西南财经大学

Southwestern University of Finance and Economics

论文题目：基于改进弗洛伊德算法的航空路线规划问题

学生信息：

姓名	学号	学院	专业	联系电话	邮箱
兰文婕	4				
李之熹	4				
印旭东	4				

基于改进弗洛伊德算法的航空路线规划问题

摘　要

本文基于航空公司运营与管理的背景，结合具体的题目要求，建立了图论模型，以 Floyd 算法为核心算法，在最少转机次数的条件下计算航线问题；在无线路重复、有线路重复的等情况下计算日航班相关问题；在划分国内航班、国际航班的情况下计算周航班的相关问题。

针对问题一，我们首先给机场编号，建立**无权重有向图模型**。求出仅转机一次的**城市对有 393 对**。紧接着运用 Floyd 算法，解得乘机旅行路线中**最大转机次数为 5**，并利用 Dijkstra 算法求得路径，分别为：**（1）义乌-汕头-杭州-北京-长沙-青岛-沈阳；（2）义乌-汕头-杭州-北京-长沙-太原-大连。**这两条路径反向亦成立。

针对问题二，我们考量了题目中的候机时间要求，在原本的 Floyd 算法中引入起飞时间、降落时间和飞行时间三个矩阵，建立**时间追踪模型**。并假设在不满足候机最短时长（1 小时）的时候，旅客会等待次日航班（>24 小时）。最终，我们得到结果：（1）共有 **91 条线路**在 6 小时之内到达；（2）转机次数最多的线路中**转机为 4 次**；转机次数最多的**线路有 20 条**。（3）旅行时间最长的城市对为**三亚到乌鲁木齐**，线路为：**三亚→南通→北京→西宁→乌鲁木齐**，总时长为 **4540 分钟（75 小时 40 分钟）**。

针对问题三，部分城市间的路线数量不唯一。经分析可知，各城市各路线组合数量有限，因此我们采取**穷举模型**的思路，建立集合对其一一排列组合，并基于问题二中的改进 Floyd 算法求解。最终，我们得到答案：（1）共有 **95 条线路**在 6 小时之内到达；（2）转机次数最多的线路中**转机为 4 次**；转机次数最多的**线路有 20 条**。（3）旅行时间最长的城市对为**义乌到乌鲁木齐**，线路为：**义乌→揭阳→杭州→北京→西宁→乌鲁木齐**，总时长为 **4500 分钟（75 小时）**。

针对问题四，我们首先对国内国际机场进行顺序编号以作为区分，再以一周（七天）为周期，将航班的起飞、降落时间进行时间戳形式的转化与记录，紧接着对其按照问题三的方法进行排列组合。为了更合理地处理庞大的数据量，我们引入概率统计中的**抽样模型**对总体数据进行合理抽样，根据多次抽样实验的结果求得问题的解：（1）共有 **192 条线路**在 6 小时内到达；（2）须转机一次的线路共有 **639 条**；（3）耗时最长的线路为 **普吉岛→西安→北京→杭州→揭阳→义乌** 或者 **普吉岛→西安→长沙→烟台→杭州→揭阳→义乌。**总时长为 **6540 分钟（109 小时）**。

全文在遵循算法核心思想的条件下，创新性地改进了 Floyd 算法，解决了航线安排，日航班管理和周航班管理等多个航空管理问题。文章的模型对航空公司的运营与管理具有一定的借鉴意义。

关键词：航空运营管理　图论　Floyd 算法　Dijkstra 算法

一、 问题重述

1.1 背景资料与条件

航空出行是当代的主要交通方式之一，而航空公司正确的运营与管理为安全舒适的航空出行提供了基础与保障。由此，航空管理是航空交通系统中极其重要的一环。一般的航空管理问题包括航线分析、日航班规划、周航班规划等。

1.2 需要解决的问题

（1） 求解仅转机一次就到达的城市对数量。
（2） 计算并描述乘机旅行线路中转机次数最多的城市对。
（3） 在无路线重复的条件下，求解并描述 6 小时内可到达的线路，转机次数最多的线路以及耗时最长的线路。
（4） 在有路线重复的条件下，求解并描述 6 小时内可到达的线路，转机次数最多的线路以及耗时最长的线路。
（5） 在区分国际航班与国内航班的情况下，求解并描述 6 小时内可到达的线路，须转机一次的线路以及耗时最长的线路。

二、 问题分析

2.1 问题一的分析

针对问题一，我们建立无权重有向图模型，并利用弗洛伊德（Floyd）算法和迪杰斯特拉（Dijkstra）算法进行求解。

第（1）问要求计算仅转机一次就能到达的城市对数量。

首先，我们先将每个机场简化为单个节点，建立无权重的有向图模型；其次，我们进行问题转化，把“仅转机一次就能到达的城市对”问题转化为寻找“间隔两个单位权重的城市对”，并利用 Matlab 进行编程求解。

第（2）问要求乘机旅行路线中转机次数最多的城市对。

我们先利用 Floyd 算法求出城市对间的最短路径，构成最短路径集合 D；紧接着，在集合 D 中找出路径最大，即转机次数最多的城市对，并描述其路径。

2.2 问题二的分析

问题二给定无线路重复的条件，要求我们求最短旅行时间中的最长时间。我们分步骤求解，建立赋权有向图，并利用改进的 Floyd 算法进行求解。

我们对于 Floyd 算法的改进主要基于对实际情况下真实旅行时间的考量。真实旅行时间既包括航空飞行中的耗时，也包括在机场候机的耗时。在考虑到限定条件的情况下（候机时间≥ 1），我们建立模型追踪转机的每一个节点以及旅行总时间，改进 Floyd 算法进行求解。

2.3 问题三的分析

问题三与问题二类似，但无线路重复的条件改变为有线路的重复，即城市对之间存在不止一条线路。针对于此，我们采用穷举法，用集合收纳所有可能线路，并一一排列组合，再利用第二问改进的 Floyd 算法进行求解。

2.4 问题四的分析

问题四涉及国际、国内两类航班以及拓展为以周为期限的航班管理。经过建模与分析，其数据量达到庞大的指数级数据量。因此，我们引入概率统计模型，借助样本量模型确定我们抽样的样本容量，进而通过对样本的估计求得近似解。

三、 模型假设

1. 一个机场代码代表一个城市
2. 假设图形为有向图
3. 航空旅行总时长超过一天是合理的，即旅客会等待次日航班（>24 小时）

四、 符号说明

符号	说明	单位
Ci	城市代码（$i=1\sim n$）	-
A	邻接矩阵	-
D	城市对之间最短路径的集合	-
T	总旅行时长	分钟
TP_{ij}	C_i到C_j所有路线的集合	-
S_{ij}	C_i到C_j各路线耗时的集合	-

