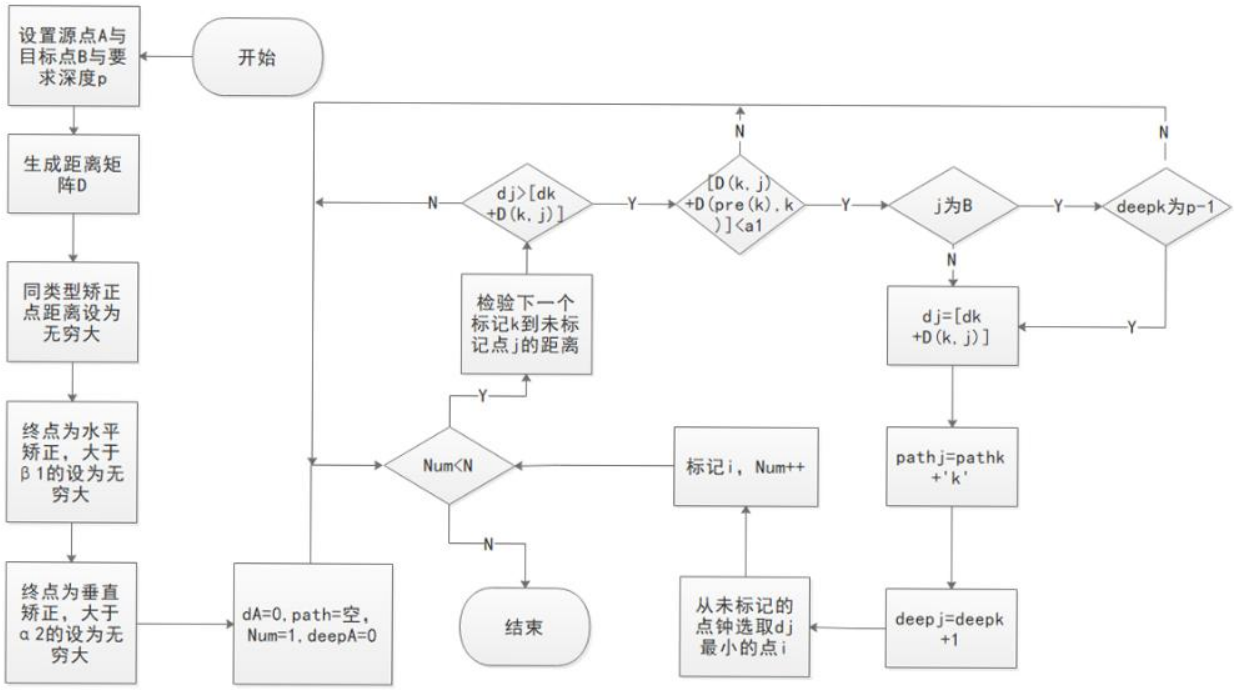

图 4-2 带约束的优化 Dijkstra 算法流程图

4.3.2 问题一求解结果

按照 4.3.1 节所述的算法步骤，我们进行了带约束的 Dijkstra 算法的编程，得到了数据集 1 和数据集 2 的最优飞行航迹。数据集 1 经过**校正点个数为 8** 的飞行路径编号为：**A-->503-->69-->237-->155-->338-->457-->555-->436-->B**，飞行航迹长度为 **104898.37m**；经过**校正点个数为 9** 的飞行路径编号为 **A-->503-->294-->91-->607-->170-->278-->369-->214-->397-->B**，飞行航迹长度为 **104065.88m**。数据集 1 的航迹规划结果分别如表 4-1、表 4-2 所示。

表 4-1 数据集 1 航迹规划结果表(经过校正点 8 个)

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	13.388	13.388	11
69	8.807	22.195	01
237	21.308	12.499	11
155	11.201	23.700	01
338	23.387	12.186	11
457	12.813	24.998	01

555	24.503	11.690	11
436	7.356	19.046	01
612	22.314	14.958	终点 B

表 4-2 数据集 1 航迹规划结果表(经过校正点 9 个)

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	13.388	13.388	11
294	10.181	23.569	01
91	17.536	7.355	11
607	8.353	15708	01
170	11.946	3.593	11
278	10.457	14.050	01
369	21.893	11.436	11
214	13.314	24.750	01
397	22.331	9.017	11
612	16.973	25.990	终点 B

最终得到的数据集 1 的航迹规划路径三维图如图 4-3 所示。

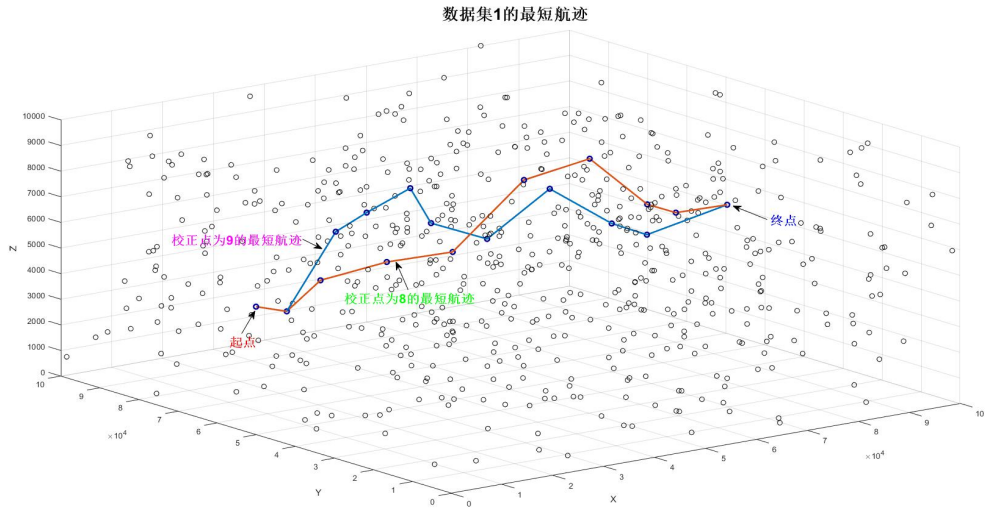

图 4-3 数据集 1 航迹规划路径图

一般地，多目标优化会得到对于每一个指标都有对应的一个最优解，这个最优解就是最优集的一部分，对于这个集合里的最优点而言，任何一个目标上的改进都会带来其他目标的下降，因此整个最优集成为 Pareto 集或者 Pareto 前沿。

对这个问题的两个 Pareto 最优解来说，矫正个数最优为 8，航迹轨迹最优为 104066。通过建立综合评价函数：

$$Score = \frac{R_c}{8} + \frac{L_f}{104066} \tag{4-17}$$

其中 R_c 表示飞行器经过校正点的个数，L_f 为航迹长度。

代入两组参数(8，104898.37)和(9，104065.88)，我们得到经过 8 个校正点的航迹得分为 2.008，经过 9 个校正点的航迹得分为 2.125，得分越小说明越逼近整体最优，因此我们认为经过 8 个校正点且航迹长度为 104065.88m 的解是数据集 1 的最优解。

数据集 2 经过**校正点个数为 12** 的飞行路径编号为：**A-->163-->114-->8-->309-->121-->123-->45-->160-->92-->93-->61-->292-->B**，飞行航迹长度为 **110772.81m**。数据集 2 的航迹规划结果如表 4-3 所示。

表 4-3 数据集 2 航迹规划结果表(经过校正点 12 个)

