城市交通网络优化策略——以北京地铁为例

队伍：Mathorcup900494

2019年4月14日

城市交通网络优化策略——以北京地铁为例

摘要

随着我国城市化进程的推进，城市居住人口剧增，为了缓解交通运输压力，中国内地累计已超过30座城市建成并投运城市轨道交通。但由于客流量不断攀升，北京、上海、广州等地的轨道交通部分线路已呈现客流量过载、运力不足的问题。轨道交通作为大城市居民出行的主要载体，客流拥挤问题成为了降低运输效率、造成市民出行不便的重要因素。因此，制定科学合理的轨道交通优化方案，是缓解客流拥挤、提高运输效率的重要措施，该方案包括针对乘客出行的优化方案和轨道交通的乘客限流方案。

首先，我们基于附件1中的线网乘客O-D数据，计算了每个站点不同时段进站和出站人数的方差、平均值，乘客出行时段和出行距离、时间的方差、平均值等统计量，并结合这些站点在地理上的位置，分析不同站点乘客对轨道交通的需求量和出行规律，总结出乘客出行特征。然后结合以上统计量和其余附件的信息，我们从点（单个站点），线（单线），网（轨道网络）三个层次，对北京市轨道交通运输压力和客流量进行分析，并对每个站点不同时段的拥堵概率进行等级划分。其中划分依据主要基于三个指标，在此列表如下：(1)不同时段该站点进出站客流量，直接反映该站点附近乘客出行需求量；(2)该站点在轨道网络中的作用，包括可换乘站、起点站、终点站等；(3)该站点邻近站点的进出站客流量，可以衡量其它站点带来的运输负担。

对于问题2，我们基于迪杰斯特拉(Dijkstra)算法，设计出了一套智能路径规划算法，并用C++程序实现，该程序通过读取附录3的列车运行图数据，可以根据输入的乘客O-D信息，查询并自动还原该乘客完整的地铁出行路线。为了进一步优化乘客在有轨道交通路网中的路径选择，我们基于前面对站点拥堵情况的分析评估，生成邻接矩阵，表示选择特定站点的出行代价，并将该邻接矩阵加载入智能算法中，作为路径选择的依据之一，实现了缩短行程、减少拥挤的路径优化。

对于问题3，我们基于凸优化和对偶理论，构建了多目标的限流优化模型。限流的实质是通过调控各站点进站和乘车人数，缓解当前站点和邻近站点运输压力的同时，实现轨道线路各站点协同，使乘客平均延迟时间最小化和列车运载效率的最大化。基于附录的数据，我们对八通线上各站点包括实时客流量、出发站到不同终点站概率等进行数学表达，将不同站点不同时段的限流百分比当作优化变量，同时以列车满载量等为约束条件，建立模型，最终通过对偶变换和随机优化等方法，找出最优解，并分析不同限流方案的优劣。

最后，我们基于前三问的研究，提出具体限流措施，包括限流客流量的阈值，限流时段和车站的选择，并总结出普遍适用的限流策略，为轨道交通管理者提供建议。

关键词：轨道交通，客流量，出行特征，智能算法，路径优化，限流策略

Contents

List of Figures

List of Tables

1 问题重述

1.1 背景知识

地铁，指在地下运行为主的轨道交通系统。[1] 很多城市都修建了地铁，其主要目的是为城市服务，缓解交通拥堵问题。现在，地铁成为了城市发展的一张名片。在中国，无论是一线城市，还是像武汉、重庆等新一线城市，抑或厦门、长春等二线城市，都修建了地铁。并且在不断地修建新的地铁线路。

世界上第一条地铁是修建于1863年的伦敦。[1] 而中国的第一个开通地铁的城市是北京。于1971年开始运营一号线，从苹果园站到北京火车站。[2] 在2000年前，中国大陆只有北京、上海和广州这三个城市有少量地铁线路，其他城市都没有地铁。进入到21 世纪，地铁的建设如雨后春笋般在各大城市建立起来。截至2018年12 月31日，中国内地共有35座城市建成并投运城市轨道交通。其中，很多城市的地铁成为了当地居民或游客出行的主要载体，也是城市经济的动脉。

北京是中国的首都，是全国的政治中心和文化中心。不仅是世界著名古都，还是一座充满活力的现代化国际大都市。北京有着错综复杂的地铁线路，年载客量可达45.3亿次。[2]而现在，我们讨论的是北京地铁的综合调度问题，希望能够最大化优化乘客的出行。

作为一名地铁乘客，可以通过导航软件来规划自己的出行路线。导航软件（如百度地图、高德地图等）有自己的算法，可以计算出几条可行的路线，并告诉乘客哪条路线是最短的（或时间最少，换乘最少）以供乘客使用。并且能够把换乘的情况考虑在内。还有，对于某些乘客来说，舒适度也是考虑的因素。可能还要考虑地铁的拥挤程度以做出更智能的行程规划。

作为地铁的管理人员，对地铁情况进行调度和优化也是必要的。每个城市都有市区和郊区，市区的繁华程度也不一。所以，必然会有些站上下车的人比较多，有些站上下车的人比较少。然而，地铁的作用是方便每一位市民出行。如果说如果一个站人太多，导致地铁开到下一个站时总是全满，导致下一个地铁站的乘客等待很久都无法上车，这样对下一个站的乘客是不公平的。所以，我们需要通过限流的方式进行数据的优化，对人多的站进行限流，以保障每一个地铁站的乘客的公平。通过合适的调度，使得地铁的运载能力最大化，给当地民众和游客提供相对最好的服务，是每一个繁华城市的地铁公司需要认真考虑的问题。

Figure 1: 北京地铁徽标

我们通过分析北京地铁的相关数据，让管理人员能更好地管理地铁的运营，并对地铁乘客做出更好的决策。

1.2 相关数据

1. 题目提供了四个附录。附录1为“线网乘客O-D 数据”，里面有大约一百万个乘客的出行数据。包括始发车站、目的地车站、进站刷卡时刻、出站刷卡时刻。通过附录1，我们可以大致了解每个站在不同时间的拥挤程度和人们出行的总体规的律。附录2为基础数据-车站表和基础数据-线路表。通过附录2我们可以知道每一条地铁线路的走向。附录3 为列车运行图的数据，可能是某一天从早上地铁开始运行到中午12 点的列车运行数据。附录4 为直观的北京地铁线路图。通过附录3 可以对北京地铁的线路有一个直观的认识。

2. 按照题目所示，北京地铁在当时有328个站。每个站的经纬度都是确定的。因此，我们运用了高德地图坐标拾取系统[4] 来确定每个地铁站的经纬度。并通过经纬度来估算两个站之间的行走距离。

3. 旅行也是推动地铁运载的重要因素。所以我们通过携程旅行网[5]和高德地图[6]，了解了北京的前100 名旅游景点和北京的交通枢纽。并通过高德坐标拾取系统[4] 来获取其经纬度数据，还用高德位智指数[7] 去调取每个旅游景点的热度。通过社会特征去验证数据特征的正确性。

4. 为了更好地了解北京这个城市，我们去查询了诸多北京的相关资料。了解了北京的市区和郊区，住宅区、交通枢纽、CBD 等位置特点。以及现实生活中北京地铁的相关信息。