五、 模型的建立与求解

5.1 问题一模型的建立与求解

此题中，我们建立了图论（无权重有向图）模型，并运用 Floyd 算法求解问题。

5.1.1 模型的建立

首先，我们将每个机场编号为单个节点 $Ci(i=1\sim 49)$，如下表所示：

表 1 机场编号

机场代码/节点编号							
AEB	$C1$	HFE	$C13$	LYG	$C25$	SZX	$C37$
AOJ	$C2$	HGH	$C14$	MIG	$C26$	TAO	$C38$
BHY	$C3$	HKT	$C15$	NGB	$C27$	TYN	$C39$
BKK	$C4$	HND	$C16$	NKG	$C28$	URC	$C40$
CAN	$C5$	HRB	$C17$	NNG	$C29$	WXN	$C41$
CKG	$C6$	HUZ	$C18$	NTG	$C30$	XIY	$C42$
CSX	$C7$	JJN	$C19$	NZH	$C31$	XMN	$C43$
CTU	$C8$	KHN	$C20$	OSA	$C32$	XNN	$C44$
DLC	$C9$	KLO	$C21$	PEK	$C33$	YIW	$C45$
DYG	$C10$	KMG	$C22$	SHE	$C34$	YNT	$C46$
FOC	$C11$	KWL	$C23$	SWA	$C35$	YNZ	$C47$
HAK	$C12$	LHW	$C24$	SYX	$C36$	ZHA	$C48$
						ZUH	$C49$

由此，我们得到图的点集 V 以及由各节点组合成的有序数对的边集 E：

$$V = \{Ci | i = 1 - 49\}$$
$$E = \{(Ci,Cj) | i,j = 1 - 49\} \tag{1}$$

综上，我们的所建立机场/城市图 G 可记作：$G =< V,E >$。

紧接着，我们建立邻接矩阵 A：

$$A = (a_{ij})_n \tag{2}$$

其中，$n = 49$ 且：

$$a_{ij} = \begin{cases} 0 & (Ci,Cj) \in E \\ 1 & (Ci,Cj) \notin E \end{cases} \tag{3}$$

元素a_{ij}表示从机场C_i到机场C_j 的连接，若C_i有到达C_j的航班，则取值 1；若没有航班连接两城市(i,j)，则取值 0。

根据题目所提供的航线数据，可填充得到邻接矩阵：

$$A = \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}_{49 \times 49} \tag{4}$$

航线数据经 Matlab 处理得到图 G 的示意图如下：

（注：图中城市代码省去字母 C，只保留序号 i）

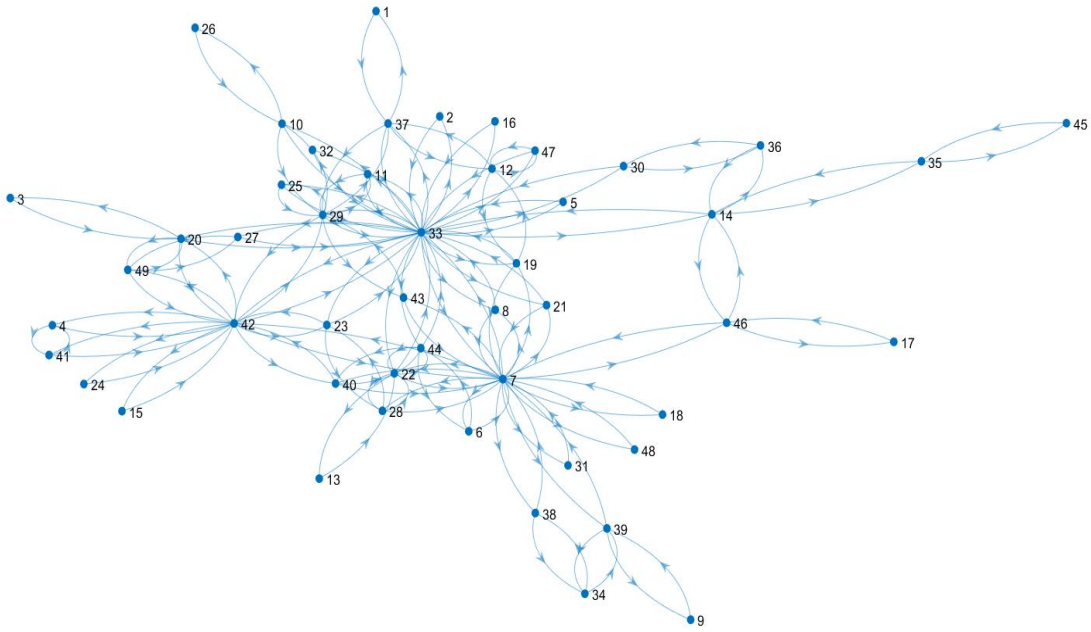


图 1 机场无权重有向图

5. 1. 2 模型的求解

我们所建立的第一题的图论模型为无权重的有向图，我们不妨把单位权重记为 1。

经分析，**第（1）问**的“仅转机一次就能到达的城市对”可以转化为寻找“间隔两个单位权重的城市对”，即寻找距离为 2 的城市对。

经 Matlab 编程求解（代码见附录 1），我们求得仅转机一次就到达的城市对数量为 393 对。

表 2 转机一次到达的城市对

城市对	393

注：在计算此题时，我们假设一个机场代表一个城市，因此我们是通过找机场对来确定的城市对。

第（2）问，我们先采用弗洛伊德算法（Floyd）求解最短路径以确定乘机旅行路线。由于本题建立的模型为无权重有向图，故我们的路径长度用单位权重的累计进行表示。如：一个单位权重路径长度为 1；两个单位权重路径长度为 2。

Floyd 算法步骤：（Matlab 代码见附录 2）

第 1 步：将各机场编号为 $C1\sim C49$，并构造邻接矩阵 A，其中，元素a_{ij}表示从机场C_i到机场C_j 经过节点序号不大于 k 的最短路径长度。

第 2 步：随机插入城市节点 k（k 也为迭代次数），分别计算 $a(i,j)$以及 $a(i,k)+a(k,j)$，应用下列迭代公式进行迭代：

$$a_k(i,j)=min(a_{k-1}(i,j)\ ,a_{k-1}(i,k)+a_{k-1}(k,j)) \tag{1}$$

第 3 步：每确定一个元素a_{ij}时，我们记录下它所代表的路径及其路径长度。算法终止时，矩阵 A 中的元素(i,j)就表示从机场 Ci 到 Cj 的最短距离。

