西北工業大學

NORTHWESTERN POLYTECHNICAL UNIVERSITY

复杂空域下无人机路径规划

姓　　名　　路为民

指导老师　　宗亚雳

院　　系　　电子信息学院

完成日期　　2022年12 月 30 日

目录

复杂空域下无人机路径规划

一、引言

无人机作为现代科技产物之一，其研究与应用也随着现代科技的发展而突飞猛进。目前无人机飞行器种类日渐多元化，应用日渐广泛，著名的有专门用作植保的大疆 PS-X625 无人机，用作街景拍摄与监控巡察的宝鸡行翼航空科技的 X8 无人机，以及用作水下救援的白鲨 MIX 水下无人机等。随着无人机成本不断降低，种类与功能不断丰富，其在未来应用的前景也将不断扩大。因此，研究无人机在一些复杂空域情况下的飞行问题极具现实意义和应用价值。

在无人机的一些应用领域，飞行过程中所遇到的障碍物往往是复杂的，并且无人机所处的环境也是多变的，为了确保无人机自身的安全，并且节约机载能源的消耗，在复杂环境中规划出一条最安全、最直接的路径是很有必要的，一个好的路径规划会给无人机的性能带来很大方面的提升，并且能够有效保障无人机的飞行安全，大大降低无人机飞行的时间成本，进而可以提升无人机完成任务的效率。

本文作为《最优化理论与方法》这门课的结课作业，就以无人机在复杂空域下的飞行这一情景作为研究背景，通过使用蚁群算法这种路径规划经典算法来进行无人机的最短路径规划，并且通过 Matlab 软件进行计算与仿真，将结果可以直观地展示出来，验证所用算法的可行性。

图：DJI 无人机

二、问题描述与简化

对于无人机飞行过程中复杂的现实情况，本文会对其进行必要的简化与抽象，以避免在无谓的节点上浪费过多的口舌，同时本文会对所作简化与抽象作出一定解释，以确保他们均是必要且合理的。

简化 1：无人机自身处理为点目标。

无人机的自身尺寸有小有大，但是相对于他们所处空域来说自身尺寸以及飞行过程中所遇障碍则可以忽略不计，视为点目标处理。

简化 2：障碍归一化。

无人机在飞行中可能遇到各种各样的障碍，这些障碍有可能是来自陆地的树木、山体、房屋，也有可能是来自空中的气流等，但是他们有一个共同的本质属性就是会“阻碍”无人机的飞行，本文将这种阻碍作用进行归一化，均为“不可穿越”，都需要进行躲避（或称为确保飞行最安全目标）

简化 3：全局环境已知。

实际环境中确实存在一些不可预知的突发障碍，但是这种随机情况不在本文研究范围内，这也是本文的缺陷之一，但是本文会在已知全局环境研究结束后，尝试在此环境上构建一些随机障碍来进行拓展。

简化 4：最短路径原则

适于目标空域的飞行路径有可能有很多条，不同的飞行目的选择的原则各有不同，本文选取最常用的“最短路径原则”。

基于以上简化，我们现在可以清晰地描述这个问题：在一个 3D 地图上，寻找从一点到零一点的最短路径。为了能够使用 Matlab 进行仿真与验证，我们给予这个问题一个实际的背景：在 21*21*5000 的空间中，由起始坐标(1,10,800)至终点坐标(21,8,1200)。

三、建模与模型分析

3.1 全局环境建模

通过上面的分析，我们要建立的是一个障碍物既可能在顶部也有可能在底部的环境，与实际中的“山洞”十分相似，但是直接建立这样吧的山洞模型却十分困难，因为相当于在(x,y)平面上任意一点都会有两个不同的 z 取值(底部障碍最高点和顶部障碍最低点，如图 3.1.1 所示的 a、b 点)。为此，我们将其切为上下两个空间，在两个空间中分别进行求解，最后再对得到的结果进行综合。