2 问题分析

第一题要通过附件来分析北京市人民的出行情况。需要分析乘客的出行时段分布、出行距离分布、出行市场分布等。我们可以用数理统计的方法来分析这些问题。然而，我们认为，还要分析地铁的运行效率，比如说同一条路线在两辆相邻的地铁的间隔时间等。另外，还要尽可能在那张表中读出尽量多的信息，以便对北京地铁有更全面的了解，有助于后续的研究。

第二题要设计算法还原乘客出行的准确信息，并计算特定乘客编号的完整出行线路。其实根据常识，我们乘坐地铁都会优先搭乘时间最短的线路。（如果有多于一个到达方法）但有一种特殊的情况，就是一条线上的人特别多，为了避免在那条线上因为拥挤而浪费时间，所以避开那条线，优先选择稍远一点的线。还有就是换乘站一般是大站，也是需要考虑的因素。

第三题要以八通线为例，研究一条相对拥挤的线路对当地居民的影响。因为列车的乘客数量有限，对于一个繁忙的地铁站可能要采取限流的措施。乘客需要在地铁站滞留较长

时间。所以，为了更好地照顾最多人的利益，减少他们的等待时间，地铁站是否要限流，用什么方式限流等问题。

第四题要提出具体的限流措施来提高地铁的服务水平。这也是地铁公司，交通部门的难题之一。现在北京有不少站进行了限流，但如何限流可以最大限度地优化乘客出行是值得分析的。限流和时间相关，和车站也相关。还分为不同的限流强度。

3 模型假设

1. 我们认为，所有的数据均为来源于题目或者权威机构的原始数据，这些数据真实可靠。但是，由于北京城市发展日新月异的变化，通过查询北京地铁官方网站的线路图[4]，发现在题目所示的线路图基础上又增加或者延长了一些线路。如西北部开通了16 号线，18号线的延长等。新线路的开通或延长必然会带来地铁相关数据的变化。由于研究问题的方便性，所有的地图以附录3 和附录1,2 相关数据所示的情况展开讨论。与此同时，在本篇论文中可能引用了一些外部的数据。我们在此假设，不考虑新增线对地铁布局的影响，所有的数据都是基于题目中的线路数据来统计的。

2. 我们在分析这个问题时，经常要引用外部数据。主要是高德地图数据，还有一些来源于维基百科、百度百科、国家统计局等数据。我们假设这些数据都是真实可靠的。

3. 虽然地铁会有极小可能性会出现故障，导致某条线停运一段时间。我们假设，数据和文章的分析均建立在“地铁没有出现故障”的基础上。

4. 我们认为，提供附件的数量足够大并且足够均匀。可以代表整个地北京地铁一个上午的运行特点。另外，不考虑地铁出现的特殊情况。另外，地铁的班次和地铁的繁忙程度相关。在单位时间内地铁的班次数可以说明该线路地铁的繁忙程度。对于条状线路，我们以起点站和终点站的发车频率来了解地铁的繁忙程度，对于环形线路，我们以相邻两个站的发车频率来了解繁忙程度。

5. 我们假设，每两班地铁的间隔最少为1分钟。如果出现间隔非常接近0 的，可能是因为来了比较长的一趟地铁，然后就把它视作两趟车。在数据预处理时会将它剔除。

6. 对于一些离群较大的乘客数据，我们可以认为出现了特殊问题。因此在分析之前进行了数据的预处理。

4 名词解释和符号说明

4.1 名词解释

1. 地铁限流[7]：为了避免大量客流对线路或路网造成过大压力，地铁运营方采取的一种短期应对措施。北京地铁有不少车站都采用了限流方式。八通线是限流的重灾区，除了东端土桥站和临河里站之外，其他的车站全部都是常态早高峰限流。

2. 拥挤程度：对于乘客对地铁的选择，我们把拥挤长度作为重要的指标。因为乘客一般都不愿意乘坐特别拥挤，导致很长时间不能上地铁的线路。但是，每个人对拥挤程度的把握不一。但我们把拥挤程度进行简单的量化评估。

4.2 符号说明

符号	含义
$\alpha_{(i,j)}$	i站点上车人数占空位的百分比
A	八通线上各站点不同时间段的上车总人数
α	关于$\alpha_{(i,j)}$的13×30特殊矩阵
$\beta_{(i,j)}$	代表i站点上车的乘客在j点下车的概率
B	由$\beta_{(i,j)}$组成的13阶矩阵
S	列车上的剩余人数
X_{in}	上车人数
A(i,j)	在第i个站点，第j个时间段新到达的人数
A	一个关于$A(i,j)$的13×30的特征矩阵
Δt_1	地铁的发车间隔
Δt_2	地铁的放行间隔
$x_n(i,j)$	j时段i站新到站乘客
$x_o(i,j)$	j时段在i站外等候的人
$x_w(i,j)$	j时段在i站内等候的人
$X_n(i,j)$	一个关于x_n的13×30的特征矩阵
$X_o(i,j)$	一个关于x_o的13×30的特征矩阵
$X_w(i,j)$	一个关于x_w的13×30的特征矩阵
T_1	发车特征值
T_2	放行特征值
T_{delay}	延迟时间
X_i	列车的总人数
P	该列车总共能容纳的乘客数
W_1	地铁的载客率
w_i	某个站的拥堵人数
W_2	总共的拥堵人数

Table 1: 符号说明

5 问题1的求解

5.1 数据预处理

关于附件1，存在一些特别的数据。可能是因为系统故障，有可能是因为有个别特殊的乘客。为了更好地分析数据，把以下的乘客数据剔除。

1. 同一个站进和同一个站出的乘客。
2. 进站和出站时间差大于三小时的乘客。
3. 刷卡站点代号不在附件1中列取的。
4. 在非地铁运营时间乘坐地铁的。

5.2 统计量的意义

基于提供的乘客O-D数据，对以下重要的统计量的含义依次进行说明。

1. 进站

   每一时间段进站客流量总数 直观上实时反映了进站乘车的乘客数目，在时间维度上反映了的该地居民对轨道交通的使用需求和出行习惯中的出行时间分布。进站客流量多的时段，可能是上下班高峰期，而早晨进站客流量大的很可能是住宅区，临近下班时间进站客流量大的可能是公司企业的聚集区。

   进展乘客总数 反映了该车站/线路的总进站客流量以及居民对该站点/线路的使用需求量。客流量大的站点/线路可能位于重要交通枢纽或热闹繁华地带。

   进站刷卡时间的方差 反映了该地居民出行时间的分布，若方差较大，则说明该站附近乘客出行习惯本身就具有错峰的特征，即出行人数在一天之内分布较为均匀。若方差较小，则乘客大都集中在某时间段出行，容易造成拥堵，给附近站点带来较大压力。

2. 出站

   每一时间段出站客流量 直观上反映了前往该地的乘客数目，以及对到达该站点的交通线路运载力的需求。同时根据时间维度和空间维度上反映的特点，可推断该站点附近区域的社会功能。如：如果某一时间段该站下车的人很多，说明通往该站的线路和站点压力较大，可推断该地为CBD或重要交通枢纽。