Floyd 算法可以用以下流程图进行更直观的演示。

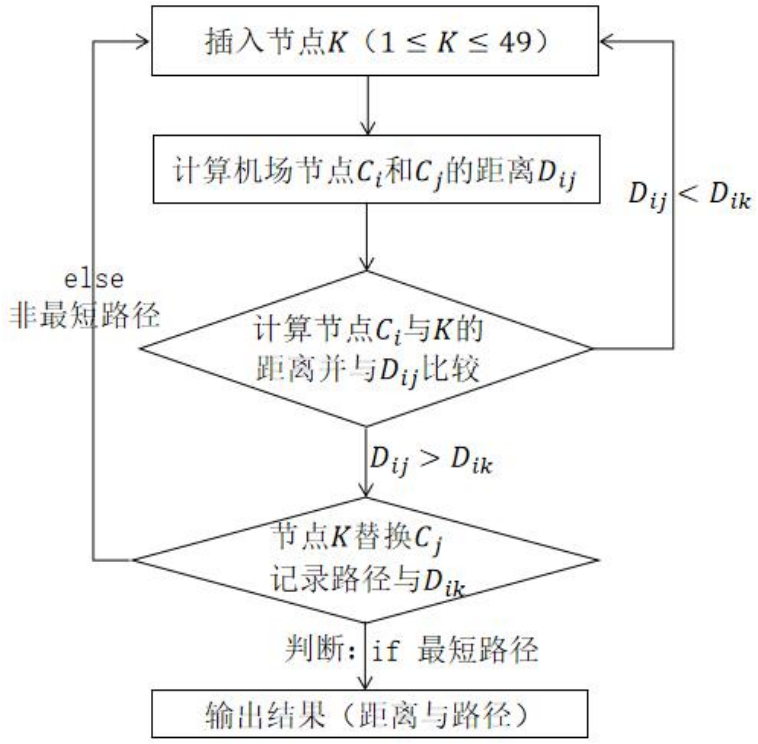


图 2 Floyd 算法流程

求出所有城市对间的最短路径后，紧接着我们能找出其中转机次数最多的城市对（即最短路径集合中的最长路径）。

然后再利用已经封装好的 Dijkstra 算法函数求得该路径所经城市（Matlab 代码见附录 2）。

数学语言表达如下：

决策变量：$dist$ 为两城市之间的路径。

目标函数：

$$max\ dist \tag{2}$$

约束条件：

$$dist \in D \tag{3}$$

其中，D 为各个城市之间最短路径构成的集合。

根据 Floyd 算法结果，和图 2 中的算法流程，我们找出最少转机次数线路中（即本题的乘机旅行路线中）转机次数最多的城市及其路径，如表 3 所示：

表 3 乘机旅行路线中最大转机次数及其路径

	路径	转机次数
路径代码	45-35-14-33-7-38-34	5
路径名称	YIW-SWA-HGH-PEK-CSX-TAO-SHE	
路径代码	45-35-14-33-7-39-9	5
路径名称	YIW-SWA-HGH-PEK-CSX-TYN-DLC	

其中路径名称多对应的城市分别为：

（1）义乌-汕头-杭州-北京-长沙-青岛-沈阳

（2）义乌-汕头-杭州-北京-长沙-太原-大连

（注：以上路径反向亦成立）

5.2 问题二模型的建立与求解

第二题我们建立赋权有向图模型，并基于实际情况改进 Floyd 算法进行求解。

5.2.1 模型的建立

第二题求在所有旅行路线中（即最短时间的路线）旅行时间最长的路线，我们先基于“最短时间”求解出所有旅行路线，再从中选出旅行时间最长的路线。

我们以航班的飞行时间为权重，建立赋权有向图，示意图如图 3 所示。

（注：为清晰观察路径，示意图中为标注权重，仅给出权重示意）

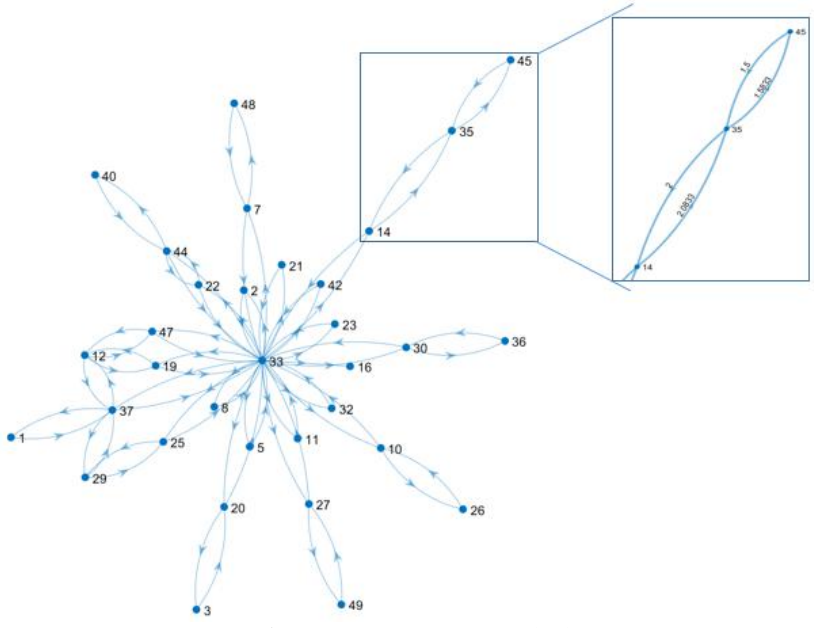


图 3 机场飞行时间赋权有向图

首先，我们确定最短时间。与一般最短时间问题不同，总的旅行时间除了考虑飞机航行中的耗时，还要考虑中转时的候机时间，即在每个城市/机场节点所停留的时间。

而追踪中转站的候机时间需要我们追踪每一次中转前的航班降落时间和下一轮航班的起飞时间。

最终的旅行总耗时如图所示：

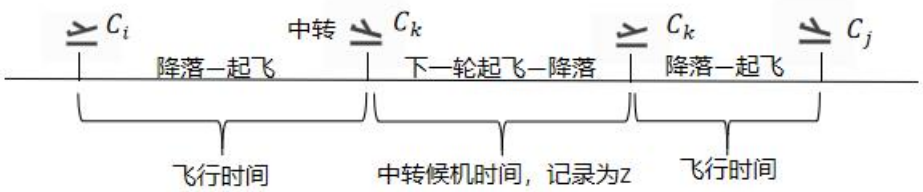


图 4 追踪时间足迹

由图可见，从城市 Ci 到城市 Cj 的耗时包括以下部分：

（1） 从 Ci 到中转城市/机场 Ck 的飞行耗时（降落时间减去起飞时间）；

（2） 在中转机场 Ck 等候下一班飞机的耗时（下一班飞机的起飞时间减去上一班的降落时间）。可能有多个中转机场；

（3） 从中转机场 Ck 到 Cj 的飞行耗时（降落时间减去起飞时间）。

根据图示分析，我们可以建立以下模型追踪旅行时间：

首先是对所有飞行时间的追踪，记T_f为总飞行时间，表示如下：

$$T_f = \sum_{i \in F} (t_{il} - t_{id}) \tag{1}$$

其中，t_{il}表示从 Ci 起飞的飞机的落地时间（24 小时制），t_{id}表示从 Ci 起飞的飞机的起飞时间，F 为旅途中所有飞机场的集合。

然后是对中转站候机时间的追踪，记中转站候机总时间为 Z，表示如下：

$$Z = \begin{cases} \sum_{k \in F'} (t_{kd} - t_{kl}) & t_{kd} \le 24 \\ \sum_{k \in F'} (24 \cdot 60 + t_{kd} - t_{kl}) & t_{kd} > 24 \end{cases} \tag{2}$$

其中，t_{kd}表示在 Ck 的下一班飞机的起飞时间（24 小时制），t_{kl}表示上一班飞机在 Ck 的降落时间，F'为集合 F 去掉起点机场和终点机场。**（注：此处我们假设，如果候机时间小于一小时，则无法赶上飞机，需要再额外等 24h）**

最后，总的旅行时长 T 可以表示为：

$$T = T_f + Z \tag{3}$$

基于以上分析，我们建立了三个矩阵：飞行时间矩阵 LL，起飞时间矩阵 LL2，落地时间矩阵 LL3，分别如下：

LL 部分示意为：

$$LL = \begin{bmatrix} 33 & 14 & 120 \\ 5 & 33 & 185 \\ \vdots & \vdots & \vdots \\ 33 & 42 & 125 \end{bmatrix} \tag{4}$$

其中第一二列为城市序号，第三列为飞行时间（单位：分钟）。

LL2 部分示意为：

$$LL2 = \begin{bmatrix} 33 & 14 & 410 \\ 5 & 33 & 415 \\ \vdots & \vdots & \vdots \\ 33 & 42 & 1355 \end{bmatrix} \tag{5}$$