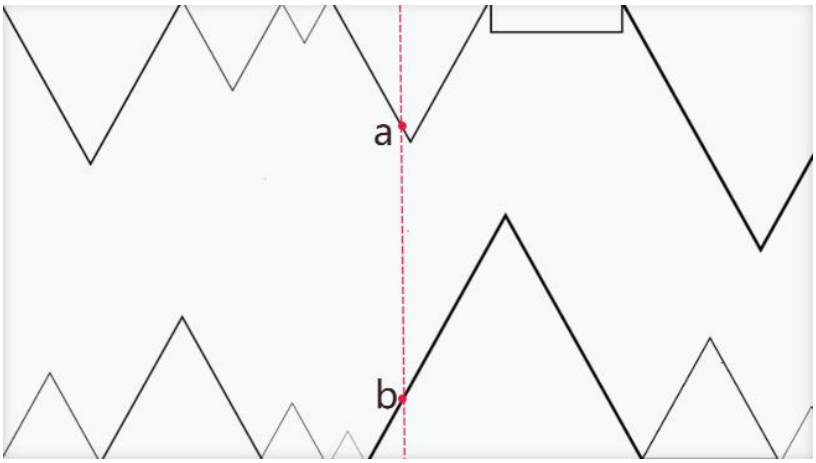


图 3.1.1：模型 z 向切面图

以下为生成的两部分环境：

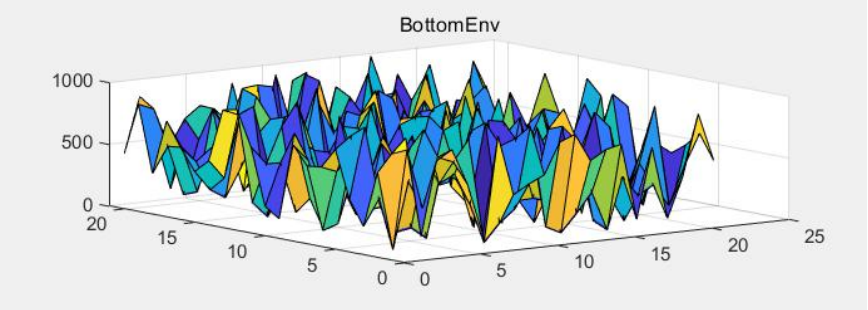


图 3.1.2：底部环境三维图

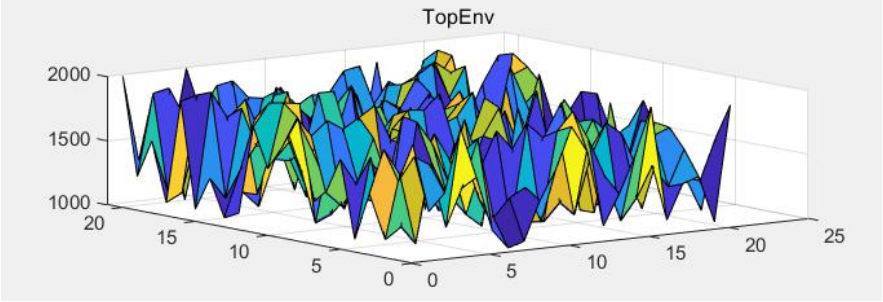


图 3.1.3：顶部环境生成图

3.2 目标函数

上文已经提及，该问题可简洁地描述为“一点到另一点的最短路径”，并考虑到我们是分为两部分来解决，由此我们可以得到目标函数为：

$$\min L_1 = \sum \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}$$

$$\min L_2 = \sum \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}$$

式中，(x_i,y_i,z_i)表示路径中的各个点的坐标。

3.3 模型分析

该模型为路径规划问题中常用的到的模型，是非线性、无约束且具有多个极值点的模型。对于固定的起点与终点，有多条最短的路径的情况是存在的，并且我们已经将这个问题一分为二，所以我们最终得到的最优路径一定会有两条：一条是来自顶部环境(TopEnv)中的，另一条则是来自底部环境(BottomEnv)中的,对于两个路径的取舍，存在多种原则，比如可以取两个中更短的(坚持最短原则)，或者取两个中平均高度较低的(换为低空原则)。

四、算法选择与理论描述

4.1 算法选择

目前关于路径规划的算法十分多样，著名的有 Voronoi 图法、遗传算法、粒子群优化算法、人工势场法、 A*算法等，这些算法虽然核心思想虽然各有千秋，但是具有共同的缺点，就是对于数据规模十分大的问题，其要求的运算能力很高，占用的资源很大，并且所用时间也不容乐观。

就拿 Voronoi 图法来说，作为路径规划问题中的一种经典空间分割算法，他被广泛应用于安全行驶情景中(Voronoi Planner)。通过一系列种子节点(Seed Points)将一个区域分割为众多的子区域，然后通过是种子节点间距最大来确保规划的路径远离所有障碍物，如下图所示。但是这种算法对于三维情景的处理却显得十分乏力。

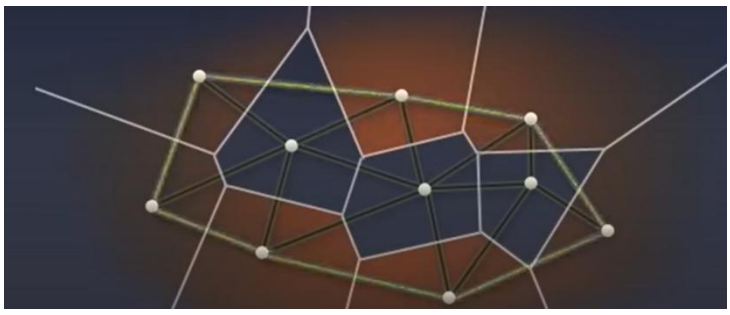


图 4.1.1：Voronoi Planner 示例

蚁群算法由于其具有分布式计算能力，群体智能、正反馈等优点，是一个比较适合解决这种三维路径问题的算法，因此本文是采用的蚁群算法。

4.2 蚁群算法

蚁群算法（Ant Clony Optimization， ACO）是一种群智能算法，它是由一群无智能或有轻微智能的个体（Agent）通过相互协作而表现出智能行为，从而为