   出站刷卡时间的方差 反映了前往该地乘客到达时间的分布，若方差较大，则说明该站作为目的地的乘客出行本身就具有错峰的特征，即到达人数在一天之内分布较为均匀。

3. 出行距离/时长

出行距离/时间的平均值 反映了该站点附近居民出行对轨道交通的依赖性，以及该地居民的出行习惯。

出行距离/时间的方差 反映了该站点附近居民的出行规律，若方差较大，则说明该站点附近居民出行活动范围较大，方差较小，则该站点附近居民出行活动范围较小。

4. 进站和出站的统计变量之间的相关性

这些统计变量的组合可以更全面地反映出轨道交通站点/线路上的乘客出行的规律。且这些组合可关联到该地的社会经济属性，在客流量和高峰期等特征上反映了该地附近的功能和在城市中的地位：住宅区，交通枢纽，CBD（上班族集中的地方），旅游景点。比如：出行距离/时间较长的站点可能分布在郊区，这些站点附近居民通常需要乘坐轨道交通到市中心地带；若同一站点在进站和出站表现出相近的特征，客流量都很大，则可认为是交通枢纽（机场）和繁华地带，附近居民出行需求量大，地铁运载压力较大，容易造成堵塞/使用率低。

下面在对北京的轨道交通信息和乘客O-D数据进行实际分析时，将直接运用这些统计量的含义。

5.3 简单的分析

问题一是通过附件1-4分析乘客的出行特征。包括出行时段分布、出行距离分布、出行时长分布等。出行时段分布就是从什么时候出发到什么时候到达目的地。根据猜想，工作日上午的出行距离分布则为两个地方的距离如何。出行时长分布即为从进站到出站总共用了多长的时间。简单的分析如下。

1. 北京地铁站的分布

通过分析和资料的查找。我们得到，北京地铁的主要的站点都是在北京的中心区。根据了解，北京总共有6条环线。二环、三环是北京比较繁华的地段；四环、五环等地方稍微偏一点。我们观察图1和2，发现越靠近市区的地铁站越密集，也有很多站位于郊区。最近十几年，北京地铁在不断发展。因此很多地铁线路是连接市区和郊区的枢纽。如昌平线、15号线等。对北京地铁有初步印象有助于我们进一步研究。

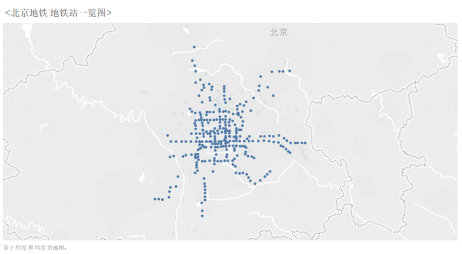

Figure 3: 北京地铁站的分布图

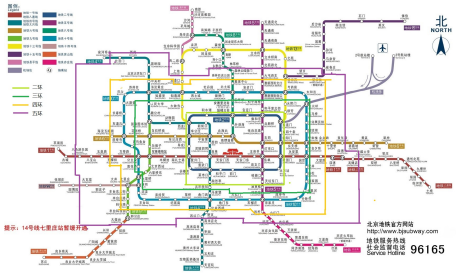

Figure 4: 北京地铁线路分环图

2. 整个上午最多人的车站排名

统计发现，13号线的回龙观、13 号线龙泽等地方的起始车站最多。可能是因为那里是重要的住宅区。目的地最多的车站是18号线的西二旗和2 号线的阜成门等。上车最多的站往往是住宅区，下车最多的站往往是旅游景点或工业区、CBD等。查阅资料，可以得知，进站最多的回龙观（龙泽、回龙观东大街）站、天通苑北（天通苑站）、沙河站均为北京历史悠久的住宅区。出站最多的西二旗在中关村软件园、盈创动力科技园等高新技术产业区；排名第2阜成门在四川大厦、万通金融中心、阜外医院等中心城区；排名第3的东直门则在北京的中央。附近有天安门广场、故宫、雍和宫、王府井大街等诸多旅游资源，所以下车的人很多。事实基本和我们的猜测相符合。

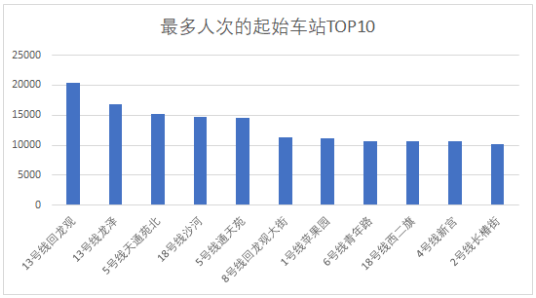

Figure 5: 北京地铁出发站排名TOP10

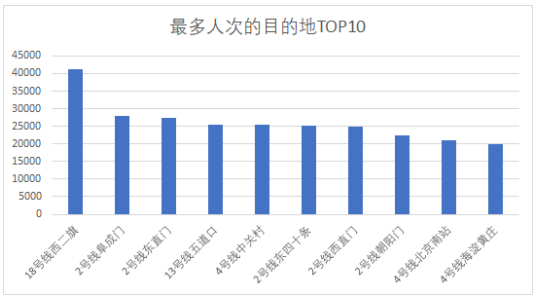

Figure 6: 北京地铁目的地排名TOP10

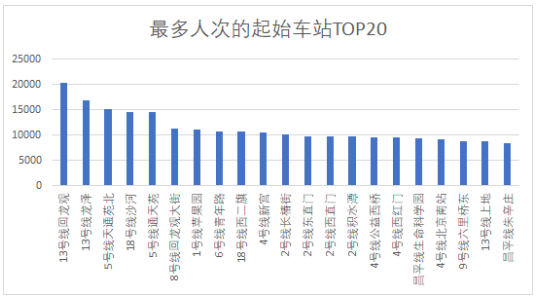

Figure 7: 北京地铁出发站排名TOP20

Figure 8: 北京地铁目的地排名TOP20

3. 以地铁线路作为分析

我们发现，进站最多的线路是2 号线和4号线。出站最多的线路是4号线，10号线和13号线。有一些线进站的人和出站的人都很多。有的站出站的人很多，但进站的人非常少，如14,15,16,17号线。根据地图，可以看到这些线路大多都是郊区线。我们可以认为，下车的是来旅游或者是上班的，但是当地居民的出行需求不高，因为他们就在附近上班。11号机场线有较多人进站，但很少人出站。说明乘坐飞机来北京出差或旅游的人较多，但是上午乘坐地铁去机场的人很少。

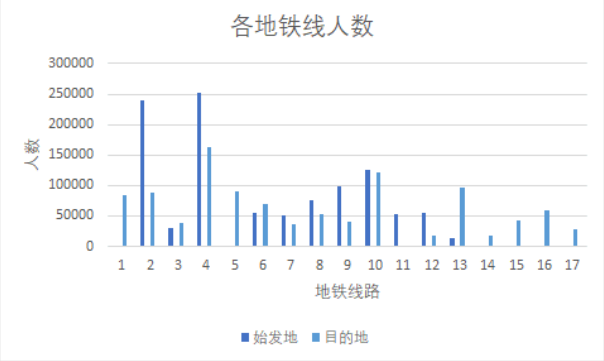

Figure 8: 各地铁线的人数

4. 对地铁其他站的上下车总体分析。

   我们发现，乘车地点的选择面更广。目的地更加集中。可能是因为每个人所住的地方不一，但是目的地相对来说没有那么多变化。另外，参考数据发现，有不少站是有人上车，没有人下车。这说明这些地方总体来说上车的人多，下车的人少。