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	13.288	13.288	01
114	18.622	5.334	11
8	13.922	19.256	01
309	19.446	5.524	11
121	11.252	16.776	01
123	16.603	5.352	11
45	10.006	15.358	01
160	17.491	7.485	11
92	5.776	13.261	01
93	15.601	9.484	11
61	9.834	19.319	01
292	16.388	6.553	11
326	6.961	13.514	终点 B

最终得到的数据集 2 的航迹规划路径三维图如图 4-4 所示。

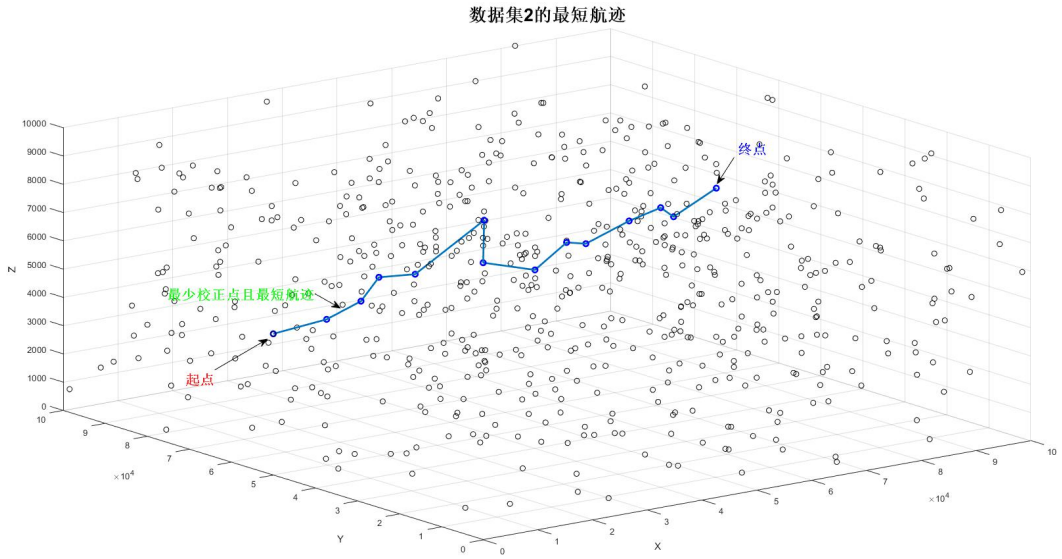

图 4-4 数据集 2 航迹规划路径图

4.3.3 算法有效性和复杂度分析

算法有效性分析：当前算法是可以得到当前约束下的最优解的，证明过程如下

证明：所得路径是在要求约束下的最短路径
假设所得最小值 d[B]不是源点 A 到达 B 的最短路径 S，那么不妨设最短路径是从 A 出发经过 N_到达 B，其中 N_是这条路径最后第二个定点，N_存在三种可能：①属于已经产生的集合 K 但不是最后第二个点，②不属于集合 K 且符合约束，③不属于集合 K 且不符合约束。 1.如果不符合约束，则不可加入路径。 2.如果属于集合 K，由于每一个中间值在求出一个最短距离时都是在符合约束的情况下比较得到，原先得到的最短路径是符合约束里的最小距离，如果 N_到达 B 的更短就应该会保存 N_到达 B 的路径，与现有条件矛盾。 3.如果不属于集合 K 且符合约束，不妨设路径为 A-K1-G-N_-B,其中 K1 是属于集合 A，G 和 N_可以是一个点也可以是不同点，但是都不属于集合 K，则 A-K1-G 的距离明显小于 A-K1-G-N_-B 的距离，G 会比 B 更早进入集合 K，与 G 不在集合 K 矛盾。 所以这样一个 N_点不存在，得证。

算法复杂度分析：因为图比较稠密，因此该算法没有用最小堆改进，而增加的约束判断和深度记录的复杂度较小，时间复杂度为$O(N^2)$，程序运行时间约为 0.6s。

5. 问题二模型建立与求解

5.1 问题二分析

问题二中增加了一个约束，既飞行器的前进方向无法突然改变，即从第一问中折线航迹转换为空间中的圆弧与直线的组合航迹。由于 Dubins 距离最短[1]，我们研究在空间中的三维 Dubins 路径规划。飞行器从一个高度飞到另一个高度的路径是圆形螺旋的一部分，可以想象成圆柱体表面的路径。如图 5-1（飞行器的三维 Dubins 路线）所示，t_s 和 t_f 由 t_0 相连接。首先定义一个初始弧操作，将切向量 t_s 带入 t_0 和 t_f 所在的平面得到 t_{sr}，使用这个旋转向量 t_{sr} 与起始向量计算二维 Dubins 路径。法向量 n_{sr} 为开始向量的法向量，且位于 t_0 和 t_{sr} 所在的平面，与副法向量 b_{sr} 构成右手坐标系。初始操作根据副法向量的曲率来完成二维 Dubins 操作，然后进行滚动操作，这时法向量与副法向量对齐，进行直线操作，然后进行最后的二维 Dubins 操作。这样得到一个综合的 Dubins 策略，即先进行 Dubins 操作到达 t_0 和 t_f 所在的平面,然后在平面上走直线运动，最后在平面上完成 Dubins 操作。

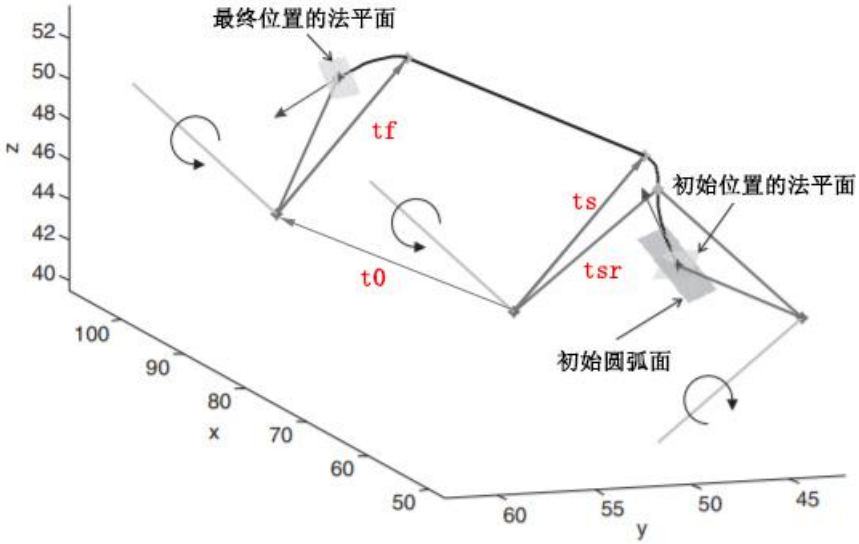

图 5-1 飞行器的三维 Dubins 路线

5.2 问题二模型建立

由于飞行器在转弯时受到结构和控制系统的限制，在某一时刻的飞行方向无法突然发生改变，因此该问题中需要考虑飞行器的最小转弯半径(不小于 200m)，同时修改问题一模型中两校正点之间的欧式距离为两校正点间直线部分和经过校正点附近的圆弧长度之和。建立如下飞行航迹长度尽可能小和经过校正区域进行校正的次数尽可能少的双目标规划模型：