其中第一二列为城市序号，第三列为起飞时间（起飞时间、落地时间以时间戳的形式记录，即 00：00 记录为 0000，00:01 记录为 0001。之后每增加一分钟依次加 1）。

LL3 部分示意为：

$$LL3 = \begin{bmatrix} 33 & 14 & 530 \\ 5 & 33 & 600 \\ \vdots & \vdots & \vdots \\ 33 & 42 & 40 \end{bmatrix} \tag{6}$$

其中第一二列为城市序号，第三列为降落时间（以时间戳的形式记录）。

5. 2. 2 模型的求解

为了在 Floyd 算法中实现对中转前后机场以及中转时间的追踪，我们在原本的 Floyd 算法中加入了 Path 矩阵与时间追踪变量 Z。

其中 Path 矩阵用于记录机场 K 前后的节点，即上一班降落在此的飞机的起飞机场以及即将起飞的飞机将要飞往的目的地机场。

而变量 Z（即上述模型中的 Z）在算法中以矩阵的形式放在 Ci 和 Cj 之间记录中转候机的时间。

完整的 Matlab 代码见附录 3。

此处，算法主要的改进部分如下图所示：

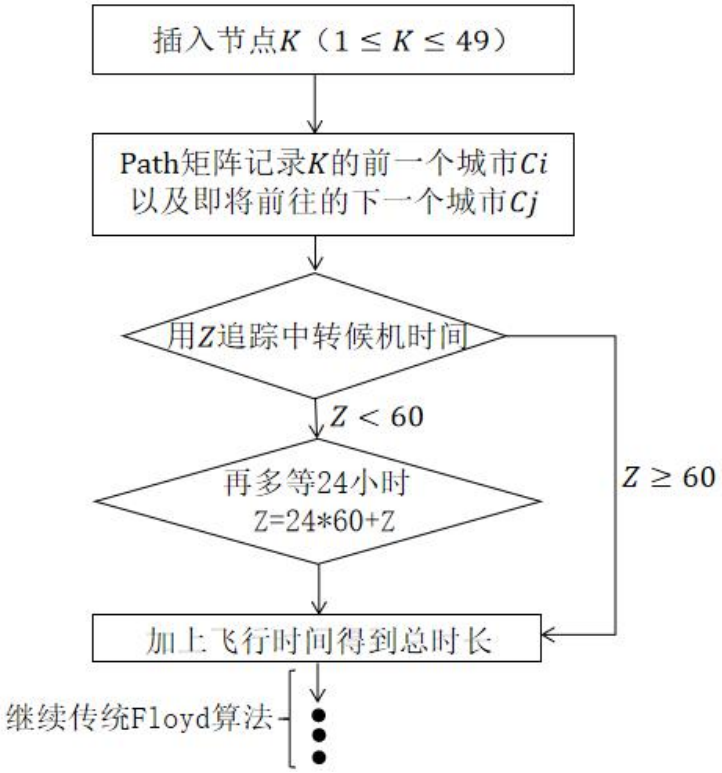


图 5 基于 Floyd 算法的改进

在记录并获得所有航班的总旅行时长之后，我们还绘制了箱线图以观察总旅行时长的总体分布，如图 6 所示：

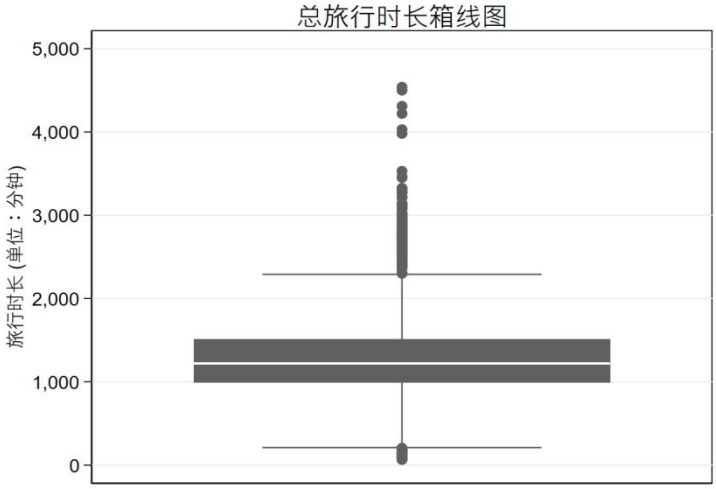


图 6 总旅行时长的箱线图

由图可见，总旅行时长的均值落在 1000 分钟至 1500 分钟之间，即平均旅行时长在 16 小时到 25 小时之间。同时，从图中可大致看出，满足 6 小时以内要求的线路有多条，经 Matlab 编程求解可知，共有 91 条线路满足此要求。

除此之外，图中有较大的极端值出现，这可能与旅行起点、中点以及中转途中各航班排班计划相关。

最终，问题二的所有解答如下：

第（1）问，**共有 91 条线路**可在 6 小时内到达。

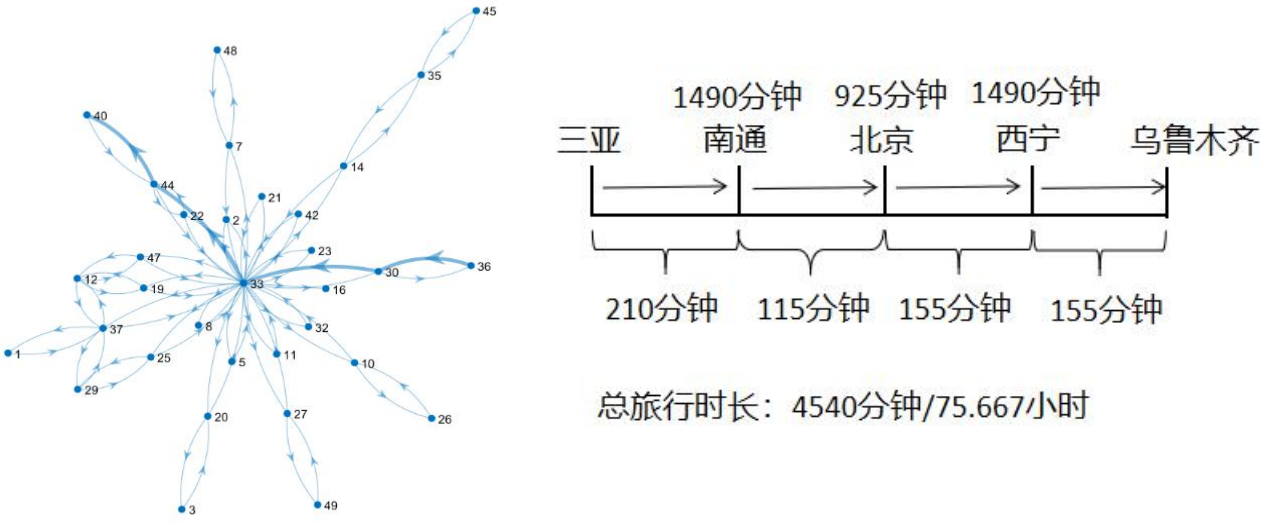
第（2）问，转机次数最多的线路中**转机 4 次**；转机次数最多的**线路有 20 条**。