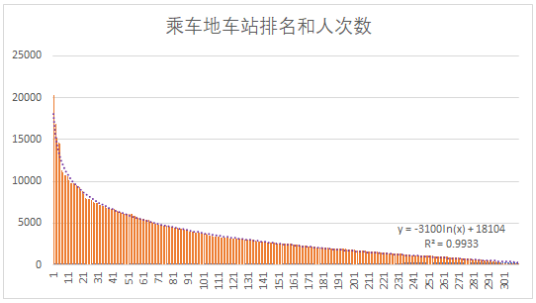

Figure 10: 北京地铁出发站总排名

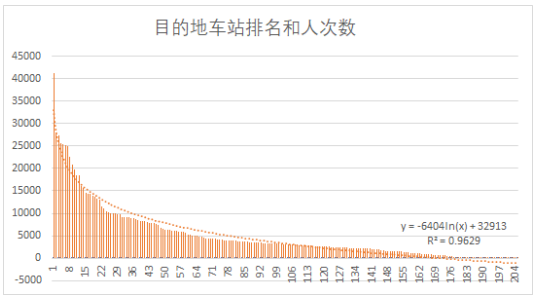

Figure 11: 北京地铁目的地总排名

另外，还将地铁的车站上下车量做成了热度图，以获得更直观感受。其中圆形越大表明来往越多。可以看到，上车比较多的地方是郊区，当然城区也不少。下车比较多的地方是二环、西三环沿线，以及西二旗。

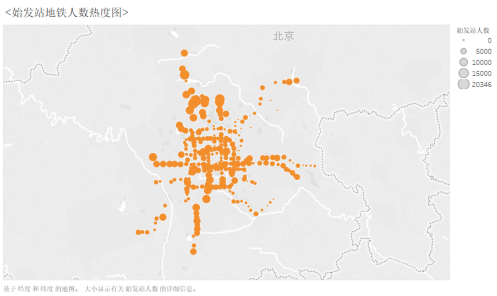

Figure 12: 出发站点热度图

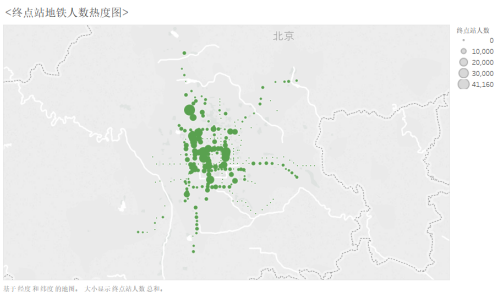

Figure 13: 目的地热度图

5. 地铁首班车发车时间

   分析表明。一般线路都会在5 点到6 点之间发车。最早发车的是5号线，5点准时发车。相对较晚发车的是机场线和15 号线，6点左右发车。可能是因为照顾某些起得比较早的上班族而较早地发车。

6. 地铁线路的繁忙程度

   分析发现，最繁忙的线路是2 号线。因为2号线是北京二环的环线，必然有很多人来往，会繁忙是在情理之中的。最不繁忙的线路是机场线和14号线东线。在早高峰期，1 号线、2 号线等都是为1∼2.5分钟一班地铁，可见北京地铁在上班期间非常繁忙，接近运行的饱和状态。

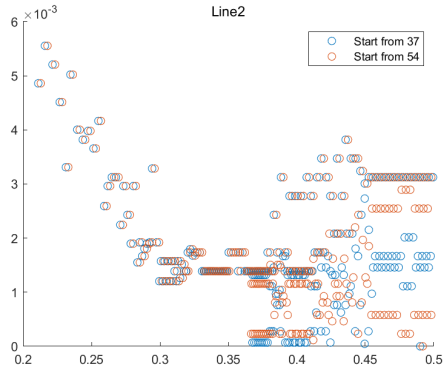

Figure 14: 2号线高峰期盛况（未剔除接近0的数据）

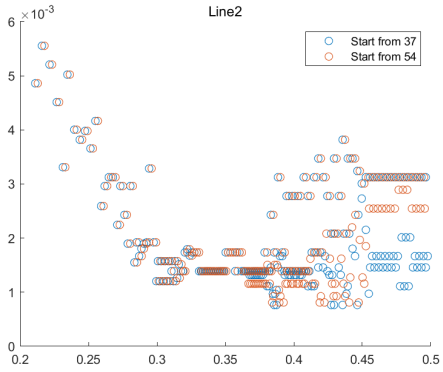

Figure 15: 剔除接近0的数据，仍可以看到地铁班次间隔最快1 分钟

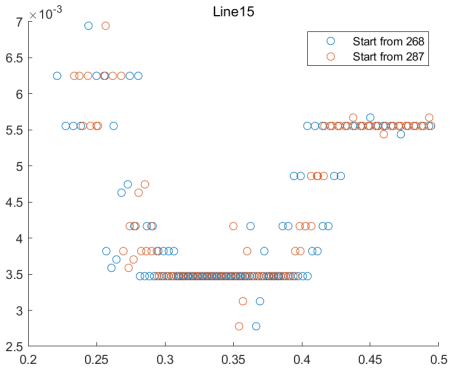

Figure 16: 相对不那么繁忙的15号线

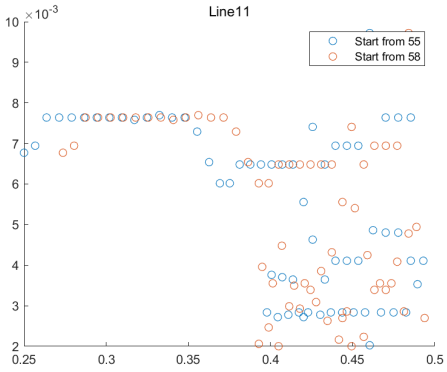

Figure 17: 班次较少的机场线

7. 地铁繁忙程度与旅游景点的关系：一般来说，旅游景点聚集的地方，乘坐地铁的人

也会比较多。如2号线基本上就是旅游景点聚集区。这条位于中央城区的内环线，环绕着的是中国古代光辉灿烂的历史，还见证了中国发展一个个奇迹。

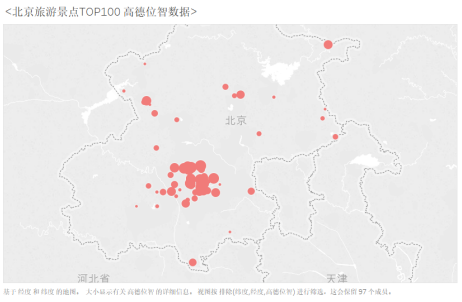

Figure 18: 北京重要旅游景点分布

Figure 19: 北京地铁2号线（内环线）

5.4 出行时段分布

为了分析出行时段，我们用进站人数和出站人数两个维度进行分析。

关于进站，主要集中在7点到9点。9点之后进站人数在减少，在10点左右达到较稳定。我们可以认为，7点到9点是早高峰。9点以后主要是去办事、业务人员或者是游客的出行。

