

算法设计综合实验报告

无人机最优路径规划

姓名	承担的主要工作及贡献度
Lifei Zhang	网络搭建，算法实现 36%
Yaping Guo	算法实现，前端制作 33%
Haoran Tang	网络搭建，前端制作 31%

2023 年 11 月 25 日

目录

第一部分 题目介绍	3
1.1 题目背景	3
1.2 题目内容	3
1.2.1 基本题目	3
1.2.2 题目扩充与亮点	3
第二部分 仿真建模	4
2.1 设计框架	4
2.1.1 抽象数据结构	4
2.1.2 函数操作	5
2.2 问题建模	6
2.2.1 单一条件最短路径	6
2.2.2 多目标路径规划	6
2.2.3 k 最短路径路由	7
2.2.4 动态路由规划	7
2.3 本章总结	8
第三部分 相关算法	8
3.1 算法概述（使用的算法和算法设计方法）	8
3.2 算法描述（包含算法实现框架与框图）	9
3.2.1 深度优先搜索算法	9
3.2.2 Dijkstra 算法	10
3.2.3 K-最短路径算法	11
3.2.4 Multi-Objective Path Planning 算法	11
3.2.5 Q-Learning 算法	12
3.3 本章总结	13
第四部分 测试与分析	13
4.1 功能测试	13
4.1.1 前端测试	13
4.1.2 基础功能测试（包含用例与结果）	14
4.2 功能实现程度	18
4.3 性能分析（主要分析功能实现上的）	19
第五部分 收获与思考	19
5.1 问题解决	19
5.2 编码过程总结	19
5.3 设计与实现	20
5.3.1 现存不足	20
5.3.2 改进设想	20
5.4 收获与思考	20

第一部分 题目介绍

1.1 题目背景

在我国疫情期间，为了实现无接触收发物资，很多城市内部开始使用无人机来运送居民所需物资。请以某城市的地图和相关信息作为数据源，为该市负责运送物资的无人机提供各种信息查询和路径规划服务。鉴于小型无人机具备体型小，噪音小，安全系数高，姿态控制灵活等优势，城市内部一般使用小型无人机在城市内部进行物资运送，而这种无人机的飞行高度通常较低，对于一般建筑物不能直接穿行，也需要绕行，因此在规划无人机飞行路径的过程中也需遵循城市道路交通标准。

1.2 题目内容

1.2.1 基本题目

基于题目背景和设计要求，我们需完成以下基本内容：

①设计所在城市的城市平面图，所含地标不少于 10 个。以图中顶点表示市内各地标，存放地标名称、所需物资种类、物资数量等信息；以边表示各地标之间互通的路径，存放路径长度等相关信息。

②为无人机提供图中任意地标相关信息的查询服务。

③为无人机提供图中任意地标的的所有路径，即求出任意两个地标之间的所有路径。

④为无人机提供图中任意地标的的路径规划服务，即求出任意两个地标之间的一条最短的简单路径。

⑤为无人机提供图中多个地标的最优路径规划服务，即求出途径多个地标的最优路径。

1.2.2 题目扩充与亮点

在现有题目的基础上，我们基于已有的工作和目前设计上存在的不足，对题目进行了相应的扩充，具体内容如下：

①由于现实中我们可能需要知道从七点到终点尽可能多的待选路径，以便从中挑选最符合我们需要的，我们根据用时长短选择了 k 条待选路径集合，供使用者兼顾用时长短与其他需求。具体的，我们使用 k 最短路算法选出了多条相对来说较优的路径，并对它们进行排序，从而提供给用户，供用户按照需求进行选择。

②由于在现实情况下，飞机道路的通信情况是实时变化的，我们进行了动态路由以考虑道路信息，即在道路条件变化的情况下为无人机进行动态路径规划。

具体的，我们根据道路状况的变化更新链路权值，采用 Q_learning 算法通过学习训练根据全局信息进行最优路径选择。

我们对题目扩充的亮点在于考虑到了客观道路条件与用户的主观情况，为用户提供了多种选择，并充分考虑到了道路条件的动态性，更好地与现实情况进行了融合。

此外，在算法上，我们的亮点陈述如下：我们采用了深度优先搜索算法、迪杰斯特拉算法、k-最短路算法（yen 算法）、Q-Learning 算法等多种算法，并将其与项目进行了融合，以达到我们预想的目标。

第二部分 仿真建模

2.1 设计框架

2.1.1 抽象数据结构

我们将设计中的城市拓扑、城市节点、节点间路径、无人机表示为以下四类：

①城市拓扑类

我们将城市拓扑表示为以下结构：

$$G = \{N, E\} \quad (1)$$

其中 N 和 E 分别代表了地标和链路， $P_{i,j}$ 表示了一条完整的路由，具体的图由 Python 库 networkx 建立，具体见下表：

部分物理量及其含义	
$N = \{N_1, N_2, \dots, N_n\}$	城市节点集合
$E = \{E_{i,j} i, j \in \{1, 2, \dots, n\}\}$	链路集合
$P_{i,j} = \{V_i, E_{i,m}, V_m, E_{m,n}, \dots, E_{k,j}, V_j\}$	从 V_i 到 V_j 的路径