第（3）问，答案如表 4 以及图 7 所示：

表 4 问题二中时间最长的线路

路径代码	36—30—33—44—40
路径名称	SYX—NTG—PEK—XNN—URC
城市名称	三亚→南通→北京→西宁→乌鲁木齐

第（3）问中，时间最长的路线以及对应时长如图 7 所示：



（a）时间最长路线示意图 **（b）路线对应时长与总耗时**

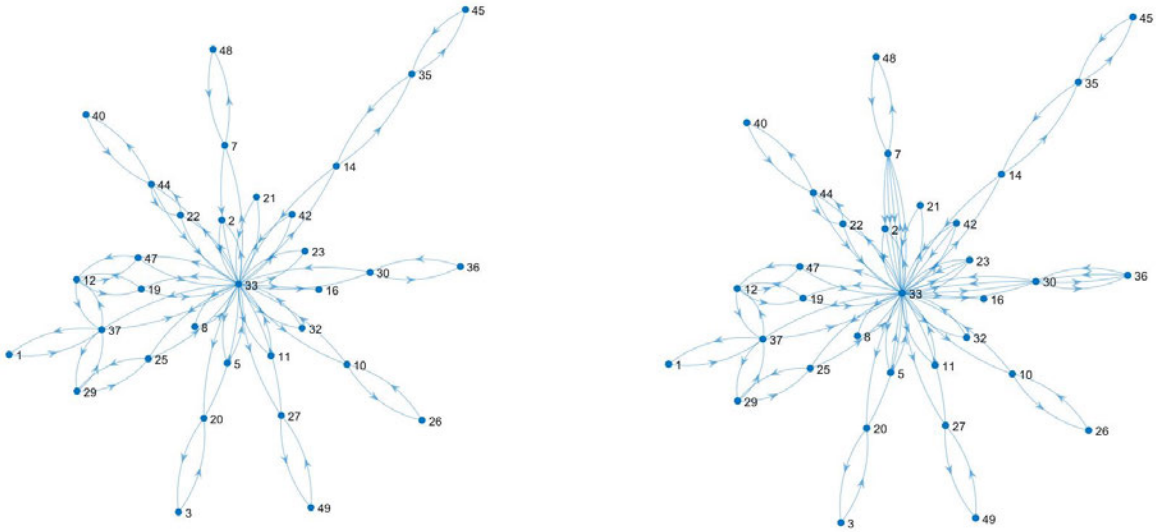
图 7 问题二中时间最长的线路

5.3 问题三模型的建立与求解

第三题我们依托于第二题中改进的弗洛伊德（Floyd）算法，利用穷举法的思想进行建模求解。

5.3.1 模型的建立

问题三与问题二最大的区别在于有无线路重复。根据题目所提供的数据绘图（如图 8 所示）可知，有线路的重复即两个城市之间存在不止一条线路。



（a）问题二的有向图　　（b）问题三的有向图

图 8 问题二与问题三的有向图对比

线路的不同带来的是总时长的变化。因此，求解第三问的关键在于更新第二问中的三个矩阵，即飞行时间（LL）、起飞时间（LL2）和落地时间（LL3）矩阵。

为了考虑到每一种线路组合的总旅行时长，加之此处总组合有限且数据量不大，故我们采用穷举法的思想进行如下建模。

首先，我们建立集合TP_{ij}，收纳所有城市对之间存在的所有线路。

$$TP_{ij} = C_{ij}^{(1)} \cup C_{ij}^{(2)} \cup C_{ij}^{(3)} \cup \ldots\ldots \cup C_{ij}^{(n)} \quad (i,j = 1、2、3\ldots\ldots 49) \tag{1}$$

其中，TP_{ij}表示C_i到C_j所有路线的集合；$C_{ij}^{(k)}$表示C_i到C_j的第 k 条路线的集合；n 为C_i到C_j之间的最大路线数量。

紧接着，我们计算出每个集合中每条线路所耗费的旅行时间，并用新的集合 S 来储存：

$$S_{ij} = T_{ij}^{(1)} \cup T_{ij}^{(2)} \cup T_{ij}^{(3)} \cup \ldots\ldots \cup T_{ij}^{(n)} \quad (i,j = 1、2、3\ldots\ldots 49) \tag{2}$$

其中，S_{ij}表示C_i到C_j所有路线耗时的集合；$T_{ij}^{(k)}$表示C_i到C_j的第 k 条路线的起飞时间、落地时间和飞行时间的集合；n 为C_i到C_j之间的最大路线数量。

最后，再取每一个S_{ij}中的每一个元素进行排列组合，组合所得到的元素作为新的起飞时间矩阵、落地时间矩阵和飞行时间矩阵的元素。

$$\varphi_{ij} = C_{S1}^{1} \cdot C_{S2}^{1} \cdot C_{S3}^{1} \cdot \ldots\ldots \cdot C_{SN}^{1} \quad (i,j = 1、2、3\ldots\ldots 49) \tag{3}$$

其中，φ_{ij}表示相应矩阵中位于(i,j)处的元素；C_{Sk}^{1}是组合数，其中 Sk 表示第 k 个S_{ij}集合，即第 k 个城市对的所有路线的耗时集合。

上述模型得到的三个新矩阵将输入到第二问改进的 Floyd 算法中进行求解。

5.3.2 模型的求解

对于问题三的求解，我们以问题二改进的 Floyd 算法为基础，在此之前更新了三个矩阵（Matlab 代码见附录 4）。

最终，问题三所有解答如下：

第（1）问，**共有 95 条线路**可在 6 小时内到达。

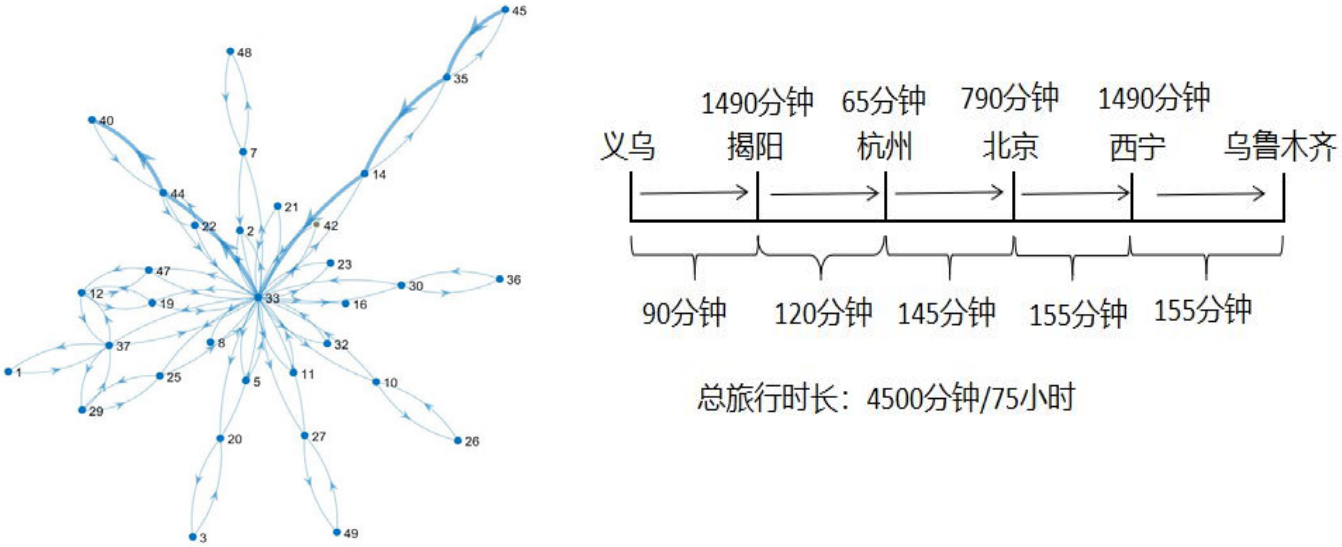
第（2）问，转机次数最多的线路中**转机 4 次**；转机次数最多的**线路有 20 条**。