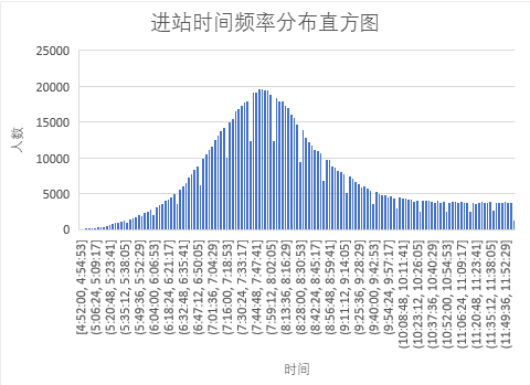

Figure 20: 进站时间频率分布直方图

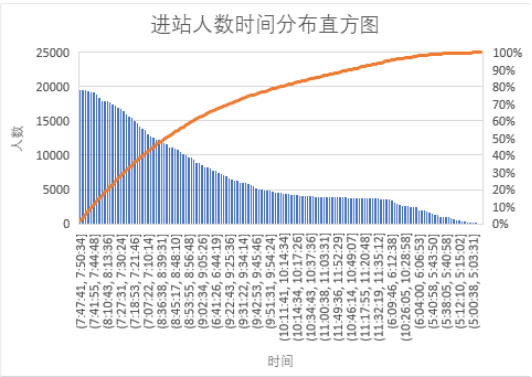

Figure 21: 进站时间段排名

关于出站，主要集中在7点半到9点半。可以大致地认为部分上班族乘坐半小时地铁最后到达了上班的地点。10点之后的主要是业务人员和游客。出站和进站的分析是对立统一的。

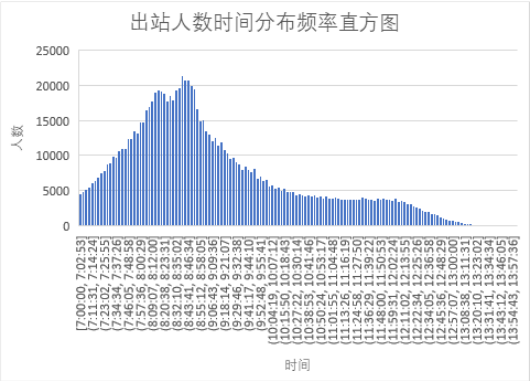

Figure 22: 出站时间频率分布直方图

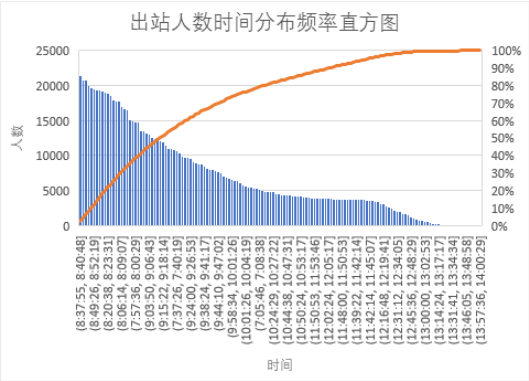

Figure 23: 出站时间段排名

5.5 出行距离分布

在这里，“出行距离”指的是两个地铁站之间的直线距离。我们使用了拾取坐标系统得到所有地铁站的经纬度，并计算出直线距离。

经过分析，出行距离分布如图所示。出行距离的平均值为14.79千米，中位数为13.08千米。出行距离主要集中在5∼20 千米。也有少数乘坐地铁超过30km。可能是居住在远郊的上班族或出远门的游客。

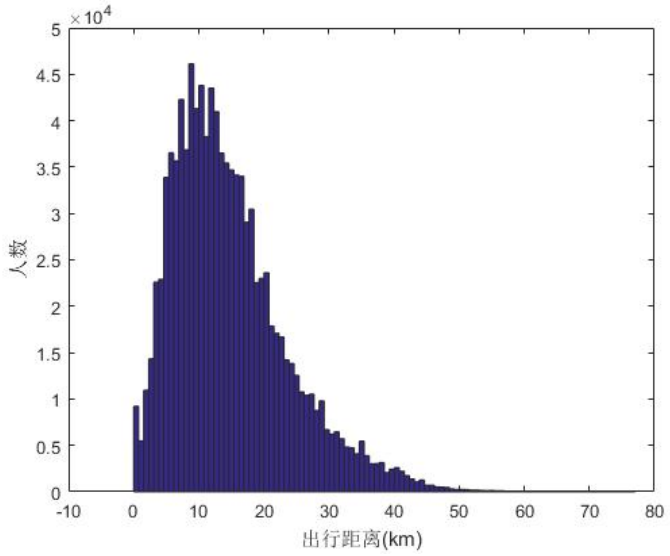

Figure 23: 出行距离分布图

5.6 出行时长分布

在这里，我们将出站刷卡时刻减去入站刷卡时刻得到出行时长。

经过分析，出行时长分布如图所示。出行时长的平均值为40分36秒。出行时长主要集中在20~50分钟。超过1小时15分的相对比较少。出行时间较长的可能是居住在远郊的上班族或出远门的游客，和出行距离比较远的是同一批人。

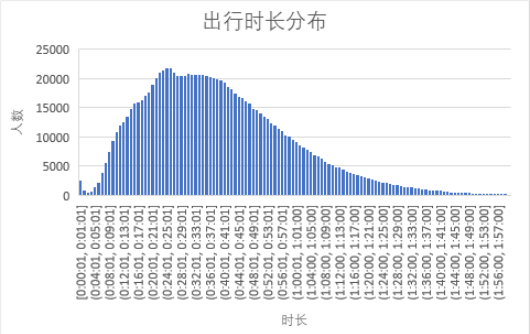

Figure 25: 出行时长分布

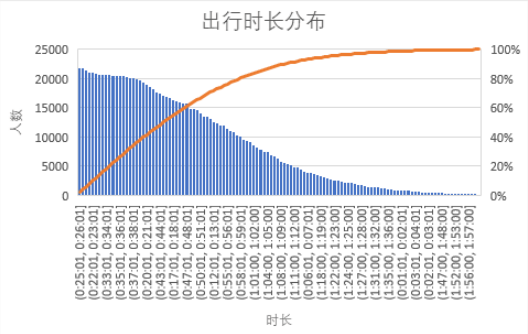

Figure 26: 出行时长排名

6 问题2的求解

6.1 线路规划算法

符合大部分人的出行习惯，乘客在选择轨道交通出行时，首要选择是换乘次数最少同时途径的站点数较少的路径，基于以上假设，还原乘客完整的地铁出行路径，实质上是设计一个匹配北京地铁网络和铁路出行数据的最短路径算法。而由于各个站点在不同时段的拥堵情况存在差异，同时受到客流量和限流方案等的制约，原本的最短路径可能并非是代价最小的路径，并且由于乘客没有掌握关于当前线路的运行情况的信息，直接进行最短路径的选择，具有盲目性，可能造成途径路线的拥堵等，因此通过智能算法，提供实时出行路径，有利于交通部门对乘客进行合理的分流和路径引导，缓解局部线路的运输压力。

6.2 还原乘客出行的线路

该优化算法的核心原理为Dijkstra算法和深度优先搜索，它是基于贪心策略，即每次选取到根节点代价最小的结点，使树向外伸展，直到伸展到目标结点为止。而权值矩阵(邻接矩阵)的生成，则是依据上述提到的（？）分析方法。

由于轨道交通数据量庞大，本算法的难点之一在于（1）邻接表和邻接矩阵的建立（因为庞大的数据和重复的计算，倘若人工输入邻接矩阵等信息，可能要花上大量时间，并且若要修改，或者加入新的路线站点，又要重新输入，不合适），（2）将附录3的地铁运行信息加载，并用一种高效的数据结构封装，便于查询和匹配。