求解复杂问题提供了一个新的可能性。蚁群算法最早是由意大利学者 Colorni A., Dorigo M. 等于 1991 年提出。经过 20 多年的发展，蚁群算法在理论以及应用研究上已经得到巨大的进步。

蚁群算法是一种仿生学算法，是由自然界中蚂蚁觅食的行为而启发的。在自然界中，蚂蚁觅食过程中，蚁群总能够按照寻找到一条从蚁巢和食物源的最优路径。下图显示了这样一个觅食的过程。

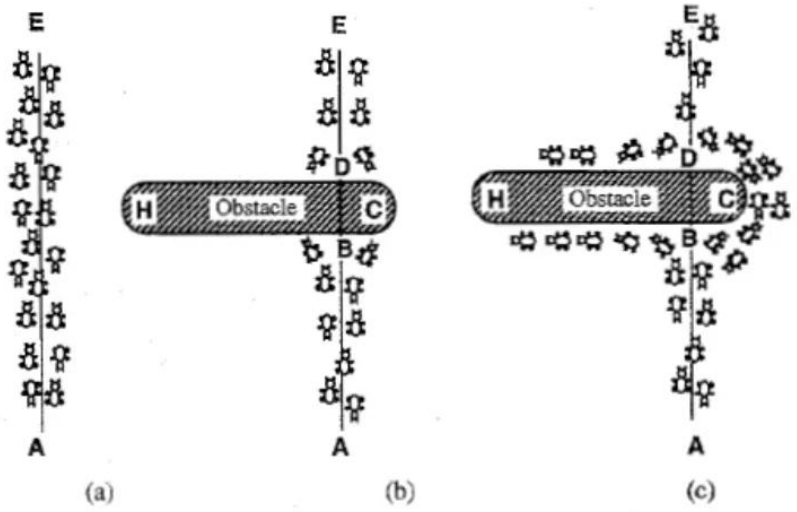


图 4.2.1：蚁群算法示意图

在图（a）中，有一群蚂蚁，假如 A 是蚁巢，E 是食物源（反之亦然）。这群蚂蚁将沿着蚁巢和食物源之间的直线路径行驶。假如在 A 和 E 之间突然出现了一个障碍物（图（b）），那么，在 B 点（或 D 点）的蚂蚁将要做出决策，到底是向左行驶还是向右行驶？由于一开始路上没有前面蚂蚁留下的信息素（pheromone），蚂蚁朝着两个方向行进的概率是相等的。但是当有蚂蚁走过时，它将会在它行进的路上释放出信息素，并且这种信息素会议一定的速率散发掉。信息素是蚂蚁之间交流的工具之一。它后面的蚂蚁通过路上信息素的浓度，做出决策，往左还是往右。很明显，沿着短边的的路径上信息素将会越来越浓（图（c）），从而吸引了越来越多的蚂蚁沿着这条路径行驶。

4.3 算法流程

蚁群算法总的来说可以分成四个大致步骤：

(1) 初始化参数

正如其他算法所需要的，在计算之初，蚁群算法需要对相关参数进行初始化，如蚁群规模(蚂蚁数目)、信息素重要程度因子(常用α来表示)、启发函数重要程度因子(常用β来表示)、信息素挥发因子(常用ρ来表示)、信息素释放总量、最大迭代次数、迭代次数初始值(通常取 1)等。