第（3）问，答案如表 5 以及图 9 所示：

表 5 问题三中时间最长的线路

路径代码	45—35—14—33—44—40
路径名称	YIW—SWA—HGH—PEK—XNN—URC
城市名称	义乌→揭阳→杭州→北京→西宁→乌鲁木齐

第（3）问中，时间最长的路线以及对应时长如图 9 所示：



（a）时间最长路线示意图 **（b）路线对应时长与总耗时**

图 9 问题三中时间最长线路

5.4 问题四模型的建立与求解

经分析，问题四若基于上述的 Floyd 算法求解，其求解时间将会远超出我们的接受范围。因此，我们引入统计学的抽样模型，并基于此模型得到可靠的抽样结果，从而得出该题的近似解。

5.4.1 模型的建立

首先，为了方便区分国内航班（1 小时候机）和国际航班（1.5 小时候机），我们需要先区分国内机场和国际机场。因此，我们对所有机场按照国内、国际进行分类，并按照先国内再国际的的顺序对机场进行重新编号，以方便在算法中进行提取与判断。

各机场重新编号如下表所示：

表 6 新机场代码/节点编号

机场代码/节点编号							
AEB	$C1$	HRB	$C13$	NTG	$C25$	XNN	$C37$
BHY	$C2$	HUZ	$C14$	PEK	$C26$	YIW	$C38$
CAN	$C3$	JJN	$C15$	SHE	$C27$	YNT	$C39$
CKG	$C4$	KHN	$C16$	SWA	$C28$	YNZ	$C40$
CSX	$C5$	KMG	$C17$	SYX	$C29$	ZHA	$C41$
CTU	$C6$	KWL	$C18$	SZX	$C30$	ZUH	$C42$
DLC	$C7$	LHW	$C19$	TAO	$C31$	AOJ	$C43$
DYG	$C8$	LYG	$C20$	TYN	$C32$	BKK	$C44$
FOC	$C9$	MIG	$C21$	URC	$C33$	HKT	$C45$
HAK	$C10$	NGB	$C22$	WXN	$C34$	HNT	$C46$
HFE	$C11$	NKG	$C23$	XIY	$C35$	KLO	$C47$
HGH	$C12$	NNG	$C24$	XMN	$C36$	NZH	$C48$
						OSA	$C49$

紧接着是对**周航班**的考量，我们以一周为时间周期，以时间戳的形式记录起飞、落地时间。即，周　00:00 记为 0，周　的 00:01 记为 1，以　周为周期，之后每增加一分钟，时间戳随之加 1。到周二 00:00 就记录为 1440，周三 00:00 记录为 2880，一直到周末 24:00，时间戳更新（又从零开始）。时间戳图示如下：

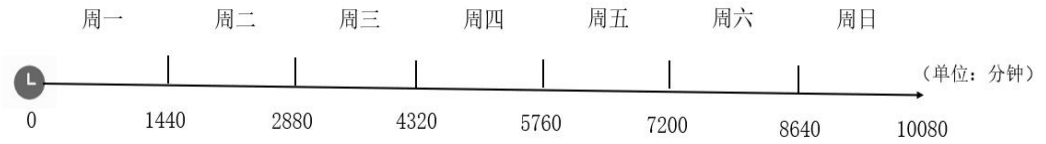


图 9 时间戳表示法示意图

在用时间戳的形式转化好所有数据的时间后，我们对所有城市间的路线的可能性进行组合。经过估算，其数量级在10^{101} -10^{137}之间。如图 10 所示。

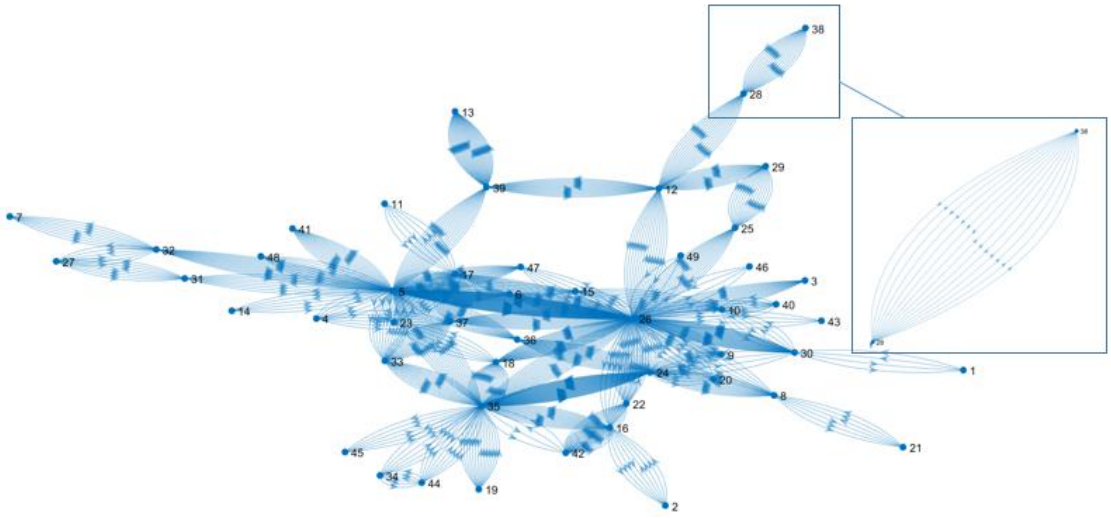


图 10 问题四有向多重图

若继续利用问题三中的穷举模型，其运算时间将远超过我们算法的接受范围。但基于其数据量大的特点，我们可以引入统计概率中的抽样模型，在总体中抽出可靠的样本量放入我们的算法中进行近似求解。

$$SS = \frac{Z^2 \cdot \hat{P} \cdot (1 - \hat{P})}{c^2} \tag{1}$$

其中，SS 为样本容量；Z 为统计上的 Z 值，每给定一个置信水平都有一个对应的 Z 值；$\hat{P}$为人为选择的百分比；c 为置信区间。

在本题的抽样中，我们选择 99%的置信水平，以及 5%的$\hat{P}$。经计算，得到我们的抽样样本容量应该为：96040000。

5.4.2 模型的求解

本题的求解便是基于上述的抽样公式，进行多次抽样实验，最终求得各题的近似解。所有实验与答案如下：（算法代码见压缩文件中 Q4）

第（1）问：我们共进行 12 次实验，结果如下表所示。

表 7 第（1）问抽样实验结果

第一次：191	第二次：192	第三次：192	第四次：192
第五次：192	第六次：192	第七次：192	第八次：192
第九次：191	第十次：192	第十一次：192	第十二次：192

最后，我们取其中的最大值 192

即，共有 192 条线路在 6 小时内到达。

第（2）问：同样地，我们共进行 12 次实验，结果如下表。

表 8 第（2）问抽样实验结果

第一次：639	第二次：639	第三次：639	第四次：639
第五次：639	第六次：639	第七次：639	第八次：639
第九次：639	第十次：639	第十一次：639	第十二次：639