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} N_{ij} \cdot \tilde{d}(i,j)$$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} N_{ij}$$

$$
s.t.\begin{cases}
\sum_{i=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} (N_{ji} - N_{ij}) = \begin{cases} -1, & i = 1 \\ 0, & i \in (1,n) \\ 1, & i = n \end{cases} \\
\delta \cdot \left[\tilde{d}(k,i) + \tilde{d}(i,n) \right] < \theta \\
\delta \cdot \left[\tilde{d}(k,i) + \tilde{d}(i,j) \right] \leqslant \alpha_1 \\
\delta \cdot \tilde{d}(i,j) \leqslant \alpha_2 \\
\delta \cdot \tilde{d}(j,l) \leqslant \beta_2 \\
\delta \cdot \left[\tilde{d}(i,j) + \tilde{d}(j,l) \right] \leqslant \beta_2 \\
T_R \geqslant 200
\end{cases}
$$

其中，$\tilde{d}(i,j)$表示两点间直线和经过校正点附近的圆弧长度之和，T_R为飞行器最小转弯半径。

5.3 模型求解及分析

5.3.1 模型求解方法

三维 Dubins 路径的长度为

$$
h_{Dubins} = h_s + a_t + h_f = \frac{\alpha_s}{k_s} + a + \frac{\alpha_f}{k_f} \tag{5-1}
$$

其中，h代表的是路径的长度；s、t和f分别表示起始弧，直线和结束弧；α和k分别是弧角和曲率。

在本问中，曲率半径不大于 200m，由于要得到最短航迹，曲率半径应越小越好，(可以证明)，所以开始和结束的两个 Dubins 曲线的曲率k_s和k_f都设为 1/200。

采用微分几何方法求解上面三维的 Dubins 路径方程。根据前面的分析中，我们使用 frenet 坐标[2]，定义第一次操作的面为$[t_s \quad n_s \quad b_s]$，第二个操作面为$[t_f \quad n_f \quad b_f]$。两个操作面有一条交线，起始操作面和结束操作面通过t_s向量旋转得到。

因此我们有

$$
\begin{aligned}
[t_{ms} \quad n_{ms} \quad b_{ms}] &= [t_s \quad n_s \quad b_s] R_s \\
[t_{mf} \quad n_{mf} \quad b_{mf}] &= [t_f \quad n_f \quad b_f] R_f
\end{aligned} \tag{5-2}
$$

其中

$$R_s=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\phi_s) & -\sin(\phi_s)\\ 0 & \sin(\phi_s) & \cos(\phi_s)\end{pmatrix}$$
$$R_f=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\phi_f) & -\sin(\phi_s)\\ 0 & \sin(\phi_f) & \cos(\phi_f)\end{pmatrix} \tag{5-3}$$

其中，ϕ_s和ϕ_f分别是起始操作平面和结束操作平面的旋转角。

半径向量r_s是

$$r_s=\begin{bmatrix}t_{ms} & n_{ms} & b_{ms}\end{bmatrix}\begin{pmatrix}0\\ \pm 1/k_s\\ 0\end{pmatrix} \tag{5-4}$$

类似的，半径向量r_f是

$$r_f=\begin{bmatrix}t_{mf} & n_{mf} & b_{mf}\end{bmatrix}\begin{pmatrix}0\\ \pm 1/k_f\\ 0\end{pmatrix} \tag{5-5}$$

两个平面的坐标的关系是

$$\begin{bmatrix}t_f & n_f & b_f\end{bmatrix}=\begin{bmatrix}t_s & n_s & b_s\end{bmatrix}R \tag{5-6}$$

则可以得到更改轴集所需要的旋转矩阵R

$$R=\begin{pmatrix}t_f\cdot t_s & t_f\cdot n_s & t_f\cdot b_s\\ n_f\cdot t_s & n_f\cdot n_s & n_f\cdot b_s\\ b_f\cdot t_s & b_f\cdot n_s & b_f\cdot b_s\end{pmatrix}$$

连接向量a_s，a_f，a_c是一个正交集合，连接向量a_s和a_f与a_c垂直，但不平行。每个向量位于相应的操作平面上，都不重合。两个操纵平面的内部连接向量都是a_c，所以可以写成

$$a_c=a\begin{bmatrix}t_{ms} & n_{ms} & b_{ms}\end{bmatrix}\alpha_s=a\begin{bmatrix}t_{mf} & n_{mf} & b_{mf}\end{bmatrix}\alpha_f$$

$$\alpha_s=\begin{pmatrix}\alpha_{ts}\\ \alpha_{ns}\\ \alpha_{bs}\end{pmatrix},\quad \alpha_f=\begin{pmatrix}\alpha_{tf}\\ \alpha_{nf}\\ \alpha_{bf}\end{pmatrix} \tag{5-7}$$

两种操作平面的 frenet 框架可以由(5-8)联系起来

$$\begin{cases}\begin{bmatrix}t_f & n_f & b_f\end{bmatrix}=\begin{bmatrix}t_s & n_s & b_s\end{bmatrix}R\\ \begin{bmatrix}t_{mf} & n_{mf} & b_{mf}\end{bmatrix}=\begin{bmatrix}t_f & n_f & b_f\end{bmatrix}R_f\\ \begin{bmatrix}t_{ms} & n_{ms} & b_{ms}\end{bmatrix}=\begin{bmatrix}t_s & n_s & b_s\end{bmatrix}R_s\end{cases} \tag{5-8}$$

因此，我们得到

$$\begin{bmatrix}t_{mf} & n_{mf} & b_{mf}\end{bmatrix}=\begin{bmatrix}t_{ms} & n_{ms} & b_{ms}\end{bmatrix}R_s'RR_f \tag{5-9}$$

这意味着

$$\begin{cases}\alpha_s=R_s'RR_f\alpha_f\\ \alpha_f=R_f'RR_s\alpha_s\end{cases} \tag{5-10}$$

而且，半径向量可以用起始操作坐标给出

$$\begin{cases} r_s = \begin{bmatrix} t_{ms} & n_{ms} & b_{ms} \end{bmatrix} \begin{pmatrix} 0 \\ \pm 1/k_s \\ 0 \end{pmatrix} \\ r_f = \begin{bmatrix} t_{ms} & n_{ms} & b_{ms} \end{bmatrix} R_s' R R_f \begin{pmatrix} 0 \\ \pm 1/k_f \\ 0 \end{pmatrix} \end{cases} \tag{5-11}$$

操作平面上的向量a_s和a_f与a_c垂直，他们也可以用起始操作坐标来定义

$$\begin{cases} \alpha_s = \dfrac{\pm 1}{k_s} \begin{bmatrix} t_{ms} & n_{ms} & b_{ms} \end{bmatrix} \beta_s \\ \alpha_f = \dfrac{\pm 1}{k_f} \begin{bmatrix} t_{mf} & n_{mf} & b_{mf} \end{bmatrix} \beta_f = \dfrac{\pm 1}{k_f} \begin{bmatrix} t_{ms} & n_{ms} & b_{ms} \end{bmatrix} R_s' R R_f \beta_s \end{cases} \tag{5-12}$$

这里，为了确保连接向量在操作平面上并且垂直于内部连接向量，则

$$\begin{cases} \beta_s = \dfrac{1}{b_s} \begin{pmatrix} -\alpha_{ns} \\ \alpha_{ts} \\ 0 \end{pmatrix}, \quad \beta_f = \dfrac{1}{b_f} \begin{pmatrix} -\alpha_{nf} \\ \alpha_{tf} \\ 0 \end{pmatrix} \\ b_s = \sqrt{\alpha_{ns}^2 + \alpha_{ts}^2}, \quad b_f = \sqrt{\alpha_{nf}^2 + \alpha_{tf}^2} \\ \beta_s \alpha_s = 0, \qquad \beta_f \alpha_f = 0 \end{cases} \tag{5-13}$$