(2) 构建解空间

将蚂蚁随机放置在不同的节点，对每个蚂蚁计算其下一个目的地，直到所有蚂蚁访问完成所有目的地。

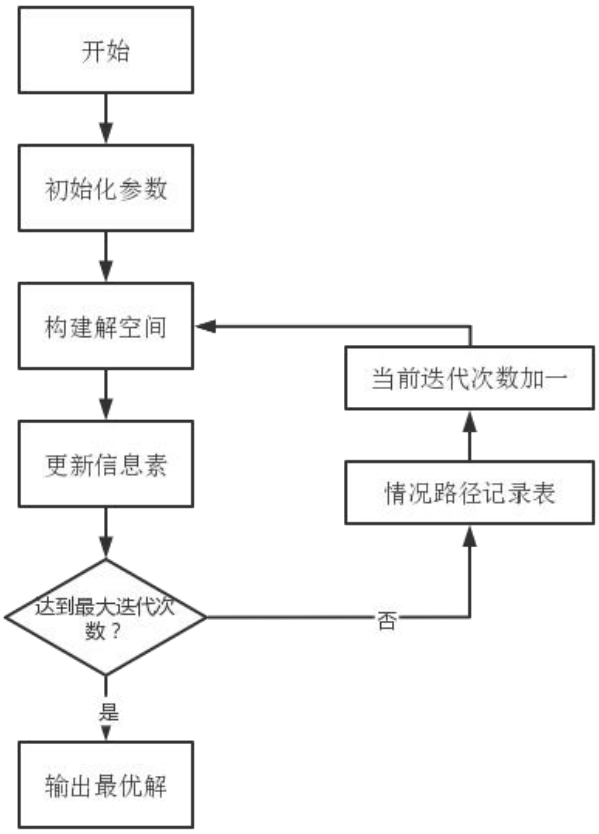
(3) 更新信息素

计算各个蚂蚁的路径长度，记录当前迭代次数中最优解，同时对于信息素浓度进行更新。

(4) 判断是否终止

根据当前迭代次数与允许的最大迭代次数比较来决定是进行下一次迭代还是退出并返回最优解。

以上的过程在下面的流程图中可以很直观地体现：



4.4 算法实现

4.4.1 信息素更新

根据蚁群算法原理，信息素是蚁群在觅食过程中对蚂蚁产生吸引作用的信息载体，在一定程度上对算法的收敛速度和路径规划效果有着十分重要的影响。信息素主要留在路径上，与寻找路径构造图的边相对应，在三维空间内进行路径规划时，会造成庞大的计算量。因此本文采用环境模型中的离散点作为载体，将信息素存储到各个离散点上，在节省存储空间的同时，降低计算的复杂度。在对无人机进行路径规划的过程中，需要对信息素进行更新，信息素更新分为局部信

息素更新和全局信息素更新。局部信息素更新是指当无人机每经过一个路径点时，该路径点上的信息素浓度会相对减少，其目的是为了减少其他无人机选择该点的可能性，增加无人机发现更多路径的机会，避免算法陷入局部最优。局部信息素更新公式如下：

$$\varepsilon_{xyz} = (1-\delta)\,\varepsilon_{xyz}$$

其中，ε_{xyz}表示点（x,y,z）上的信息素的值，$\delta(\delta<1)$表示信息素衰减率系数。

全局信息素更新就是当无人机从起点飞到终点规划一条飞行路径时，以无人机规划的路径长度作为评价来判断信息素的多少，然后从路径集合中选择最优路径，对该路径上各点的信息素进行更新。其更新规则如下：

$$\varepsilon_{xyz} = (1-\rho)\,\varepsilon_{xyz} + \rho\Delta\varepsilon$$

$$\varepsilon_{xyz} = \frac{C}{\min(\text{length(i)})}$$

其中， $\Delta\varepsilon$表示信息素增量，*length*(i) 表示第 i 个蚂蚁走完路径时的路径长度， ρ 表示信息素更新系数，C 表示常数。

4.4.2 启发式函数确定

启发式函数设计的目的是利用启发信息来引导无人机寻找从起点到终点的最优路径。根据蚁群算法的原理，当无人机从当前一个飞行节点飞向下一个飞行节点的时候，需要根据启发式函数来计算无人机在当前飞行节点飞向空间中其他节点的概率。启发式函数为：

$$H(x,y,z) = D_1(x,y,z) \times T(x,y,z) \times D_2(x,y,z)$$

其中，$D_1(x,y,z)$表示当前点飞行节点到下一个飞行节点的距离，主要用来引导无人机飞向距离当前节点较近的下一个节点；$T_1(x,y,z)$ 表示威胁代价函数，主要用来引导无人机飞向安全的区域； $D_2(x,y,z)$表示无人机从下个飞行节点飞向终点的距离，主要用来引导无人机飞向路径规划的终点。

上式中， $D_1(x,y,z)$ 的表达式如下：

$$D_2(x,y,z) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$

其中x_i, y_i, z_i表示当前点在在该坐标系下的坐标，而$x_{i+1}, y_{i+1}, z_{i+1}$下一个节点的坐标。

T(x, y, z)的表达式如下:

$$T(x,y,z)=\frac{(n_r-n_{ur})}{n_r}$$

其中，n_r 表示在点(x,y,z) 时无人机飞行空间内可通过节点的数量；n_{ur} 表示无人机飞行空间内不可通过的节点的数量。

$D_2(x,y,z)$的表达式如下:

$$D_2(x,y,z)=\sqrt{(x_{i+1}-x_d)^2+(y_{i+1}-y_d)^2+(z_{i+1}-z_d)^2}$$

其中 x_d，y_d，z_d 表示终点的坐标。

4.4.3 算法总体描述

本文首先对无人机飞行环境进行建模，划分飞行空间中的可通过区域以及不可通过区域，构建三维环境模型，然后确定信息素的选取方式，通过路径点选择，确定启发式函数。本文的算法步骤如下：