为了解决以上两个问题，我们设计了一种自动根据地铁路线，创建邻接表和邻接矩阵的方法。程序依次读入每条路线的站点，每次读入当前站点后，与前一站点建立联系，倘若重复出现的站点，那么说明该站点在不同的线路中，是不同线路交汇的中转站，最终生

成包含邻接表，然后再根据邻接表内容建立邻接矩阵。就拥有了实现路径规划所需的数据内容了。

实现第一个要求，只需要将站点之间的途经代价设置为相距时间，对于指定的起点和终点，算法会自动规划出时间总长最少的路径，即符合大部分人出行习惯的路径，同时算法会根据进站时间和出站时间，去创建的存储了地铁线路数据的数据结构中，找到与进站时间相距适当时间的到站列车(我采用的策略为，假设从刷卡进站到列车站台，至少需要1.5 分钟，因此乘坐的列车到站时间为进站1.5分钟后的下一班列车)，以相同的策略，将距离出站时间最近的到达列车作为到达终点站乘坐的列车。

根据上述的算法，我们能还原每一位乘客的出行的数据。根据以下的描述，我们可以填写题目表2中的乘客出行准确数据。（由于位置不够，所以对于转了不止一次线的乘客无法一一列出来，但相关数据都可以通过程序获取。）

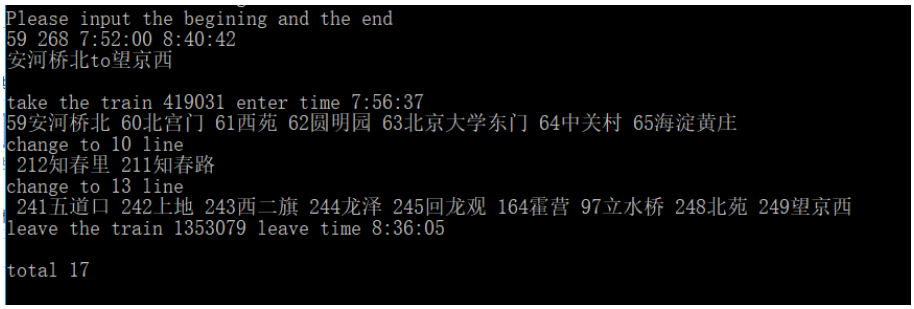

Figure 26: 地铁路线规划程序示例

乘客	车站	进站时间	首辆列车	换乘站	最后列车	车站	出站时间
2	西红门	8:28:32	419091	宣武门	213001	北京站	9:11:09
7	育新	7:02:08	809030	无	809030	鼓楼大街	7:27:09
19	光熙门	8:51:25	1349072	东直门	211157	前门	9:19:59
31	天通苑	7:34:26	500096	雍和宫	212040	西直门	8:13:30
41	苹果园	8:20:58	102044	复兴门	212126	雍和宫	9:13:47
71	天通苑北	9:05:34	501119	崇文门	211171	北京站	9:50:20
83	霍营	8:36:45	1353069	东直门	213005	东直门	9:17:59
89	十里堡	9:27:55	601066	无	601066	车公庄	9:53:10
101	七里庄	10:01:35	802481	南邵、生命科学园、东直门	212180	北京站	11:01:54
113	潞城	6:03:26	602036	湾子、菜市口	208016	北京站	6:52:38
2845	天宫院	7:29:56	414020	宣武门	211115	东四十条	8:41:55
124801	安河桥北	7:56:37	419031	海淀黄庄、知春路	1353079	望京西	8:36:05
140610	平西府	9:24:38	803583	霍营	1304029	望京西	10:01:24
164834	南礼士路	8:20:30	106101	复兴门、鼓楼大街、奥林匹克公园	106101	顺义	8:20:30
193196	光熙门	6:59:20	1353031	芍药居、惠新西街南口、大屯路东p	1353031	后沙峪	6:59:20
223919	草桥	8:02:35	101463	角门西、宣武门、积水潭	8:44:22	211105	积水潭
275403	草房	8:55:47	602019	什刹海、北土城、知春路	9:37:20	五道口	1353097
286898	高米店南	11:17:31	419149	角门西、十里河、大钟寺	11:17:31	龙泽	419149
314976	八角游乐园	7:26:18	106069	公主坟、知春路	1349076	五道口	8:16:40
315621	永安里	8:17:45	103104	建国门、东直门	1353085	光熙门	8:39:45

Table 2: 题目表2 乘客出行准确数据

6.3 智能优化算法

智能算法的核心在于将实时的轨道交通情况作为路径规划的指标。在建立模型时，我们将每个站点不同时段的拥堵概率和运载压力作为指标，生成权值矩阵，作为衡量途径该站点花费的代价标准之一，并在原本算法中加载该权值矩阵，使之自动规划出缩短行程、减少拥挤的路径。

权值(代价)矩阵生成的方法：我们从点（单个站点），线（单线），网（轨道网络）三个层次，对每个站点不同时段的拥堵概率进行等级划分。其中划分依据主要基于三个指标。分别如下：

1. 不同时段该站点进出站客流量，是从点的层面，直接反映该站点附近乘客出行需求量；

2. 该站点邻近站点的进出站客流量，可以衡量其它站点带来的运输负担，根据单线站点的协同关系，从线的层面，评估该站的拥堵等级；

3. 该站点在轨道网络中的作用，包括可换乘站、起点站、终点站等，这是在网络的层面进行评估。

其中对(1)号指标p_1，不同区间进行赋值，对进出站总人流进行归一化预处理后，乘以权重α_1。

对(2)号指标p2，赋予从1-10的等级，邻近的多个站点之间具有连锁的关系，以下列举三种主要类型。分别是多-多-多，多-少-多，少-少- 少，

	进站人数	出站人数
站点1	多	少
站点2	多	少
站点3	多	少

Table 3: 类型1：多-多-多

这是种病态的轨道交通运行状态，连续三个站点的居民都对地铁出行的需求量大，但由于每个站出站人数都很少，只能释放很少的运载空间，导致列车迅速满载，使站点2和站点3的乘客进站后无法乘车，造成拥堵，并且这种拥堵会加剧，增大了意外事故发生的风险。

因此途径这些站点或从这些站点上车或换乘，代价较大，我们赋予较高权值。假设这三个站点依次拥堵等级依次为5，7，10。

	进站人数	出站人数
站点1	多	少
站点2	少	多
站点3	多	少

Table 4: 类型2：多-少-多

这是种较为合理的轨道运载方案，站点1进入大量乘客，此时列车内乘客运载量大，但由于第二个站点大部分乘客出站，且该站的居民对轨道交通的需求量较小，很少乘客上车，使列车中运载空间得到释放，缓解了站点3 的运载压力，我们赋予较低权值。假设则这三个站点的拥堵等级依次为5，3，4。

	进站人数	出站人数
站点1	少	多或少
站点2	少	多或少
站点3	少	多或少

Table 5: 类型2：少-少-少

这是种运载效率较低的组合情况，连续几个站点都只进入少量乘客，此时列车内乘客运载量小，空间未充分使用，将乘客规划到这种线路上，可以起到分流、缓解拥堵线路压力的作用。因此将这些等级依次为3，2，1。