②城市节点类

我们将城市节点表示为以下集合：

$$N = \{N_1, N_2, N_3, \dots, N_n\} \quad (2)$$

代表了西安市各个地标，也是路由的相应节点，存放地标名称、所需物资种类、物资数量等信息。

该类中包含地标名称、所需物资种类、物资数量、是否可达等属性。

③节点间路径类

我们将城市各地标间可供飞机飞行的链路表示为：

$$E = \{E1, E2, E3, \dots, En\} \quad (3)$$

代表了两个节点间的链路，存放路径长度、节点间链路质量等信息。

该类中包含路径长度、通信状况、路径上飞机数、可容纳飞机数等属性。

④无人机类

我们将无人机表示为以下集合：

$$F = \{F1, F2, F3, \dots, Fn\} \quad (4)$$

表示了无人机的起始节点、目标节点(可能多个)、承载物资的情况与传输过程中所经过路径的集合。

该类中包含了起始节点、目标节点、路径集合记录等属性。

2.1.2 函数操作

在以上四个类的构建中，我们还为其添加了若干类方法，实现各种函数操作，具体如下：

①城市拓扑类

在该类中，主要方法有：

- 初始化一个无向图。
- 添加地标节点到图中，包括节点标识、名称、需求和可访问性信息。
- 添加连接信息到图中，包括连接的两个节点和连接的延迟。
- 绘制图形，使用 Spring 布局展示节点关系，并在图上显示边的权重信息。
- 根据当前节点和下一个节点获取连接的延迟，通过检查一个预定义的拓扑图来实现。
- 找到从起始节点到目标节点的最短路径。如果给定参数 n 小于等于 0，返回 0；如果 n 等于 1，使用 Dijkstra 算法找到最短路径；如果 n 大于 1，使用 Yen's K-Shortest Path Algorithm 找到前 n 条最短路径。

②城市节点类

在该类中，主要方法有：

- 构造方法，用于初始化节点对象，接受节点的序号、名称、需求和容量等信息。
- get_requirement(self)：返回节点的需求。
- get_sign(self)：返回节点的序号。
- get_name(self)：返回节点的名称。
- get_datasize(self)：返回节点的容量。

③节点间路径类

在该类中，主要方法有：

- 初始化链路对象，接受链路两端连接的节点序号，链路长度，链路能承载的最大飞机数，起飞时延和标志位。
- 占用和释放链路：如果链路上的占用量加上新的数据大小不超过最大承载量，则占用链路并返回 True；否则返回 False。如果链路上的占用量大于等于传入的数据大小，则释放链路并返回 True；否则返回 False。
- 更新链路信息：根据链路占用量更新链路信息，例如延迟、最大飞机数量等，设置标志位以表示链路通信状况。
- 更新连接信息，例如延迟或标志位。

- e) 查询链路信息:查询链路信息, 例如延迟, 链路上的飞机占用量等, 通过打印属性值的方式展示。
- f) 获取链路长度: 返回链路的长度。

④无人机类

在该类中, 主要方法有:

- a) 初始化:接受起始节点标号、终止节点标号、目标路径数量、飞机大小、起飞时延等信息。
- b) 飞行模拟:根据城市地图和目标路径数量计算最短路径, 模拟飞机的飞行过程, 并记录事件到日志。
- c) 地标信息查询:查询城市地图中特定地标的信息, 返回相应地标对象。
- d) 地标信息更新:更新城市地图中特定地标的信息, 根据地标名称和新信息进行更新。
- e) 事件记录:记录事件到日志, 将事件信息添加到事件日志中。
- f) 全局飞行模拟:初始化飞行队列, 模拟所有无人机的飞行过程, 记录飞行事件到日志, 并返回飞行过程中记录的事件日志。

2.2 问题建模

2.2.1 单一条件最短路径

单一条件最短路径, 即为无人机提供图中任意地标的路径规划服务, 并指出最优路径。

我们将从起点到终点的所有路径表示为 $P_{i,j}$, 该过程所需花费的时间为 $T(P_{i,j})$, 则该目标可以表述为以下内容:

$$\min\{T(P_{i,j}=\{V_i, E_{i,m}, V_m, E_{m,n}, \dots, E_{k,j}, V_j\})\} \quad (5)$$

此时, 可以得到在单一条件下的最短路径, 即给定起点和终点之间的最短路径。

2.2.2 多目标路径规划

多目标路径规划, 即为无人机提供图中多个地标的最优路径规划服务, 即求出途径多个地标的最优路径。

我们将从起点开始, 经过多个中间目标节点到达终点的所有路径表示为 $P_{i,j}$, 在这个过程中, 有 V_m, V_n 等中间目标节点, 将全过程所需花费的时间为 $T(P_{i,j})$, 每个中间节点的等待时间为 $T(P_{i,k})$, 理想情况下, 该目标可以表述如下:

$$\min\{T(P_{i,j}=\{V_i, E_{i,m}, V_m, E_{m,n}, \dots, E_{k,j}, V_j\})\} \quad (6)$$

$$\min T(P_{i,m}), T(P_{i,n}), \dots, T(P_{i,j}) \quad (7)$$

然而, 现实情况下, 同时做到(6)(7)两式是很困难的, 所以, 针对这种情况, 无人机在选择路径时, 我们为其设计了以下两种策略供其选择:

- a) 采取全局最优化路由: 即优先满足(6)式条件, 其次考虑(7)中的局部最优解;

b) 采取综合评价机制:即综合考虑整体时间与中间节点的等待时间(见(8)式)。

$$\min (\alpha\{T(P_{i,m})\}+\beta\{T(P_{i,n})\}+\cdots+\tau\{T(P_{i,j})\}) \quad (8)$$

S.t.

$$\alpha+\beta+\cdots+\tau=1$$

$$0<\alpha,\beta,\cdots,\tau<1$$

以上部分物理量及其含义如下:

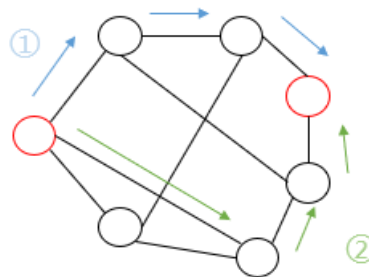
部分物理量及其含义	
$E = \{E_{i,j}\}$	链路集合
$P_{i,j} = \{V_i, E_{i,m}, V_m, E_{m,n}, \dots, E_{k,j}, V_j\}$	从 V_i 到 V_j 的路径
$T(P_{i,k})$	每个中间节点的等待时间

2.2.3 k 最短路路由

K-最短路路由即为无人机的路径查询结果提供更详尽的导向信息,求出到达一个目标节点的多条(所有)最短路径

由于现实中我们可能需要直到尽可能多的待选路径,以便从中挑选最符合我们需要的(不一定是用时最短的),我们根据用时长短选择了 k 条待选路径集合,供使用者兼顾用时长短与其他需求

这个时候,所需满足的目标由使用者决定。该问题可以表述为下图:

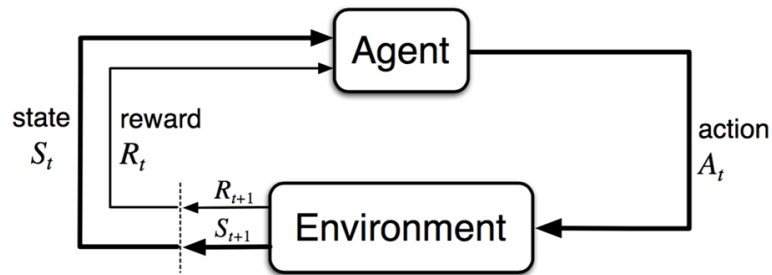


2.2.4 动态路由规划

动态路由规划考虑道路信息,即在道路条件变化的情况下为无人机进行动态路径规划。

具体的,我们根据道路状况的变化更新链路权值,采用 Q_learning 算法通过学习训练根据全局信息进行最优路径选择。

该问题可以表述为以下马尔可夫过程:



我们的需求是在不断与环境的交互中取得全局最优解，即最优路径。
关于该问题的详细表述与建模过程将会在 3.2.5 节中详细说明。

2.3 本章总结

在本节中，我们描述了程序的设计框架，如代码会涉及到的数据结构和函数操作，并针对问题需求进行了系统建模。

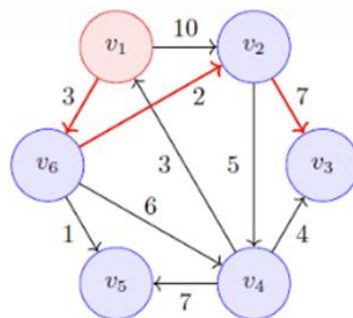
第三部分 相关算法

3.1 算法概述（使用的算法和算法设计方法）

①Dijkstra 算法及深度优先搜索算法

预期实现功能：单目标路径路由，所有路径规划。

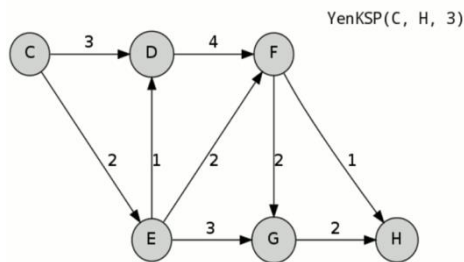
算法设计方法：利用 Dijkstra 算法实现单目标路径路由并用深度优先搜索算法实现所有单目标路由的所有路径规划。



②K-最短路算法

预期实现功能：多途径路径路由，即选定起点终点给出 K 条优选路径。

算法设计方法：利用 yen 算法实现多途径路径路由。



③Multi-Objective Path Plannin 算法

预期实现功能：多目标路径路由，即选定起点终点以及中间节点，给出最优路径。

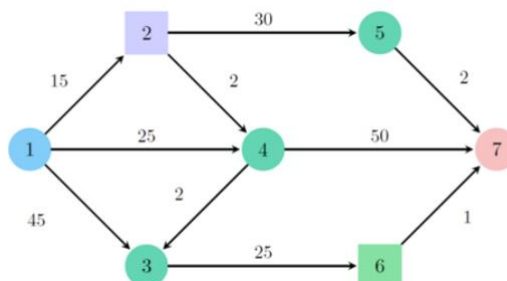
算法设计方法：利用 K-最短路算法计算出优选路径后筛选出经过中间节点的路径。

```
# 从k条路径中筛选出经过中间结点的路径
for path in shortest_paths:
    if middle in path:
        filtered_paths.append(path)
    cnt = cnt + 1
```

④Q-Learning 算法

预期实现功能：动态路由，即根据城市拓扑图实时状态进行路径规划，实时更新最优路径。

算法设计方法：利用 Q-Learning 算法，在局部网络之间实现寻路，将路径权值在每个 episode 前动态改变并更新 Q-table 实现动态路由。



3.2 算法描述（包含算法实现框架与框图）

3.2.1 深度优先搜索算法

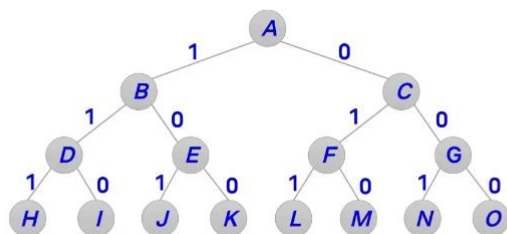
深度优先搜索（Depth First Search, DFS）是一种常用的图遍历算法，用于搜索或遍历图或树的节点。它通过从起始节点开始，尽可能深入地探索图的分支，直到达到最深的节点，然后再回溯到前一层继续探索其他分支。