（1）通过 3.1 节中的三维环境模型构建无人机路径规划的三维飞行空间，根据 飞行任务计划以及环境模型，选取合适的点作为路径规划的起点和终点。

（2）将蚁群中所有蚂蚁放置在设定的路径规划的起点，将 x 轴作为蚂蚁进行搜索的主运动方向。

（3）初始化蚁群算法中各个模型的运行参数。

（4）开始对三维路径进行搜索，更新路径点上的信息素浓度，对可通过区域采用启发函数进行计 算，同时根据计算结果选择下一个需要通过的路径节点。

（5）放置所有蚂蚁到计算的下一个路径节点，对该点的信息素浓度进行局部更新，并对更新的结果结果存储到内存中。

（6）判断规划的路径中的所有蚂蚁是否全部到达终点，是，继续执行，否，返回到步骤 3。

（7）当蚂蚁执行一次路径搜索后，采用信息素全局更新模型对路径上的信息素进行全局更新，并判断算法是否满足结束条件，是，则输出最优解，否，返回到步骤 3 继续执行。

五、结果分析

本文采用的是 Matlab2019 软件进行编程并且仿真，使用下表中的参数：

参数表

参数	值
蚂蚁数量	10
信息素更新系数	0.2
信息素衰减系数	0.5
迭代次数	300

得到仿真结果如下,虚线为路径，其中俯视图更加容易查看：

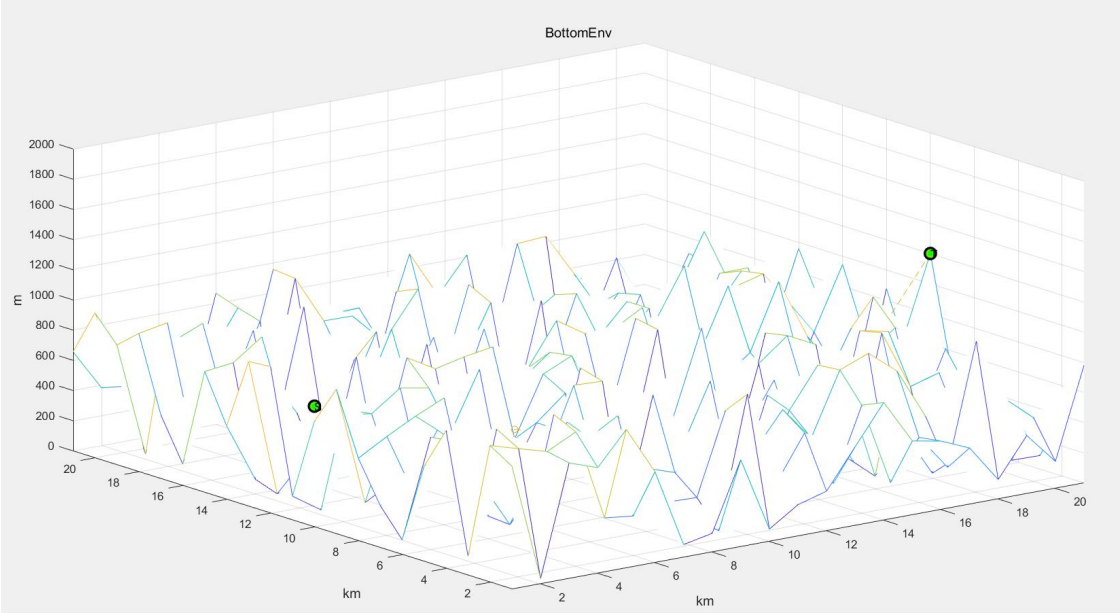


图 1:BottomEnv 斜视图

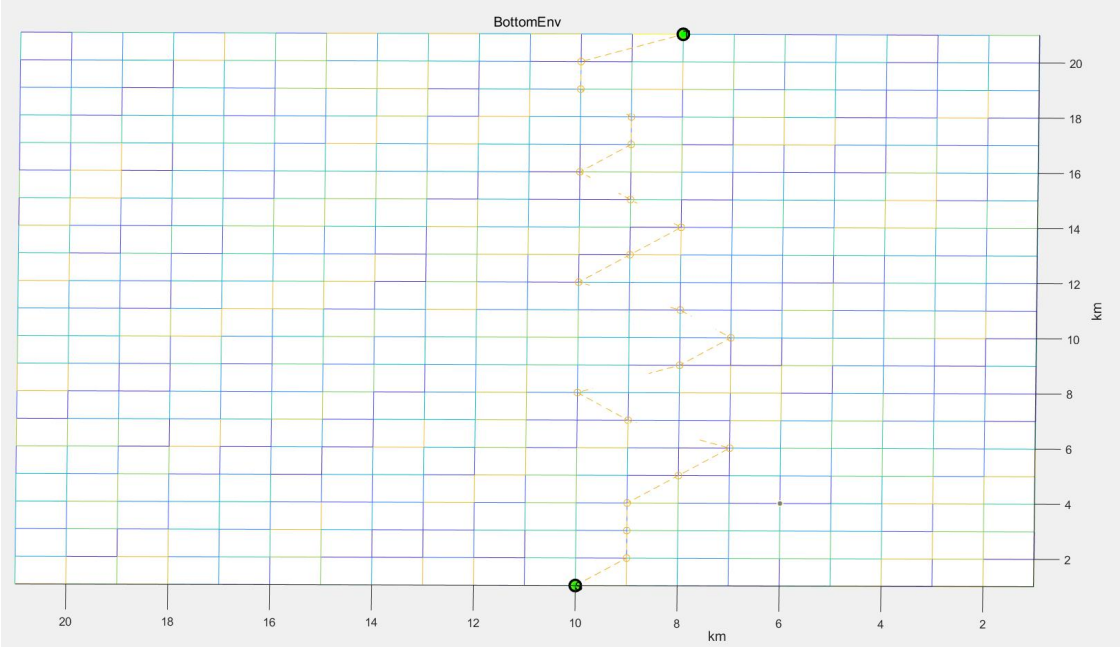


图 2:BottomEnv 俯视图

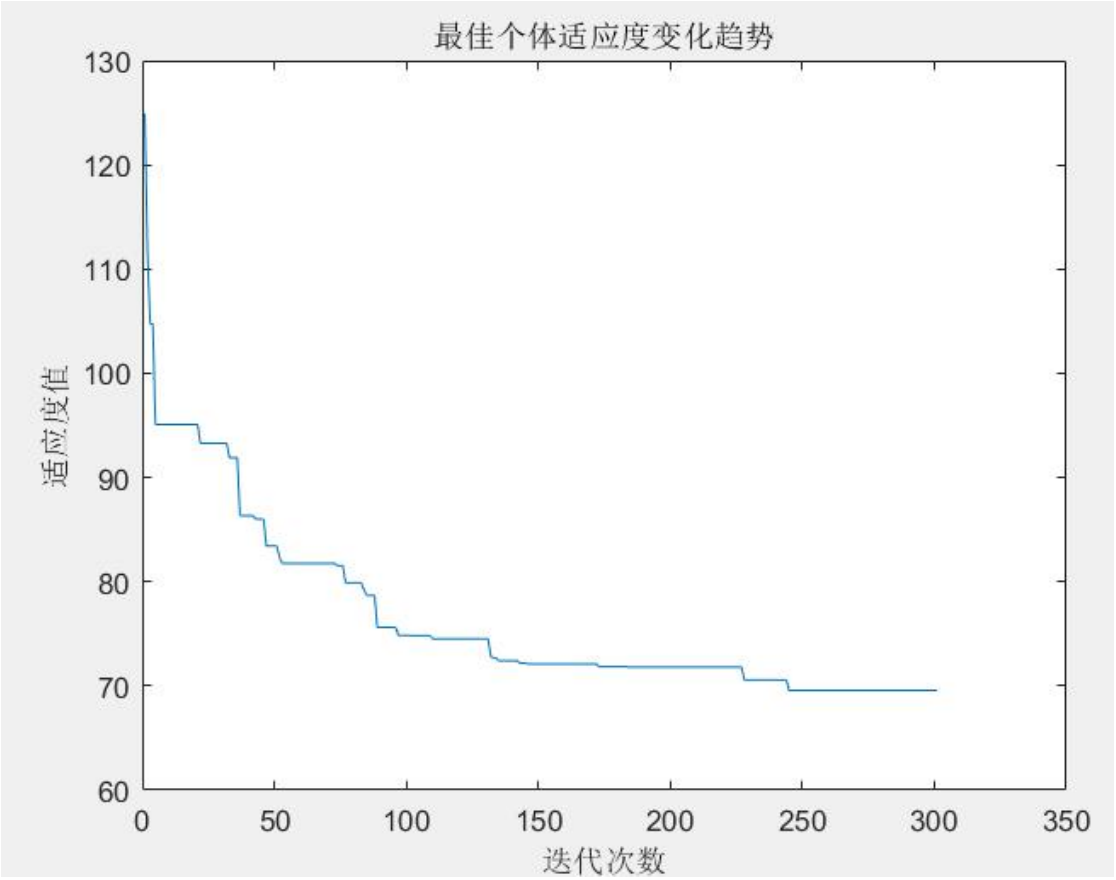


图 3:BottomEnv 适应度曲线

顶部环境中仿真过程相同，不再赘述。

六、结语

首先想回答一下宗老师在第一堂课上提出的问题，《最优化理论与方法》这门课是学长给我推荐让我务必毕业前要上一次，学长说在这堂课中可以学到很多的东西。现在，在课程的结尾，我想对我学到的东西做一个总结，一是为了巩固，另一个则是向学弟学妹们推荐的时候有话可说：