由此可见，须转机一次的线路共有 639 条。

第（3）问：经过多次抽样实验，最终我们获得的最小值为 6540 分钟（109 小时），其路线如下表所示。

表 9 第（3）问抽样实验结果

路线	耗时
普吉岛→西安→北京→杭州→揭阳→义乌	**6540 分钟**
普吉岛→西安→长沙→烟台→杭州→揭阳→义乌	**6540 分钟**

由此，我们得到第（3）问的近似解为 6540 分钟，即 109 小时。但由于每次抽样的结果不同，因此路线可能会出现多条路线，我们的实验得出的路线即为表中所

示两条。

六、 模型的分析与检验

由于我们的模型所得到的答案唯一。故在此进行答案唯一性检验。我们采取 Excel 计算结果与我们的 Matlab 程序计算结果相比较的方法以确定我们模型所得到答案的唯一性以及正确性。

在此，我们以第二题第三问的答案为例进行检验。

如图 11 所示，左图为 Excel 中的计算数据，右图为 Matlab 程序中的距离矩阵。

机场	起飞	落地	机场
36	1430	1800	30

机场	起飞	落地	机场
30	1025	1330	36
30	1850	2045	33

机场	起飞	落地	机场
33	1210	1445	44

机场	起飞	落地	机场
44	1535	1810	40
44	1610	1715	22
44	2210	0025	33

D

49x49 double

	38	39	40	41	42
31	-1	-1	-1	-1	-1
32	-1	-1	2890	-1	400
33	-1	-1	1800	-1	125
34	-1	-1	-1	-1	-1
35	-1	-1	2920	-1	1870
36	-1	-1	4540	-1	2050
37	-1	-1	2930	-1	440
38	0	-1	-1	-1	-1
39	-1	0	-1	-1	-1
40	-1	-1	0	-1	3210
41	-1	-1	-1	0	-1

图 11 答案唯一性检验

通过左图日航班计划计算得到第二题第三问三亚（编码为 36）到乌鲁木齐（编码为 40）的最短旅行时间为 4540 分钟，与程序中距离矩阵 D 中答案一致（红圈中数字）。由此可见，我们的模型具有确定性。

七、 模型的评价、改进与推广

7.1 模型的优点

（1）本文的所有模型均基于弗洛伊德（Floyd）算法，具有较广的适用范围以及良好的实用性。对于数据量小的情况可以直接应用；对于数据量大的情况可以基于问题四的模型进行抽样应用。有助于真实的航空运营和管理。

（2）本文的模型巧妙结合了时间矩阵、时间戳等形式对问题进行灵活的转化，具有创新性与开拓性。

（3）本文的模型对问题的条件与实际情况均有所考量，具有充分性和完备性。

7.2 模型的缺点

对于问题二和问题三的模型与算法。当数据量增大时，算法的运行时间也会随之延长。当数据量达到一定数量级时，算法的运行时间可能超过接受范围，从而导致算法失效。而问题四的模型采用的是抽样求解的办法，需要大量重复的抽样实验以求得近似解而非精确解。

7.3 模型的改进与推广

（1）在问题二、三的求解中，我们假设不满足最低候机时间 1 小时的情况下，旅客会等候次日的飞机（>24 小时）。同样的，我们的模型也适用于多种其他情况。例如，我们假设不满足最低候机时间的情况下，游客会放弃此条线路，从而会影响到这道题的解答。改变假设后，问题二、三新解如下表：

表 10 新假设下问题二的解

第（1）问	**91 条线路**可以在 6 小时内到达
第（2）问	最多 4 次转机；转机次数最多的线路共 5 条
第（3）问	湛江→长沙→北京→东京

表 11 新假设下问题三的解

第（1）问	**95 条线路**可以在 6 小时内到达
第（2）问	最多 4 次转机；转机次数最多的线路共 4 条
第（3）问	湛江→长沙→北京→东京

（2）问题四除了利用概率统计模型进行抽样求得近似解，也可以理解为 DSSSP 问题，即动态单源最短路径问题的延伸，因此我们还可以利用现代智能算法，如蚁群算法等对该问题进行最优化方法的求解。

八、 参考文献

[1]姜启源，谢金星，叶俊. 数学模型(第三版）[M].北京：高等教育出版社，2003.85-130.

[2]孙云龙，唐小英. 数学模型与 MATLAB 应用 [M].成都：西南财经大学出版社，2021.12

附录

附录 1

介绍：第一题第(1)问 Matlab 程序，求解恰一次转机的城市对数量

```
A=zeros(49,49);
for i=1:150
    A(LL(i,1),LL(i,2))=1; %LL为城市对矩阵（由于太长，此处不给出，详情见压缩文件）
end
B=find(A==0);
A(B)=inf; %构造邻接矩阵
for i=1:49
    A(i,i)=0;
end
[D,path]=floyd(A);
[m,n] = find(D==2);
disp('城市对数')
size(m,1)
```

附录 2

介绍：第一题第（2）问， Floyd 算法和 Dijkstra 算法求最短路径

```
[D,path]=floyd(A);
imax=max(max(D));
[m,n] = find(D==imax);
m,n
[dist, path] = dijkstra(A, m(2), n(2));
    disp(['义乌与沈阳之间最短路径为(双向均成立)：']);
    disp(path)
[dist, path] = dijkstra(A, m(1), n(1));
    disp(['义乌与大连之间最短路径为（双向均成立）：']);
    disp(path)
function [D,path]=floyd(a) %Floyd算法
n=size(a,1);D=a; path=zeros(n,n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;
        end
    end
end
for k=1:n
    for i=1:n
        for j=1:n
            if D(i,k)+D(k,j)<D(i,j)
```

```
                D(i,j)=D(i,k)+D(k,j);
                path(i,j)=path(i,k);
            end
        end
    end
end
end
function [d, p] = dijkstra(adj, s, t) % 使用dijkstra求最短路径
nodes_num = size(adj, 1);
dist = inf(nodes_num, 1);
previous = inf(nodes_num, 1);
Q = [1:nodes_num]';
neighbors = cell(nodes_num, 1);
for i = 1:nodes_num; neighbors{i} = find(adj(i, :) > 0); end
dist(s) = 0;
while ~isempty(Q) % 取出距离最小点
    [~, min_ind] = min(dist(Q));
    min_node = Q(min_ind);
    Q = setdiff(Q, min_node);
    if min_node == t
        d = dist(min_node);
        p = generate_path(previous, t);
        return;
    end
    for i = 1:length(neighbors{min_node})
        neighbor = neighbors{min_node}(i);
        alt = dist(min_node) + adj(min_node, neighbor);
        if alt < dist(neighbor)
            dist(neighbor) = alt;
            previous(neighbor) = min_node;
        end
    end
end
d = dist;
p = cell(nodes_num, 1);
for i = 1:nodes_num; p{i} = generate_path(previous, i); end
end
function path = generate_path(previous, t)
path = [t];
while previous(t) <= length(previous)
    path = [previous(t) path];
    t = previous(t);
end
end
```