实现框架：

将当前节点 current 添加到当前路径 path 中。

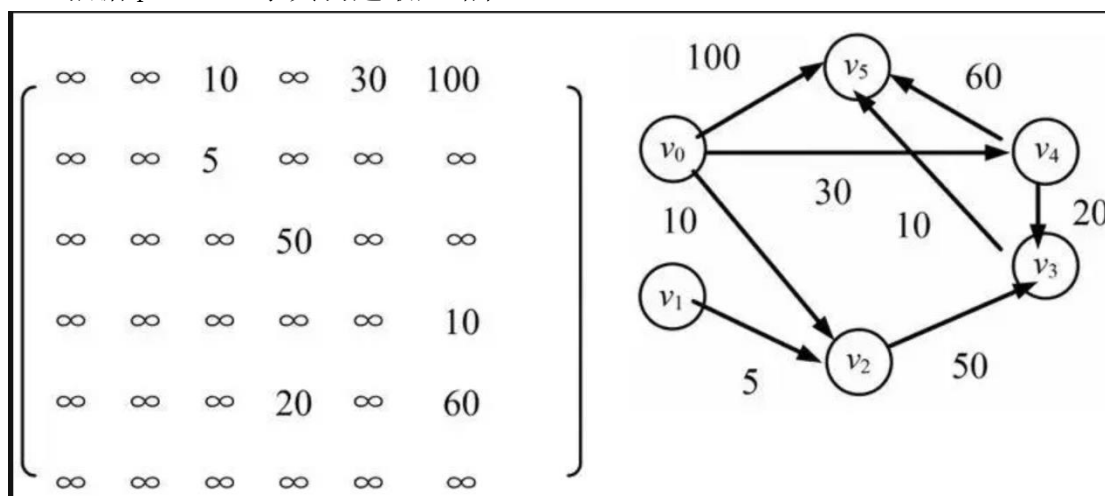
如果当前节点 current 等于目标节点 target，说明已经找到一条路径，将该路径保存到结果中。

遍历当前节点 `current` 的邻居节点 `neighbors`：
 如果邻居节点 `neighbors` 不在当前路径 `path` 中，即没有访问过：
 调用递归方式的 `dfs` 函数，传入邻居节点 `neighbors` 作为当前节点 `current`，目标节点 `target` 和更新后的当前路径 `path`。
 在函数外部调用 `dfs` 函数，传入起始节点作为当前节点 `current`，目标节点作为目标节点 `target`，并初始化当前路径 `path` 为空列表。
 返回结果中保存的全部路径。



3.2.2 Dijkstra 算法

迪杰斯特拉算法是一种用于计算加权图中的最短路径的算法。
 实现框架：
 创建一个距离字典 `distances`，用于保存起点到各个顶点的最短距离。
 创建一个父节点字典 `parents`，用于保存每个顶点的父节点，即最短路径上的前一个顶点。
 创建一个已访问集合 `visited`，用于保存已经找到最短路径的顶点。
 初始化起点的距离为 0，其他顶点距离为无穷大。
 找到 `distances` 中距离最小的顶点，并将其加入 `visited` 集合。
 遍历该顶点的邻居，更新其距离和父节点。
 重复步骤 5 和 6，直到所有顶点都被访问。
 根据 `parents` 字典构建最短路径。



3.2.3 K-最短路算法

Yen 算法是一种用于计算加权图中多个最短路径的算法实现框架：

输入起点和终点，以及图的邻接矩阵或邻接表表示。

初始化一个空的路径列表 `paths`，用于保存找到的最短路径。

初始化一个空的候选路径列表 `candidates`，用于保存待处理的路径。

使用 Dijkstra 算法计算起点到终点的最短路径，将其添加到 `paths` 列表中。

如果 `paths` 列表为空，则停止算法。

遍历 `paths` 列表中的每个路径 `path`：

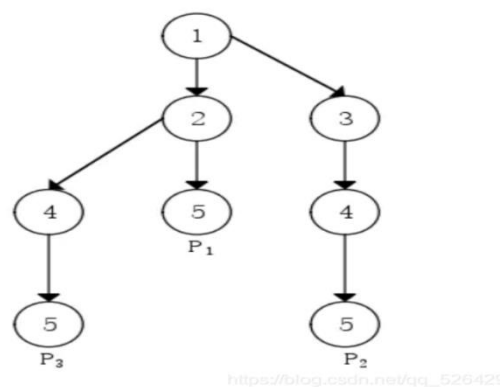
对于每个节点 `i`（除了起点和终点）：

从 `path` 中删除节点 `i` 及其后面的节点，得到一个新的路径 `partialPath`。

如果 `partialPath` 不在 `candidates` 列表中，则添加到 `candidates` 列表中。

从 `candidates` 列表中选择最短路径作为新的路径 `path`，将其添加到 `paths` 列表中。

重复步骤 6 和 7，直到找到所有的最短路径或者 `candidates` 列表为空。



3.2.4 Multi-Objective Path Planning 算法

多目标 A* 算法 (MA*) 是一种用于解决多目标路径规划问题的常见算法。

在 MA* 算法中，每个路径都有一个与之关联的代价向量，其中包含多个优化目标的值。初始路径是从起点到目标点的单一路径。在搜索过程中，通过扩展路径并计算新增路径的代价向量，将优化目标的值与已有路径进行比较。如果新增路径在某个优化目标上优于已有路径，则将其加入搜索树中。如果新增路径在所有优化目标上都优于已有路径，则将已有路径替换为新增路径。这样，搜索树中就会保留多个具有不同优化目标值的路径。MA* 算法继续在搜索树中扩展路径，直到找到一个或多个满足所有优化目标的路径。这些路径构成了最优解的近似集合。