还有一些情况，由下表总结所示。

入站人数	出站人数	简要分析
多-多-多	少-少-少	压力大，乘客出行需求大，难以满足
多-少-多	少-多-少	第二个站点分担了相邻站点压力足
多-多-少	少-少-多	乘客都前往目的站点，导致目的站点压力大
少-少-多	多-多-少	利用率低的站点，大多都是出站，没有进站

Table 6: 几种典型情况的总结与简单分析

对于指标(3)，换乘时由于中转站汇集了多条线路，是乘客拥堵的高发地，并且由于换成导致线路复杂，也对乘客出行带来了较大的代价。因此对中转站。

6.4 典型站点分析——以西二旗，龙泽、回龙观为例

在我们之前对所有站点进行客流量统计时发现，西二旗为最多乘客出站的站点的，回龙观和龙泽为进站乘客最多的两个站点。而这三个站点又相连，所以选取这三个典型的站点进行分析。

Figure 27: 西二旗、龙泽、回龙观三个站的位置关系

对于西二旗站：进站少，出站多。

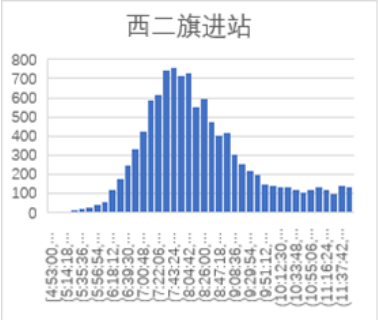

Figure 29: 西二旗站进展情况

Figure 30: 西二旗站出站情况

对于龙泽站：进展多，出站少。

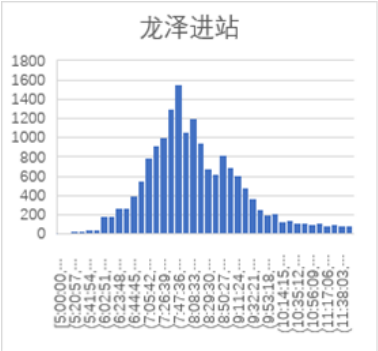

Figure 31: 龙泽站进展情况

Figure 32: 龙泽站出站情况

对于回龙观站：进展多，出站少。

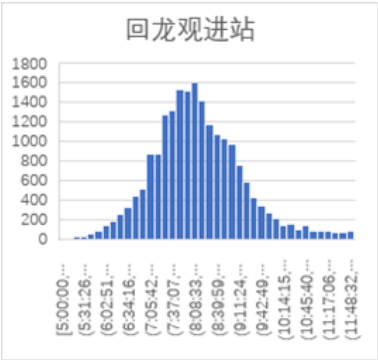

Figure 33: 回龙观站进展情况

Figure 34: 回龙观站出站情况

归纳成下表所示。

典型站点	进站人数	出站人数
西二旗	少	多
龙泽	多	少
回龙观	多	少

Table 7: 西二旗、龙泽、回龙观等站点的相关情况

西二旗出站人数多，而龙泽和回龙观上车人数多，这在很大程度上造成了该线的压力增大：一方面西二旗很多人集中下车，造成列车在站点沿线要运输大量前往西二旗的乘客，造成拥挤；另一方面，龙泽和回龙观上车人数集中，不仅造成该站点拥堵，列车在驶过这两个站后趋近饱和，在其他站的运载力会大大降低。

最终求出的权值(代价)矩阵C。C是一个$2 \times n$的矩阵。

$$C = \begin{bmatrix} c_{(1,1)} & c_{(1,2)} & c_{(1,3)} & \dots \\ c_{(2,1)} & c_{(2,2)} & c_{(2,3)} & \dots \end{bmatrix} \tag{1}$$

$$C = \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3 \tag{2}$$

c：每个站点的拥堵程度，λ指的是每个指标的权重。我们可以合理设置权重，让总的路线规划比较均衡。我们通过计算和测试可以得到：

$$C = \begin{bmatrix} 10 & 1 & 1 & 9 & 9 & 7 & \dots \\ 1 & 1 & 1 & 1 & 1 & 1 & \dots \end{bmatrix} \tag{3}$$

c的数值越大，说明线路经过该站点付出的代价越大。

为了进一步优化乘客在有轨道交通路网中的路径选择，我们在原本算法中加入实时的各站点的指标作为路径选择的参考依据，基于前面对站点拥堵情况的分析评估，生成邻接矩阵，表示选择特定站点的出行代价，并将该邻接矩阵加载入智能算法中，作为路径选择的依据之一，途径每个站点，都要在加上该站点的出行代价。实现了缩短行程、减少拥挤的路径优化。

在这里展示我们设计的C++算法。我们可以通过输入起点和终点的中文名，得到根据以上权值规划出最优路径。

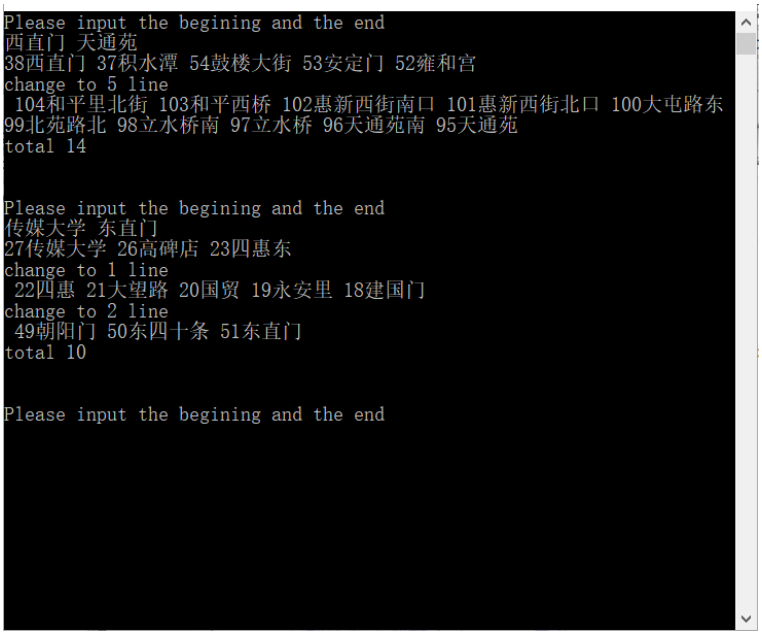

Figure 34: 算法成果展示1.智能规划两个站之间的线路

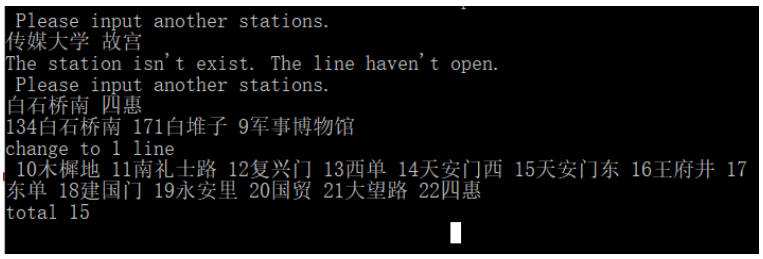

Figure 35: 算法成果展示2.若输入了不是地铁站的文字，将会提醒

7 问题3的求解

7.1 八通线概况

北京地铁八通线连接北京中心城区及东部的通州新城，属于郊区线。[8] 沿线共有13个站点，全长大约19公里。可在四惠和四惠东站换乘1 号线。早在2011年的新闻，就有说该线早上上班高峰期除了土桥站外全线限流。有一些线路是7时到9时，有一些线路是7时到8时30分。[9]