在开始平面坐标轴$\begin{bmatrix} t_s & n_s & b_s \end{bmatrix}$上，测量终点$p_f$相对于起始点的位置

$$\begin{cases} p_f - p_s = \begin{bmatrix} t_s & n_s & b_s \end{bmatrix} p = \begin{bmatrix} t_{ms} & n_{ms} & b_{ms} \end{bmatrix} R_s' p \\ p_m = R_s' p \\ p = \begin{pmatrix} p_t \\ p_n \\ p_b \end{pmatrix} \end{cases} \tag{5-14}$$

综上所述，即可求出三维 Dubins 模型的路径长度。

程序求解算法流程如图 5-2 所示。

程序的一开始，输入航迹距离矩阵，得到三维 Dubins 距离(S_d)，判断 Dubins 距离是否满足双目标规划约束，如果不满足，则重新输入N_{ij}，反之则判断校正点个数是否小于最少校正点个数与 count 之和，航迹长度是否小于最短长度与 length 之和。count 值是我们要求最大校正点个数与最小校正点个数之差，length 是我们期望的最大的航迹长度与最短长度之差。都满足判断条件，我们对N_{ij}的转角部分进行三次样条插值，得到曲线距离S_c，如果S_c与S_d的差值是否在我们期望的阈值 Y 的范围内，得到较优解，从而可以筛选出全局最优解。

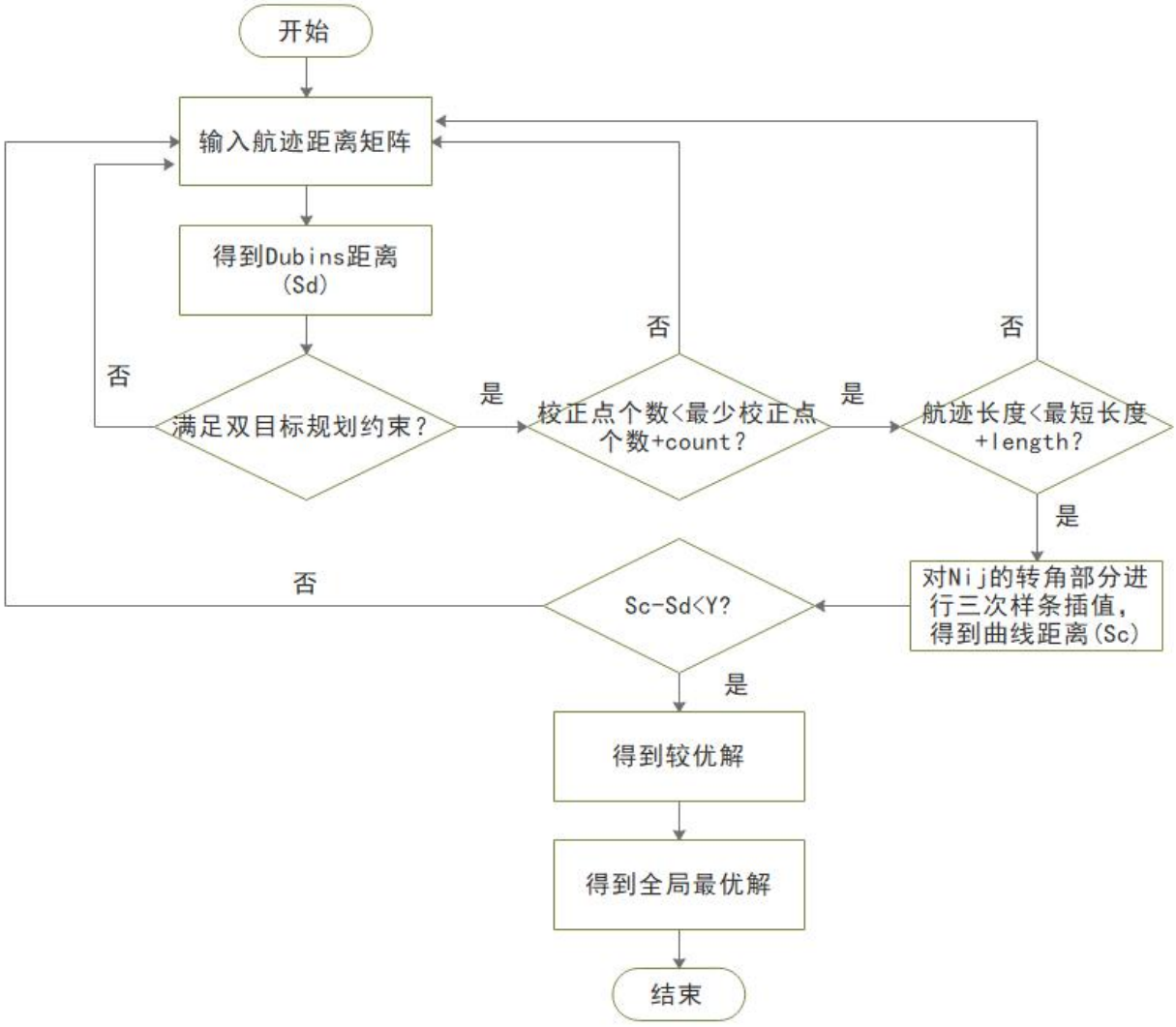

图 5-2 问题二求解算法流程

5.3.2 问题二求解结果

表 5-1 数据集 1 航迹规划结果表(经过校正点 9 个)

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	13.907	13.907	11
294	10.198	24.105	01
91	17.578	7.380	11
607	8.390	15.770	01
170	12.084	3.694	11
278	10.519	14.213	01

369	21.983	11.464	11
214	13.360	24.824	01
397	22.399	9.039	11
612	16.980	26.020	终点 B

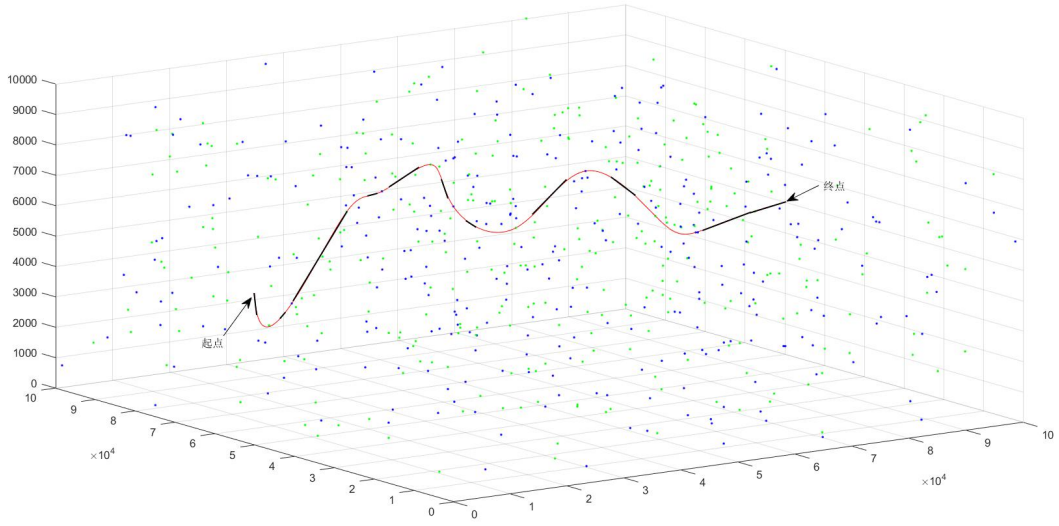

图 5-3 数据集 1 航迹规划路径

表 5-2 数据集 2 航迹规划结果表(经过校正点 14 个)