但是，MA* 算法在计算路径代价时，一般将节点间的直线距离作为参数进行衡量，然而在现实情况中，受建筑、三维空间、城市设计等因素影响，两个地标之间的路径距离往往并非直线距离、有时甚至无法直接到达需要借助其他地标进行中转。因此，在设计实现多目标路径规划功能时，我们并未采用 MA* 算法，

而是考虑城市平面图地标路径的非线性，选择通过 k -最短路算法来间接实现多目标路径的路径规划。

因此，我们在上述 k -最短路算法实现的基础上，采用了建模设计时的综合评价机制，从而求出途径多个目的地标的最优路径。具体过程如下：

①通过设置 k 的数量，挑选出尽可能多的经过起始地标和终点地标的路径，放入待选路径集合。

②从待选路径中挑出所有经过所要求的中间结点的路径，放入目标路径集合。

③根据综合评价公式，计算目标路径中每个结点的等待时间（除起始节点外），并据此计算综合时间。

$$\min (\alpha\{T(P_{i,m})\}+\beta\{T(P_{i,n})\}+\cdots+\tau\{T(P_{i,j})\}) \quad (8)$$

S.t.

$$\alpha+\beta+\cdots+\tau=1$$

$$0<\alpha,\beta,\dots,\tau<1$$

④从目标路径集合中挑选出综合时间最小的路径作为最优路径。

3.2.5 Q-Learning 算法

Q-learning 是一种机器学习方法，它使模型能够通过采取正确的操作来迭代学习和改进。Q-learning 模型通过反复试验来学习任务的最佳行为。Q-learning 过程涉及通过学习最佳动作价值函数或 Q-function 来建模最佳行为。该函数表示状态 s 中动作 a 的最佳长期价值，并随后在每个后续状态中遵循最佳行为。

在此，我们选择如下 Bellman's equation 方式来评价预期奖励：

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * \max_{a'}(Q(s',a')) - Q(s,a)) \quad (9)$$

基于此，Q-learning 算法过程是一种交互式方法，代理通过探索环境并根据收到的奖励更新 Q-table 来学习。其中算法过程中涉及的步骤包括：

- Q-table 初始化：创建 Q-table，作为跟踪每个状态下的每个动作和相关进度的地方
- Observation：代理需要观察环境的当前状态
- Action：智能体选择在环境中行动。动作完成后，模型会观察该动作是否对环境有益。
- Update：采取行动后，用结果更新 Q-table
- Repeat：重复步骤 b-d，直到模型达到预期目标的终止状态。

相比于传统算法，无模型方法是 Q-learning 的基础，也是某些用途的最大潜在优势之一，Q-learning 代理不需要关于环境的先验知识，而是可以在训练时了解环境，无模型方法特别适用于环境的底层动态难以建模或完全未知的场景，将其应用于最佳路由搜索，有着良好的应用前景。

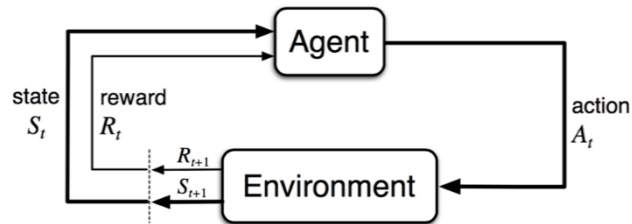
在我们对 Q-learning 算法的测试中，我们使用了如下参数：

- Env：根据西安市部分小拓扑城建的图
- Action：每个节点的下一跳，即可以选择的下一个动作，具体为{1: [2, 3, 4], 2: [4, 5], 3: [6], 4: [3, 7], 5: [7], 6: [7], 7: []}
- State：当前所处节点，即 Node (n)
- Reword：我们将奖励定义为 $\text{reward} = \text{max_length} - \text{network}[\text{arc}]$ ，即网络中所有边的最大长度减去当前行动所对应的边的长度

我们的目的是让代理寻找最短路径（奖励值与路径长度成反比），基于此，给出相关参数的解释和马尔可夫过程：

- $Q(s,a)$ 是当前状态 s 下执行动作 a 的 Q 值。
- α 是学习率（learning rate），控制新信息对 Q 值的影响程度。
- R 是在执行动作 a 后获得的奖励。
- γ 是折扣因子（discount factor），表示未来奖励的重要性。
- s' 是执行动作 a 后的下一个状态。
- Q -table 的更新公式如下：

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot (R + \gamma \cdot \max_a Q(s',a)) \quad (10)$$