系统规范的思维。之前自己也有跟别人做过数模竞赛，在做的过程中老是感觉诸多头绪无从下手，并且各个成员众说纷纭，经常出现的情况就是大家一直在说，总体进度却一直没推进多少。在这堂课中，宗老师一直要求我们无论是上黑板做题还是自己在下面写，一定要按照流程和规范写，比如下面这种(随手写的)：

$$\min z = 4x_1 + 8x_2 + 6x_3$$

$$\begin{aligned} \text{s.t.} \quad & x_1 + x_2 = 10 \\ & x_2 + x_3 = 20 \\ & x_3 + x_4 = 15 \end{aligned}$$

不知到老师是否还记得我第一次上黑板做题的情景，当时真的是想到什么写什么，但是现在我也养成了这么一个习惯，现在我觉得拿到一个问题后的思路比以前清晰很多，表面上收获的是解题步骤，实际上收获的却是系统的思维，我相信后者在我之后的学术生涯中会充当不可或缺的角色，这也是我学这门课最大的收获，因此我把它放在了第一个。

理论与实践的融合。不知道是因为课时原因还是其他原因，之前的专业课总是停留在理论层面，虽然确实有些理论确实十分美妙，但是久之难逃枯燥乏味的命运。在最优化的课堂上，我感受到另一种学习的方法，讲完就用 Matlab 进行实现，让我觉得这门课甚至有一些“实验课”的味道，我相信某些专业课能够做到这种效果挂科率一定不是现在这样。