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	13.384	13.384	01
114	18.793	5.409	11
234	4.992	10.401	01
222	12.639	7.647	11
227	8.189	15.836	01
309	14.296	6.108	11
121	11.402	17.501	01
123	17.188	5.786	11
45	10.123	15.909	01

160	17.666	7.542	11
92	5.796	13.339	01
93	15.311	9.514	11
61	9.867	19.382	01
292	16.428	6.560	11
326	7.242	13.802	终点 B

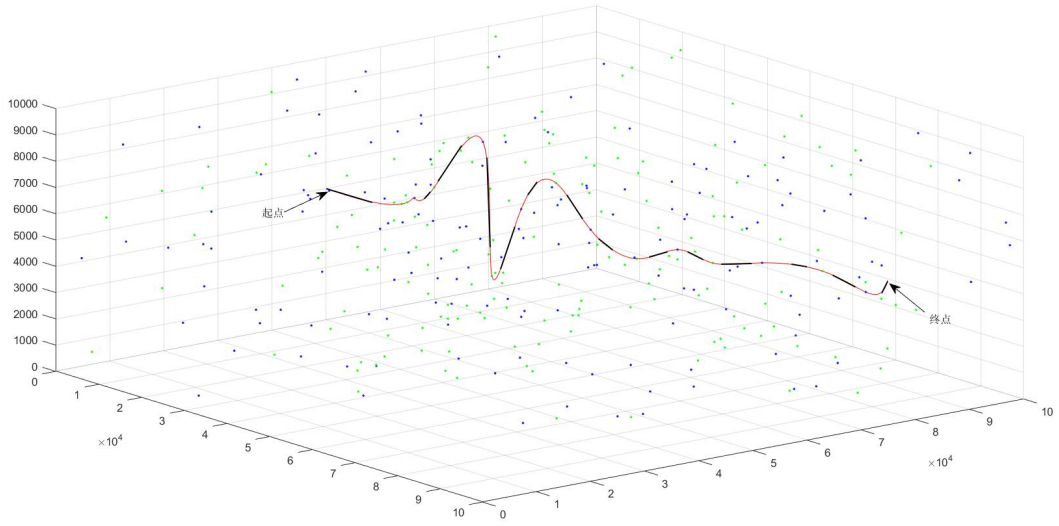

图 5-4 数据集 2 航迹规划路径

5.3.3 算法有效性和复杂性分析

算法有效性分析：不能确保得到最优解，原因如下

1.三维 Dubins 距离未必最短；

2.最优解有概率没有通过较优解筛选，这点可以通过增大 length 和 count 来降低概率，但会牺牲时间；

3.三次样条插值并不是最短曲线，只是曲率最小的曲线，方便计算而挑选，不过可以得到较优解。

算法复杂度分析：

Dubins 的复杂度并不高，对于本算法而言最大的复杂度来源于两个方面：

1.原先的 Dijkstra 的$O(N^2)$复杂度；

2.通过筛选的三次样条插值，每次进行节点数量元的微分方程求解，两者谁为主要是根据筛选条件的差异。

6. 问题三模型建立与求解

6.1 问题三分析

问题三是一个典型的随机约束规划。这类规划问题，因为随机性的存在，比较难以求解。对于特定的问题，可以通过数学分析来找到最优解的结构，但是不适用于大规模数据量的最优规划求解。

对于本题目，我们挑选了机会约束规划来描述，因为必须要在观测到随机变量的实现之前作出决策。因此需要决策在一定程度上不满足特定约束，但该决策使约束成立的概率不小于某一个足够小的置信水平。

机会约束规划的求解一般分为确定性规划和随机模拟的解法，对于可以精确求解的一般适合用前者来达到最优解，对于复杂程度过高无法精确求解的一般适合使用后者来实现较优解。

6.2 问题三模型建立

对于特定的输入N_{ij}，以及特定的问题校正点的失败成功情况（共m_f个问题校正点，其中$h(l)$为 1 代表问题校正点校正成功，为 0 代表校正失败），有一个唯一确定的是否通过结果，用$Z(N_{ij},h)$表示。而特定问题校正点的失败成功情况的各种分布的概率是已知的，对于所有情况通过结果也可以得到，因此也可以得到对于随机校正点失败情况下通过的整体概率$P(N_{ij},h)$，如式(6-1)。

$$P(N_{ij},h)=\sum_{k=0}^{2^{m_f}-1}0.8^{\sum_{l=1}^{m_f}h(l)}\cdot 0.2^{m_f-\sum_{l=1}^{m_f}h(l)}\cdot Z(N_{ij},h) \tag{6-1}$$

其中$h(l)$表示第l个问题校正点是否会发生故障，$Z(N_{ij},h)$表示路径与故障确定时飞行器能否顺利通过。

$$h(l)=\begin{cases}0, \text{第}l\text{个问题校正点发生故障}\\ 1, \text{第}l\text{个问题校正点未发生故障}\end{cases} \tag{6-2}$$

$$Z(N_{ij},h)=\begin{cases}0, \text{路径与故障确定时飞行器不能通过}\\ 1, \text{路径与故障确定时飞行器能通过}\end{cases} \tag{6-3}$$

将$P(N_{ij},h)$作为一个新的目标函数添加到问题一的模型中，可以得到成功概率最高、航迹长度最短和经过的校正点个数最少的多目标随机约束模型。

由此，可以建立问题三的数学模型为：

$$\max\ P(N_{ij},h)$$

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}N_{ij}\cdot d(i,j)$$

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}N_{ij}$$

$$
s.t.\begin{cases}
\sum_{i=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} N_{ij} \leqslant 1 \\
\sum_{j=1}^{n} (N_{ji} - N_{ij}) = \begin{cases} -1, & i = 1 \\ 0, & i \in (1,n) \\ 1, & i = n \end{cases} \\
\delta \cdot [d(k,i) + d(i,n)] < \theta \\
\delta \cdot [d(k,i) + d(i,j)] \leqslant \alpha_1 \\
\delta \cdot d(i,j) \leqslant \alpha_2 \\
\delta \cdot d(j,l) \leqslant \beta_2 \\
\delta \cdot [d(i,j) + d(j,l)] \leqslant \beta_2
\end{cases}
$$

6.3 模型的求解及分析

6.3.1 算法流程

如图 6-1 所示的算法流程图中，程序的开始，输入成功到达终点的概率下限p_s，将所有有问题的校正点D_w的误差校正结果设为 5。如果$64\% < p_s \leqslant 80\%$时，循环遍历各个有问题的$D_w$，使其中一个$D_w$的误差校正结果为 0，如果$p_s \leqslant 64\%$我们采用蒙特卡洛随机模拟，使其中两个$D_w$的误差校正结果为 0，如果$80\% < p_s \leqslant 100\%$则不对$D_w$进行操作。当完成$D_w$的判断，我们采用优化的 Dijkstra 算法进行路径搜索和扩展，获得当前概率下限p_s下最少转折点的最短航迹。