训练过程见上图，从起始节点开始，选择具有最高 Q 值的动作，直到达到目标节点，根据 Q -table 得到一条最佳路径。

3.3 本章总结

在本节中，我们描述了用以实现程序功能的五种算法，并展示了它们的实现思路及框架。

第四部分 测试与分析

4.1 功能测试

4.1.1 前端测试

为方便无人机使用和满足题目要求，我们根据程序功能设计了如下的人机交互页面。



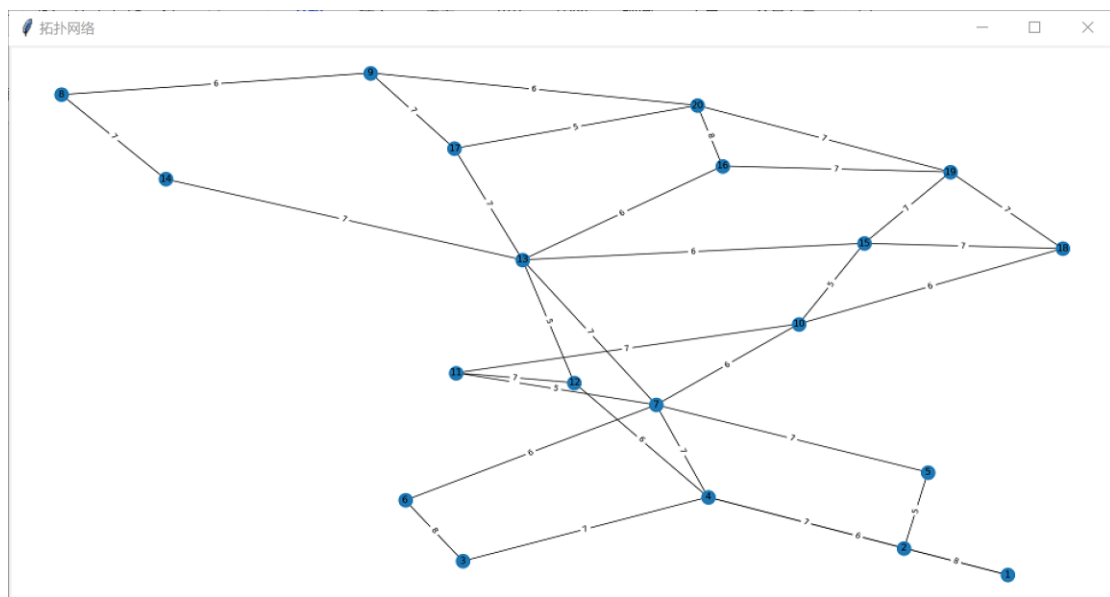
经过测试，前端主页面的五个按钮均可以正常使用并打开相应功能菜单，且相应功能均成功实现输入输出，因此我们可以认为已初步建立起前端框架，后续可进行功能测试。

4.1.2 基础功能测试（包含用例与结果）

根据题目基本要求及扩充内容，程序代码主要实现了以下四大功能，分别是查看拓扑网络、路标查询服务、显示全部路径以及路径规划服务。下面将一一进行这些功能的具体测试。

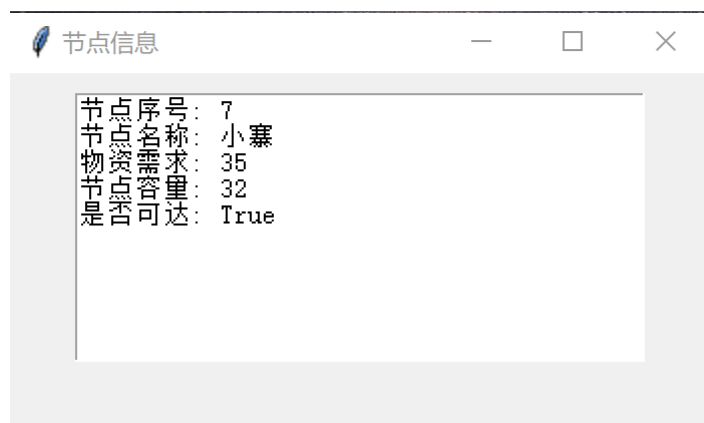
①查看拓扑网络

点击“拓扑网络”按钮，即可显示西安城市平面图的拓扑网络。



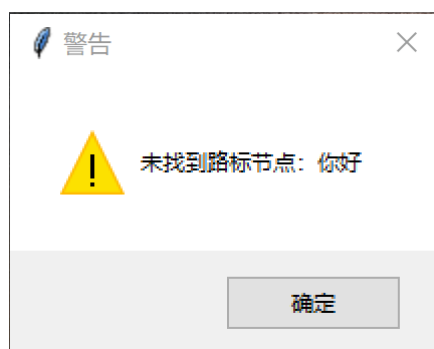
②路标查询服务

点击“路标查询”按钮，输入想要查询的路标名称，即可查看路标的相关信息。



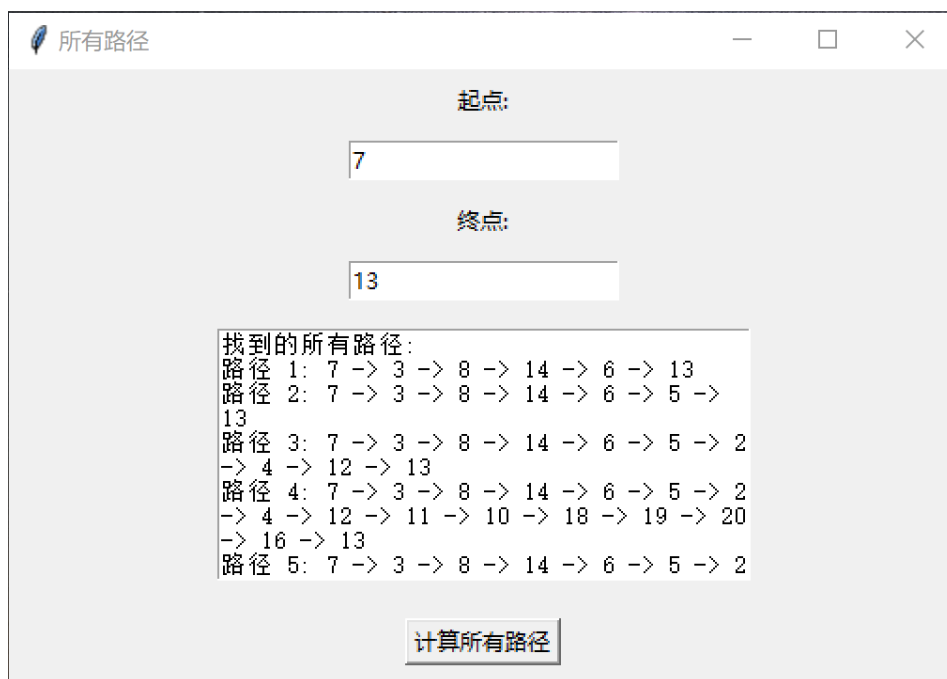
通过将显示的路标相关信息与 `router` 类中小寨的存储信息进行对比，我们可以发现信息一致，即路标查询功能已正确实现。