经过一次又一次地修改，这份课程作业也终于完成了，在文章的最后，再次感谢宗亚雳老师一个学期的教导与陪伴，提前祝老师新年快乐！

七、附录(源码)

```
% ./code/GenEnvData/gen.m
% 该文件用于生成随机环境数据并保存

% 生成随机的障碍物高度
Z1 = randi(1000,21,21);
Z2 = randi(1000,21,21);
Z2 = Z2 + 1000;
% 建立 X Y Z 坐标系
[X, Y] = meshgrid([1:21],[1:21]);
% 画图
subplot(211);
surf(X, Y, Z1);
title('BottomEnv');
subplot(212);
surf(X, Y, Z2);
title('TopEnv');
%保存
save('TopEnv','Z1');
save('BottomEnv','Z2');
```

```
% ./code/Simulation/main.m
%% 主函数入口，进行三维路径规划

%% 清空环境
clc
clear

%% 数据初始化

%下载数据
load  HeightData HeightData

%网格划分
LevelGrid=10;
PortGrid=21;

%起点终点网格点
starty=10;starth=4;
endy=8;endh=6;
m=1;
%算法参数
PopNumber=10;          %种群个数
BestFitness=[];     %最佳个体

%初始信息素
pheromone=ones(21,21,21);

%% 初始搜索路径
[path,pheromone]=searchpath(PopNumber,LevelGrid,PortGrid,pheromone, ...
    HeightData,starty,starth,endy,endh);
fitness=CacuFit(path);                              %适应度计算
[bestfitness,bestindex]=min(fitness);               %最佳适应度
```

```
bestpath=path(bestindex,:);                    %最佳路径
BestFitness=[BestFitness;bestfitness];         %适应度值记录

%% 信息素更新
rou=0.2;
cfit=100/bestfitness;
for i=2:PortGrid-1
    pheromone(i,bestpath(i*2-1),bestpath(i*2))= ...
        (1-rou)*pheromone(i,bestpath(i*2-1),bestpath(i*2))+rou*cfit;
end

%% 循环寻找最优路径
for kk=1:300
    %% 路径搜索
    [path,pheromone]=searchpath(PopNumber,LevelGrid,PortGrid,...
        pheromone,HeightData,starty,starth,endy,endh);

    %% 适应度值计算更新
    fitness=CacuFit(path);
    [newbestfitness,newbestindex]=min(fitness);
    if newbestfitness<bestfitness
        bestfitness=newbestfitness;
        bestpath=path(newbestindex,:);
    end
    BestFitness=[BestFitness;bestfitness];

    %% 更新信息素
    cfit=100/bestfitness;
    for i=2:PortGrid-1
        pheromone(i,bestpath(i*2-1),bestpath(i*2))=(1-rou)* ...
            pheromone(i,bestpath(i*2-1),bestpath(i*2))+rou*cfit;
    end
end

%% 最佳路径
for i=1:21
    a(i,1)=bestpath(i*2-1);
    a(i,2)=bestpath(i*2);
end
figure(1)
x=1:21;
y=1:21;
[x1,y1]=meshgrid(x,y);
mesh(x1,y1,HeightData)
axis([1,21,1,21,0,2000])
hold on
k=1:21;
plot3(k(1)',a(1,1)',a(1,2)'*200,'--o','LineWidth',2,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',10)
plot3(k(21)',a(21,1)',a(21,2)'*200,'--o','LineWidth',2,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',10)
                    text(k(1)',a(1,1)',a(1,2)'*200,'S');
text(k(21)',a(21,1)',a(21,2)'*200,'T');
xlabel('km','fontsize',12);
```

```
ylabel('km','fontsize',12);
zlabel('m','fontsize',12);
title('BottomEnv','fontsize',12)
set(gcf, 'Renderer', 'ZBuffer')
hold on
plot3(k',a(:,1)',a(:,2)'*200,'--o')

%% 适应度变化
figure(2)
plot(BestFitness)
title('最佳个体适应度变化趋势')
xlabel('迭代次数')
ylabel('适应度值')
```

```
% ./code/Simulation/CacuFit.m
function fitness=CacuFit(path)
%% 该函数用于计算个体适应度值
%path       input     路径
%fitness    input     路径
[n,m]=size(path);
for i=1:n
    fitness(i)=0;
    for j=2:m/2
        %适应度值为长度加高度
        fitness(i)=fitness(i)+sqrt(1+(path(i,j*2-1)-path(i,(j-1)*2-1))^2 ...
            +(path(i,j*2)-path(i,(j-1)*2))^2)+abs(path(i,j*2));
    end
end
```

```
% ./code/Simulation/CacuQfz.m
function qfz=CacuQfz(Nexty,Nexth,Nowy,Nowh,endy,endh,abscissa,HeightData)
% 该函数用于计算各点的启发值
%Nexty Nexth    input    下个点坐标
%Nowy Nowh      input    当前点坐标
%endy endh      input    终点坐标
%abscissa       input    横坐标
%HeightData     input    地图高度
%qfz            output   启发值

% 判断下个点是否可达
if HeightData(Nexty,abscissa)<Nexth*200
    S=1;
else
    S=0;
end

% 计算启发值
%D距离
D=50/(sqrt(1+(Nowh*0.2-Nexth*0.2)^2+(Nexty-Nowy)^2)+sqrt((21-abscissa)^2 ...
    +(endh*0.2-Nexth*0.2)^2+(endy-Nowy)^2));

%计算高度
M=30/abs(Nexth+1);
%计算启发值
qfz=S*M*D;
```

```
end
```

```
% ./code/Simulation/searchpath.m
function
[path,pheromone]=searchpath(PopNumber,LevelGrid,PortGrid,pheromone,HeightData,st
arty,starth,endy,endh)
%% 该函数用于蚂蚁蚁群算法的路径规划
%LevelGrid      input     横向划分格数
%PortGrid       input     纵向划分个数
%pheromone      input     信息素
%HeightData     input     地图高度
%starty starth  input     开始点
%path           output    规划路径
%pheromone      output    信息素
%% 搜索参数
ycMax=2;    %蚂蚁横向最大变动
hcMax=2;    %蚂蚁纵向最大变动
decr=0.5;   %信息素衰减概率
%% 循环搜索路径
for ii=1:PopNumber
    path(ii,1:2)=[starty,starth];   %记录路径
    NowPoint=[starty,starth];       %当前坐标点
    %% 计算点适应度值
    for abscissa=2:PortGrid-1
        %计算所有数据点对应的适应度值
        kk=1;
        for i=-ycMax:ycMax
            for j=-hcMax:hcMax
                NextPoint(kk,:)=[NowPoint(1)+i,NowPoint(2)+j];
                if (NextPoint(kk,1)<PortGrid)&&(NextPoint(kk,1)>0)&&
(NextPoint(kk,2)<LevelGrid)&&(NextPoint(kk,2)>0)

 qfz(kk)=CacuQfz(NextPoint(kk,1),NextPoint(kk,2),NowPoint(1),NowPoint(2),endy,en
dh,abscissa,HeightData);

 qz(kk)=qfz(kk)*pheromone(abscissa,NextPoint(kk,1),NextPoint(kk,2));
                    kk=kk+1;
                else
                    qz(kk)=0;
                    kk=kk+1;
                end
            end
        end
        %选择下个点
        sumq=qz./sum(qz);
        pick=rand;
        while pick==0
            pick=rand;
        end
        for i=1:25
            pick=pick-sumq(i);
            if pick<=0
                index=i;
                break;
            end
        end
        oldpoint=NextPoint(index,:);
```

```
        %更新信息素

 pheromone(abscissa,oldpoint(1),oldpoint(2))=decr*pheromone(abscissa,oldpoint(1)
,oldpoint(2));
        %路径保存
        path(ii,abscissa*2-1:abscissa*2)=[oldpoint(1),oldpoint(2)];
        NowPoint=oldpoint;
    end
    path(ii,41:42)=[endy,endh];
end
```