附录 3

介绍：第二题以及改进的 Floyd 算法

```
A=zeros(49,49);
for i=1:71
    A(LL(i,1),LL(i,2))=LL(i,3); %LL为城市对矩阵加一列飞行时间
end
B=find(A==0);
A(B)=inf; %构造邻接矩阵
for i=1:49
    A(i,i)=0;
end
for i=1:71
    A1(LL2(i,1),LL2(i,2))=LL2(i,3); %LL2为起飞时间数据（由于太长，此处不给出，详情见压缩文件；LL3同理）
end
B=find(A1==0);
A1(B)=inf; %构造起飞时间矩阵
for i=1:49
    A1(i,i)=0;
end
A2=zeros(49,49);
for i=1:71
    A2(LL3(i,1),LL3(i,2))=LL3(i,3); %LL3为落地时间数据
end
B=find(A2==0);
A2(B)=inf; %构造落地时间矩阵
for i=1:49
    A2(i,i)=0;
end
%第二题第（1）问
[D,path]=floyd(A,A1,A2);
[x,y]=find(D<=360);
disp(['6小时之内可到达的线路:',num2str(size(x,1)-49)])
%第二题第（2）问
[D,path,P]=floyd(A,A1,A2);
Q=find(P>99999);
P(Q)=-1;
disp(['转机次数最多的线路中转机次数：',num2str(max(max(P)))])
[x1,x2]=find(P==max(max(P)));
disp(['转机次数最多的线路条数：',num2str(size(x1,1))])
%第二题第（3）问
[D,path,P]=floyd(A,A1,A2);
Q=find(D>99999);
```

```
D(Q)=-1;
[v1,v2]=find(D==max(max(D)));
road(path,v1,v2)

%改进的Floyd算法
function [D,path,P]=floyd(a,A1,A2)
n=size(a,1);
P=ones(49,49);
D=a;
path=zeros(n,n);
for i=1:n
    for j=1:n
        if D(i,j)~=inf
            path(i,j)=j;
        end
    end
end
for k=1:n
    for i=1:n
        for j=1:n
            x=i;y=j; %在原Floyd算法上进行改进
                while path(x,k)~=k
                    x=path(x,k);
                    if x==0
                        break;
                    end
                end %求k的前继
                y=path(k,j); %求后继;
                if y==0
                    continue;
                end
                if x==0
                    continue;
                end
            if A1(k,y)<5000&A2(x,k)<5000&A1(k,y)>0&A2(x,k)>0
                    if A1(k,y)-A2(x,k)>0
                         z=A1(k,y)-A2(x,k); %时间差，即转机候机时间
                     if z<60
                         z=24*60+z;
                     end
                else
                    z=24*60-A2(x,k)+A1(k,y);
                    if z<60
                        z=24*60+z;%如果不满足最低候机时间，则额外等一天
```

```
                    end
                end
            else
            z=0;
            end
            if D(i,k)+D(k,j)+z<D(i,j)
                D(i,j)=D(i,k)+D(k,j)+z;
                P(i,j)=P(i,k)+P(k,j);
                path(i,j)=path(i,k);
            end
        end
    end
end
M=find(D>99999);
P(M)=inf;
P=P-1;
end

%定义路径函数
function pathway=road(path,v1,v2)
pathway=v1;
while path(pathway(end),v2)~=v2
    if path(pathway(end),v2)==0
        continue;
    end
    pathway=[pathway path(pathway(end),v2)];
end
pathway=[pathway v2];
end
```

附录 4

介绍：第三题

```
A=zeros(49,49);
for i=1:61
    A(LL(i,1),LL(i,2))=LL(i,3);
end
B=find(A==0);
A(B)=inf; %构造邻接矩阵
for i=1:49
    A(i,i)=0;
end
A1=zeros(49,49);
for i=1:61
    A1(LL2(i,1),LL2(i,2))=LL2(i,3);
```

```
end
B=find(A1==0);
A1(B)=inf; %构造起飞时间矩阵
for i=1:49
    A1(i,i)=0;
end
for i=1:61
    A2(LL3(i,1),LL3(i,2))=LL3(i,3);
end
B=find(A2==0);
A2(B)=inf; %构造落地时间矩阵
for i=1:49
    A2(i,i)=0;
end
%选出所有城市对之间的所有可能路径
S1=[7   33  1170    7   33  1310    7   33  140
7   33  440 7   33  570 7   33  130
7   33  570 7   33  700 7   33  130];
S2=[5   33  415 5   33  600 5   33  185
5   33  605 5   33  910 5   33  305];
S3=[33  7   630 33  7   775 33  7   145
33  7   1270    33  7   1405    33  7   135];
S4=[42  33  420 42  33  550 42  33  130
42  33  620 42  33  750 42  33  130];
S5=[33  23  595 33  23  785 33  23  190
33  23  1085    33  23  1245    33  23  160];
S6=[23  33  845 23  33  1010    23  33  165
23  33  1295    23  33  10  23  33  155];
S7=[33  30  480 33  30  580 33  30  100
33  30  595 33  30  720 33  30  125];
S8=[30  33  1130    30  33  1245    30  33  115
30  33  1315    30  33  1435    30  33  120];
S9=[36  30  870 36  30  1080    36  30  210
36  30  1060    36  30  1270    36  30  210];
S10=[30 36  625 30  36  810 30  36  185
30  36  785 30  36  1000    30  36  215];
k=1;
for i1=1:3
    combo=S1(i1,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
    for i2=1:2
    combo=S2(i2,:);
```

```
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
        for i3=1:2
    combo=S3(i3,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i4=1:2
    combo=S4(i4,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i5=1:2
    combo=S5(i5,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i6=1:2
    combo=S6(i6,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i7=1:2
    combo=S7(i7,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i8=1:2
    combo=S8(i8,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i9=1:2
    combo=S9(i9,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
            for i10=1:2
    combo=S10(i10,:);
    A1(combo(1),combo(2))=combo(3);
    A2(combo(4),combo(5))=combo(6);
    A(combo(7),combo(8))=combo(9);
    [D,path,P]=floyd(A,A1,A2);
```

```
    b1{k}=D; %储存距离矩阵
    b2{k}=path; %储存路径矩阵
    b3{k}=P; %储存节点数矩阵

    k=k+1;
            end
            end
            end
            end
            end
            end
            end
        end
    end
end

dist=inf(49,49);
P=zeros(49,49);
for i1=1:49
    for i2=1:49
        for i3=1:1536
            if b1{1,i3}(i1,i2)<dist(i1,i2)
                dist(i1,i2)=b1{1,i3}(i1,i2);
                P(i1,i2)=b3{1,i3}(i1,i2);
            end
        end
    end
end

[x,y]=find(dist<=360);
disp(['6小时之内可到达的线路条数:',num2str(size(x,1)-49)])
M=find(dist>100000);
P(M)=-inf;
disp(['转机次数最多的线路的转机次数：',num2str(max(max(P)))])
[x1,x2]=find(P==max(max(P)));
disp(['转机次数最多的线路条数：',num2str(size(x1,1))])
M=find(dist>100000);
dist(M)=-inf;
max1=max(max(dist));
[x1,x2]=find(dist==max1);
for i=1:1536
    if b1{1,i}(x1,x2)==max1
        Q=b2{1,i};
        break;
```

```
    end
end
disp(['时间最长的线路为：',num2str(road(Q,x1,x2))])

%对road函数的定义同第二题中的road函数（附录3中）
%对Floyd函数的定义同第二题中改进的Floyd函数（附录3中）
```