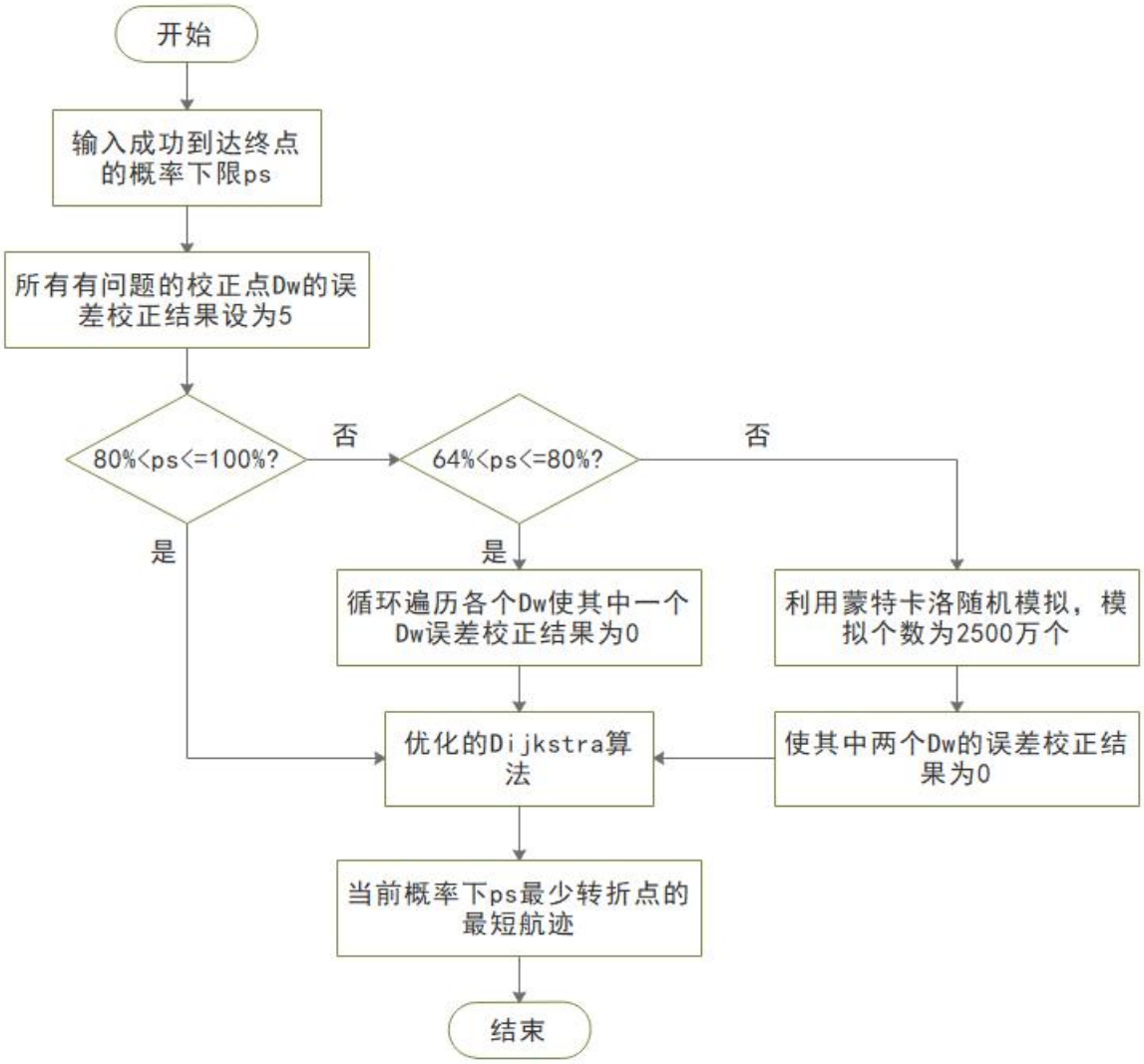

图 6-1 问题三的求解算法流程

6.3.2 问题三的求解及分析

定理：对于特定的N_{ij}输入，在所有问题校正点都校正失败的情况下不能通过的环节数是a个，则整体成功通过的可能性为0.8^a。

证明：整体成功通过的可能性如式(6-1)所示。其中的个别通过概率$Z(N_{ij},h)=0$，表示当前校正失败的环节中包括不能通过的环节，因此整体不能通过；$Z(N_{ij},h)=1$，表示当前校正失败的环节中不包括不能通过的环节。因为每个问题校正点失败的概率彼此独立，因此失败后不能通过的a个环节才决定了整体成功通过的可能性，只有当这a个环节都校正成功时，整体才能通过，而彼此独立的a个节点都成功的概率为$P=0.8^a$。$a=0$，1，2 时成功概率如图 6-2 所示。

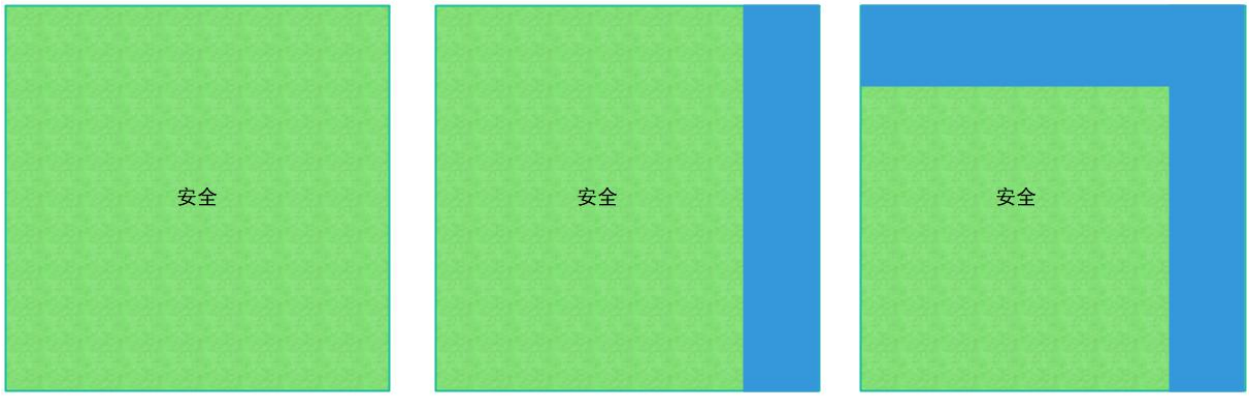

图 6-2 $a=0$，1，2 时的成功概率示意图

求解得到在不同的p_s值下，数据集 1 的可能结果如表 6-1 所示，其中黄色标记的是在当前概率p_s下的最优航迹。

表 6-1 数据集 1 在不同p_s值下的可能结果

不同 ps 下的可能结果	经过的转折点数	航迹长度(m)
100%	10	105189.50
80%	9	104239.07
	10	105166.52
	10	104562.94
	10	105443.41
	10	104757.22
	10	104367.02
	10	104984.86
64%	9	104065.88
	10	105270.21

表 6-2、表 6-3 列出了数据集 1 对于三种p_s值下的最优航迹的路径编号表和航迹规划的结果。表 6-3 中标记黄色的点表示必须校正成功的校正点。

表 6-2 数据集 1 三种p_s值对应最优航迹的路径编号表

经过转折点数	航迹长度(m)	路径编号
10(100%)	105189.50	A-->503-->69-->506-->371-->399-->194-->450-->113-->485-->302-->B
9(80%)	104239.07	A-->503-->69-->91-->607-->170-->278-->369-->214-->397-->B

9(64%)	104065.88	A-->503-->294-->91-->607-->170-->278-->369-->214-->397-->B

表 6-3 数据集 1 航迹规划结果表(经过校正点 9 个)

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	13.388	13.388	12
69	13.807	22.195	02
91	22.709	13.901	12
607	13.353	22.254	01
170	16.946	3.593	11
278	10.457	14.050	01
369	21.893	11.436	11
214	13.314	24.750	01
397	22.330	9.017	11
612	16.973	25.990	终点 B