功能完善，如果输入了非西安城市平面图节点的路标名称时，将会提示警告。



③显示全部路径

点击“全部路径”按钮，输入起点序号、终点序号，将会显示这两个地标间的全部路径。



④路径规划服务

路径规划服务是无人机路由系统的重点实现功能，包含了以下四种路径规划方法。



1) 单目标路由

采用 dijkstra 算法，输入起点序号、终点序号，查看最短路径。

单目标路由

起点:

13

终点:

7

最短路径: 13 -> 4 -> 1 -> 2 -> 3 -> 7

计算最短路径

节点 13 和 7 之间其实是有直达路径的，但因为这两个都属于热门地标，其路径上会经过大量人群，更加容易导致无人机丢失物资，因此将路径权值设为 10000，降低其选择性。根据显示结果，我们发现算法在选择路径时避开了权值很大的直达路径，而是通过计算得出了权值和最小的最优路径，即功能正确实现。

2) 多目标路由

采用 k-最短路算法+综合评价机制，输入目标节点，得到待选路径。

多目标路由

起点:

13

中间结点:

20

终点:

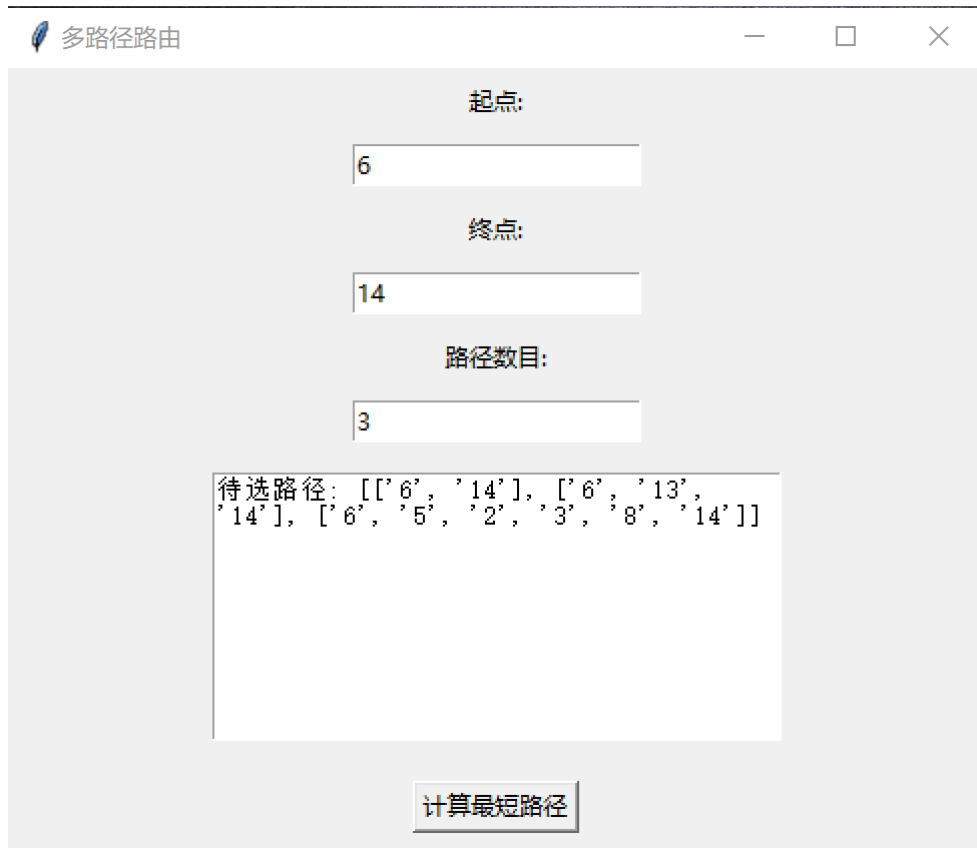
16

待选路径: ["['13','17','20','16']840"]

计算最短路径

3) 多路径路由

采用 k-最短路算法，输入 k 及起点、终点，计算路径。



多路径路由

起点:

6

终点:

14

路径数目:

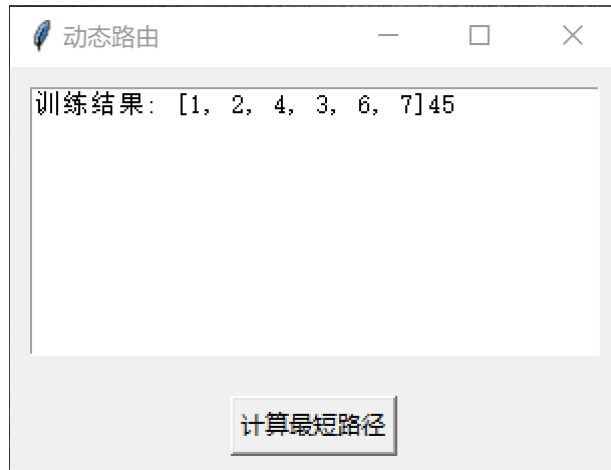
3

待选路径: [['6', '14'], ['6', '13', '14'], ['6', '5', '2', '3', '8', '14']]

计算最短路径

4) 动态路由规划

采用 Q-learning 算法，从西安城市平面图中划分局部拓扑图进行训练。



动态路由

训练结果: [1, 2, 4, 3, 6, 7]45

计算最短路径

4.2 功能实现程度

根据题目基本要求及扩充内容，程序代码主要实现了以下四大功能，分别是查看拓扑网络、路标查询服务、显示全部路径以及路径规划服务。

其中，查看拓扑网络、路标查询服务和显示全部路径功能均已成功实现。对

于主要功能路径规划服务而言，其下包括的单目标路由、多目标路由、多路径路由也均已成功实现，而动态路由规划部分，因全局模型收敛较差，最终选择了从城市平面图中划分部分拓扑网络进行 Q-learning 训练的策略进行路径选择，未能完成之前建模设计时的全部设想。

4.3 性能分析（主要分析功能实现上的）

查看拓扑网络、路标查询服务、显示全部路径以及路径规划服务中的单目标路由、多目标路由、多路径路由均按构想完成实现，满足性能要求。针对动态路由规划，因未按最初构想完成代码编写，故程序性能部分有所局限，只能在动态规划划分部分的拓扑网络根据现实情况更改链路权值进行 Q-learning 训练，与现实情况会有一定出入，还需继续优化。

第五部分 收获与思考

5.1 问题解决

问题建模过程比较顺利，搭好基本框架后进行编码过程中存在以下问题：

1. 手动输入的城市拓扑图与设置的无人机类数据类型冲突，模拟无人机飞行过程时无法正确调用城市拓扑图（修改无人机类属性之后解决）
2. 调用迪杰斯特拉算法时报错（修改函数格式解决）
3. 多途径路由算法中调用 K-最短路算法，所得结果不正确，选出的路径非最优路径（新增判别条件解决）
4. Q-learning 算法模型无法收敛于整个城市拓扑图（选用小型拓扑图进行测试，测试成功）
5. 前端基础薄弱，设计 UI 界面较为困难（学习后完成）

5.2 编码过程总结

本项目采用 python 语言编码，python 代码简洁但不易维护。各个算法编写过程都较为顺利，但算法之间的相互调用以及对城市拓扑图的调用较为苦难。同时最后设计前端 UI 界面时 GUI 的传参也给我们带来了不小的麻烦。

5.3 设计与实现

5.3.1 现存不足

在目前的项目中，存在着以下不足：

首先，对于西安市地图拓扑的建立。在建立拓扑图的过程中，我们格局西安市地铁线路图中的若干重要节点手动添加进拓扑图，并根据它们之间的实际地理距离设置权值。而这样的方法人为主观性和随意性过强，可能不是一种科学的城市拓扑建立方法。

此外，我们对 Q-learning 算法的实现部分，目前只是在小型拓扑图中可以成功实现，而一旦放入整个西安市的拓扑中，在绝大多数情况下，模型无法收敛，无法给出最终的结果。所以，对于这部分我们目前仍旧有着很大的研究与改进空间。

5.3.2 改进设想

针对 5.3.1 中提出的两点不足，我们的改进设想如下：

首先，对西安市拓扑图建立更加数学化表述的模型，并研究相关城市公交网络建模及其可靠性的相关论文与已有研究，在此基础上给出更加科学的西安市拓扑建立方案。

对于 Q-learning 算法部分，我们首先需要调整部分参数，并尝试能否找到效果最佳的参数组合，其次，更改 Q-table 的初始化策略，可以在建立 Q-table 时运用一次最短路算法，并以此建立 Q-table，便于模型收敛。最后，我们需要整合该算法到实际应用中去。

5.4 收获与思考

本次对问题的建模设计是基于传统无人机路由进行的创新性改良，考虑到城市交通条件的各种限制以及其动态变化，我们引入了深度学习算法 Q-learning，希望能通过深度学习模型实现无人机的动态路由规划，将城市实时拓扑信息保存至 Q-table 中，让无人机路径规划更具有合理性，便于提高整体的路由效率。

对于题目中要求单目标路径路由，多目标路径路由以及多途径路由我们均采用了相对传统的算法进行实现，虽然在实际编码过程中我们遇到了很多小问题，但我们都通过逐步的调整成功解决了这些问题。回顾算法编码过程中，我们认为在多目标路径路由中依然采用 yen 算法的这个举措还有改进空间，因为这种算法之间相互调用的策略会降低代码运行效率，并且出错时也不易维护。

本项目完成了所有题目包含的基本要求，但仍需改进。对于深度学习算法 Q-learning，我们未能选取合适的参数组合对其进行初始化，导致其无法在真正的大型城市拓扑图中应用，对该算法的研究还不够透彻，还有很大的提升空间。

最后，通过这次项目，我们获取了宝贵的工程经验。我们具备了一定将自己学习的算法知识应用于实际工程项目的的能力，而对代码的编译调试过程中我们也提升了自己的调试维护能力，受益匪浅。