数据集 1 在不同p_s条件下的最优航迹如图 6-3 所示。

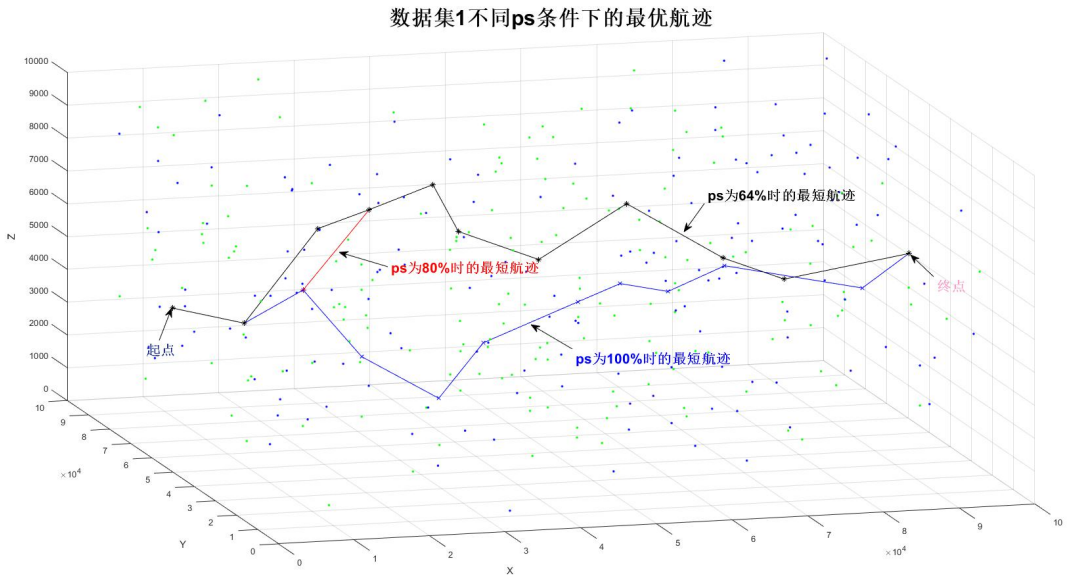

图 6-3 数据集 1 不同p_s条件下的最优航迹

求解得到在不同的p_s值下，数据集 2 的可能结果如表 6-4 所示。其中黄色标记的是在当前概率p_s下的最优航迹。

表 6-4 数据集 2 在不同p_s值下的可能结果

不同 ps 下的可能结果	经过的转折点数	航迹长度(m)
100%	21	165615.21
80%	19	147546.70
	19	145510.59
	20	159808.58
	20	162292.00
	20	158249.14
	20	161702.98
	20	164547.00
	21	164916.77
	21	165164.25
64%	16	128739.94
	16	129133.38
	18	139703.97
	18	139523.20
	18	141740.07
	19	146122.15
	19	146848.26
	19	147141.71
	19	148624.47
	19	144812.15
	20	162940.39
	20	165487.71
	20	155455.34
	21	165615.21

表 6-5、表 6-6 列出了数据集 2 对于三种p_s值下的最优航迹的路径编号表和航迹规划的结果。表 6-6 中标记黄色的点表示必须校正成功的校正点。

表 6-5 数据集 2 三种 p_s 值对应最优航迹的路径编号表

经过转折点数	航迹长度(m)	路径编号
21(100%)	165615.21	A-->169-->322-->270-->89-->236-->132-->53-->112-->268-->250-->243-->73-->249-->274-->12-->216-->279-->301-->38-->110-->99-->B
19(80%)	145510.59	A-->169-->322-->100-->137-->194-->205-->196-->119-->86-->167-->207-->44-->211-->321-->279-->301-->38-->110-->99-->B
16(64%)	128739.94	A-->163-->114-->234-->222-->8-->309-->305-->123-->231-->41-->221-->164-->50-->323-->61-->292-->B

表 6-6 数据集 2 航迹规划结果表(经过校正点 19 个)

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	9.271	9.271	01
322	13.419	4.148	11
100	11.180	15.328	01
137	17.014	5.834	11
194	9.881	15.715	02
205	17.423	12.542	12
196	8.583	16.125	02
119	15.705	12.122	12
86	12.791	19.913	01
167	16.559	3.768	11
207	7.821	11.589	01
44	15.175	7.354	11
211	5.497	12.851	01

321	13.247	5.497	11
279	10.360	18.110	02
301	15.313	9.953	11
38	9.872	19.825	01
110	13.799	3.927	11
99	9.200	13.127	01
326	17.850	8.650	终点 B

数据集 2 在不同p_s条件下的最优航迹如图 6-4 所示。

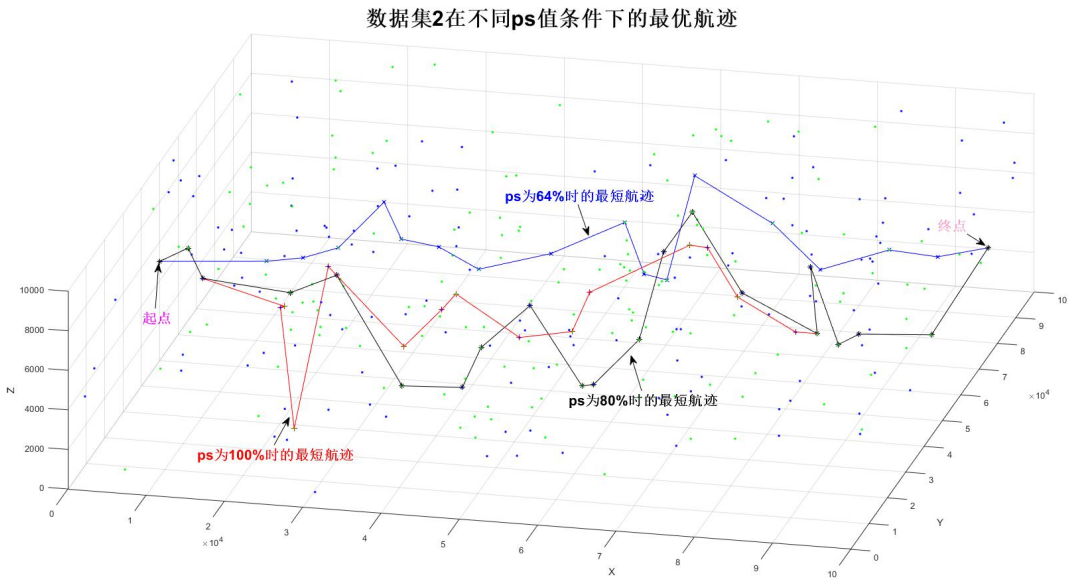

图 6-4 数据集 2 不同p_s条件下的最优航迹

7. 模型的评价

7.1 模型的优点

1.对于约束条件确定的情况可以精确得到最优解而且复杂度不高。
2.方便增加其他约束来拓展规划。

7.2 模型的缺点

1.无法得到精确的曲线最优解，这个也是受制于 3 维最短曲线求解的复杂性。
2.对于更低p_s下的最优解的搜索是通过随机模拟，大部分模拟都浪费在较高的p_s的情况中。

7.3 模型的展望

对于问题三的模型，由于添加了第三个规划目标函数，因此实际上交替序列未必是最优。对于交替序列的期望，为0.8^{α}个，而对于非交替序列，是可以做到期望不是在这些范围，比如对于一个只有一个出问题时无法通过的校正点的轨迹，在交替时通过期望为0.8，而在非交替情况下，如果满足

1.它的下一个校正点是同类型校正点；

2.它的下一个校正点也是一个问题校正点并且没有出问题；

3.在校正失败后也可以到达下个校正点

则整体的成功概率为0.96。即对于第三个规划目标带来的区别考虑的不够仔细，导致没有及时发现更多的可能性。

参考文献

[1] Tsourdos A , White B , Shanmugavel M . 无人机协同路径规划[M]. 国防工业出版社, 2013.

[2] Trajectory tracking for unicycle-type and two-steering wheelsmobile robots. Miacelli,A,Samson, C. Technical Report No.2097,INRIA . 1993

[3] 王明明. 复杂环境下无人飞行器航路规划技术研究[D].中国舰船研究院,2016.

[4]董世建. 复杂约束条件下航迹规划方法研究[D].北京理工大学,2016.

[5] 辛培源. 基于三维环境复杂约束条件的无人机航迹规划方法研究[D].首都师范大学,2014.

[6]郑昌文. 飞行器航迹规划方法研究[D].华中科技大学,2003.

[7]寇家勋. 不确定环境下无人机航迹规划研究[D].北京理工大学,2016.

附录

```
1.  import com.sun.istack.internal.NotNull;
2.
3.  import java.io.BufferedReader;
4.  import java.io.FileReader;
5.  import java.io.IOException;
6.  import java.util.*;
7.
8.  public class DY {
9.      public static void main(String[] args) throws IOException {
10.         long begin = System.currentTimeMillis();
11.         BufferedReader in = new BufferedReader(new FileReader("c:\\Users\\GZN\\Desktop\\4.txt"));
12.         String line = null;
13.         int N = 327;
14.         StringBuffer str = new StringBuffer();
15.         double[][] zb = new double[N][3];
16.         String[] zhuanzhe = new String[N];
17.         double[][] zb_jl = new double[N][N];
18.         int b1 = 15000;
19.         int b2 = 20000;
20.         int a1 = 20000;
21.         int a2 = 10000;
22.         int c = 20000;
23.         int hang = 0;
24.         final double INF = Double.MAX_VALUE;
25.         while ((line = in.readLine()) != null) {
26.             String[] temp = line.split("\\t");
27.             zb[hang][0] = Double.parseDouble(temp[0]);
28.             zb[hang][1] = Double.parseDouble(temp[1]);
29.             zb[hang][2] = Double.parseDouble(temp[2]);
30.             zhuanzhe[hang] = temp[3];
31.             hang++;
32.         }
33.         in.close();
34.         for (int i = 0; i < N; i++) {
35.             for (int j = 0; j < N; j++) {
36.                 zb_jl[i][j] = Math.sqrt(Math.pow(zb[i][0] - zb[j][0], 2) + Math.pow(zb[i][1] - zb[j][1], 2) + Ma
    th.pow(zb[i][2] - zb[j][2], 2));
37.                 if (zb_jl[i][j] > c || zhuanzhe[i].equals(zhuanzhe[j])
38.                         || (zhuanzhe[i].equals("1") && zhuanzhe[j].equals("0")) && (zb_jl[i][j] > b1)
39.                         || ((zhuanzhe[i].equals("0") && zhuanzhe[j].equals("1")) && (zb_jl[i][j] > a2))
40.                         || (zhuanzhe[i].equals("A 点") && zhuanzhe[j].equals("0") && zb_jl[0][j] > b1)
41.                         || (zhuanzhe[i].equals("A 点") && zhuanzhe[j].equals("1") && zb_jl[0][j] > a2)) {
```

```
                zb_jl[i][j] = INF;
            }
        }
    }
    Vertex[] vertexes = new Vertex[N];
    for (int i = 0; i < vertexes.length; i++) {
        vertexes[i] = new Vertex(i);
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if(zb_jl[i][j]!=INF){
                vertexes[i].addNeighbor(vertexes[j]);
            }
        }
    }
    // start search
    Vertex start = vertexes[0];
    Vertex end = vertexes[N-1];
    boolean hasPath = bfs(start, end,zb_jl);
    double sss = 0.0;
    if (hasPath) {
        // print path
        Vertex predecessor = end;
        StringBuffer strb=new StringBuffer();
        do {
            strb.append(predecessor.getId()).append("<--");
            if(predecessor.getPredecessor()!=null){
                sss+=zb_jl[predecessor.getPredecessor().getId()][predecessor.getId()];
            System.out.println(zb_jl[predecessor.getPredecessor().getId()][predecessor.getId()]);}
            predecessor = predecessor.getPredecessor();
        } while (predecessor != null);
        System.out.println(sss);
        System.out.println(strb);
    } else {
        System.out.println("不存在该起点到该终点的路径");
    }
    int[] vexs = new int[N];
    for (int j = 0; j < N; j++) {
        vexs[j] = j;
    }
    Dijkstra dijkstra = new Dijkstra(N);
    dijkstra.dijkstra(zb_jl, vexs, 0);
    in.close();
    long endd =System.currentTimeMillis();
```

```
        System.out.println(endd-begin);
    }

    public static boolean bfs(Vertex start, Vertex end,double[][] zb_jl) {
        int iii =0;
        start.setDeepth(0);
        Queue<Vertex> queue = new LinkedList<>();
        queue.add(start); // 添加到队尾
        while (!queue.isEmpty()) {
            Vertex vertex = queue.poll(); // 从队首取出一个元素
            // 如果这个顶点是否已经检索过了，则跳过
            if (vertex.isSerched()){
                continue;
            }
            if(vertex == end){
                return true;
            } else {
                if (vertex.getPredecessor()==null){
                    for (int i = 0; i < vertex.getNeighbors().size(); i++) {
                        vertex.getNeighbors().get(i).setPredecessor(vertex);
                        vertex.getNeighbors().get(i).setDeepth(1);
                        queue.add(vertex.getNeighbors().get(i));
                    }
                    vertex.setSerched(true);
                }else {
                    for (int i = 0; i < vertex.getNeighbors().size(); i++) {
                        if (vertex.getNeighbors().get(i).isSerched()){
                            continue;
                        }
                        iii=0;
                        if (vertex.getNeighbors().get(i)==end){
                            iii=1;
                             if(((zb_jl[vertex.getPredecessor().getId()][vertex.getId()]+zb_jl[vertex.getId()][ve
rtex.getNeighbors().get(i).getId()])<25000+5000*iii)){
                                vertex.getNeighbors().get(i).setPredecessor(vertex);
                                queue.add(vertex.getNeighbors().get(i));
                                return true;
                            }
                        }
                         if((zb_jl[vertex.getPredecessor().getId()][vertex.getId()]+zb_jl[vertex.getId()][vertex.
getNeighbors().get(i).getId()]<25000)){
                            if (!queue.contains(vertex.getNeighbors().get(i))) {
                                vertex.getNeighbors().get(i).setPredecessor(vertex);
                                vertex.getNeighbors().get(i).setDeepth(vertex.getDeepth()+1);
```

```
                                queue.add(vertex.getNeighbors().get(i));
                            }else{
                                if (vertex.getNeighbors().get(i).getDeepth()==vertex.getDeepth()+1) {
                                    //if (vertex.getNeighbors().get(i).getPredecessor() != vertex.getPredecessor
    () {
                                        if ((zb_jl[vertex.getId()][vertex.getNeighbors().get(i).getId()]) < (zb_
    jl[vertex.getNeighbors().get(i).getId()][vertex.getNeighbors().get(i).getPredecessor().getId()])) {
                                            vertex.getNeighbors().get(i).setPredecessor(vertex);
                                        }
                                    //}
                                }
                            }
                        }
                        vertex.setSerched(true);
                    }
                    }
                }

            return false;
        }

}
```