地铁的限流措施因各站情况不同而不一。地铁口单进单出、关闭进站通道、封站等都是限流措施。

这里引入A，B两个矩阵。

符号	大小	含义
A	$A \in R^{12\times30}$	A矩阵表示八通线各站点（将四惠和四惠东合并）在上午6 时-12 时每十分钟新到达的乘客的数量（含因限流在站外等候的与刷卡进站的）。A(i,j)表示在第i个站点，第j 个时间段新到达人数。
B	$B \in R^{14\times14}$	B矩阵表示在八通线上车的乘客，前往八通线其他各站点的可能性。第一个变量代表需搭乘其他线路在四惠或四惠东站换乘的乘客，其余十三个变量依次为八通线从四惠到土桥的13 个站点。B(i,j)代表乘客在八通线i站从上车前往八通线j 站点的可能性。

Table 8: A,B两个矩阵的介绍

我们可以分别筛选出在八通线各站点上车的乘客及其刷卡进站的时间数据，对于每个站点，统计其在上午六时到中午十二时，每十分钟为一个时间段的刷卡进站人数，以此推测出矩阵A中新到达乘客数（含刷卡进站的人数及因限流在站外等候的人数。）我们发现最偏远的土桥站在整个上午都无人光顾，其他的站的人数都不少。（由于版面限制，矩阵不能显示完全，完整的矩阵见附件）

$a_{(i,j)}$指的是在第j时间段，i站新到达的乘客数量。

$$A = \begin{bmatrix} a_(1,1) & a_(1,2) & a_(1,3) & \dots & a_(1,30) \\ a_(2,1) & a_(1,2) & a_(2,3) & \dots & a_(2,30) \\ a_(3,1) & a_(3,2) & a_(1,2) & \dots & a_(3,30) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_(12,1) & a_(12,2) & a_(12,3) & \dots & a_(12,30) \end{bmatrix} \tag{4}$$

通过计算，可以得到：

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 27 & 37 & 46 & 50 & 58 & \dots \\ 159 & 207 & 238 & 241 & 240 & \dots \\ 261 & 277 & 249 & 294 & 250 & \dots \\ 153 & 152 & 202 & 256 & 275 & \dots \\ 54 & 80 & 96 & 79 & 105 & \dots \\ 151 & 199 & 215 & 227 & 217 & \dots \\ 163 & 260 & 279 & 238 & 246 & \dots \\ 96 & 116 & 118 & 138 & 143 & \dots \\ 219 & 266 & 277 & 261 & 241 & \dots \\ 112 & 120 & 138 & 130 & 144 & \dots \\ 340 & 363 & 331 & 317 & 336 & \dots \end{bmatrix} \tag{5}$$

$\beta_{(i,j)}$指的是从i站出发的乘客到j站下车的比例。

$$B = \begin{bmatrix} 0 & \beta_(1,2) & \beta_(1,3) & \dots & \beta_(1,13) \\ \beta_(2,1) & 0 & \beta_(2,3) & \dots & \beta_(2,13) \\ \beta_(3,1) & \beta_(3,2) & 0 & \dots & \beta_(3,13) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_(13,1) & \beta_(13,2) & \beta_(13,3) & \dots & 0 \end{bmatrix} \tag{6}$$

通过计算，可以得到：

$$B = \begin{bmatrix} 0.000 & 0.000 & 0.000 & 0.146 & 0.107 & 0.149 & \dots & 0.110 \\ 0.000 & 0.000 & 0.000 & 0.039 & 0.081 & 0.089 & \dots & 0.210 \\ 0.000 & 0.000 & 0.000 & 0.081 & 0.079 & 0.095 & \dots & 0.151 \\ 0.707 & 0.000 & 0.000 & 0.000 & 0.015 & 0.013 & \dots & 0.047 \\ 0.853 & 0.000 & 0.000 & 0.016 & 0.000 & 0.010 & \dots & 0.020 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.728 & 0.000 & 0.000 & 0.077 & 0.019 & 0.059 & \dots & 0.000 \end{bmatrix} \tag{7}$$

与此同时，我们分别提取出前往八通线各站点的乘客及从八通线各站点出发的乘客。然后计算他们从该站点前往八通线各个站点的概率。

7.2 简要分析

本问建立的模型为论文的核心，基于前面的分析，我们已经建立了一套自己的轨道交通调控理论。

1. 地铁线路上的各站点具有协同和连锁关系，即每个站点会将自己的拥堵情况、运载压力传播到附近的站点，因此当拥堵情况升级时，为了避免传播效应，使下游站点的拥堵情况加剧，不得不采取限流的措施。

2. 衡量乘客出行代价的指标，包括途径站点数量、换乘次数、包括拥堵时间和乘车时间在内的总出行时间。

3. 限流措施具体实施措施可包括，封站、加绕行围栏等，但限流的实质是对当前站点所有具有乘车需求的乘客进行调度，以控制进站和上车人数比例等方式直接对人数变化进行调度，因此在本题中与具体实施措施无关，因此本问中不探讨实现限流的具体方式。

限流优化方案是通过调控各站点进站和乘车人数，缓解当前站点和邻近站点运输压力的同时，实现轨道线路各站点协同，使乘客平均延迟时间最小化和列车运载效率的最大化。

我们企图基于这些策略，建立动态的轨道交通客流量实时预测模型，同时将不同站点不同时段的限流百分比当作优化变量，把列车满载量等限制作为约束条件，建立模型，最终用对偶变换和梯度下降等方法，计算出最优化的限流方案，并分析不同限流方案的优劣。最终结合实际情况进行限流优化方案的设计。

7.3 优化变量与目标函数

由于新到站客流量和各站点出站概率确定，对轨道交通线路内总堵塞人数和堵塞情况的决定变量为限流矩阵。

主要关键在两个环节

1. 放多少比例乘客乘车，未上车的乘客会在站内等候（已过闸机）。

2. 放多少比例乘客进站，未进站的乘客在站外等候（未进闸机）。

在高峰时期，在各站进站口和乘车站台采取一定限流措施，限制部分乘客的进站和上车，实现对当前使用轨道交通的乘客规模进行控制，保证各站点乘客乘车的相对公平性，缩短所有乘客的乘车需求得到满足的平均时间。

不管是以封站、加绕行围栏等，背后实质都是通过调度乘客的数量，以实现使列车和站点内都具有合理客流量。

假设t_1为发车间隔。我们可以设定一个α函数来表示上车乘客和列车可容纳乘客数的比例。即有

$$\alpha = \frac{people_{in}}{people_{rest}} \tag{8}$$

我们对不同时段中限流措施体现在α矩阵中。

$$\alpha = \begin{bmatrix} \alpha_(1,1) & \alpha_(1,2) & \alpha_(1,3) & \dots & \alpha_(1,30) \\ \alpha_(2,1) & \alpha_(2,2) & \alpha_(2,3) & \dots & \alpha_(2,30) \\ \alpha_(3,1) & \alpha_(3,2) & \alpha_(3,3) & \dots & \alpha_(3,30) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_(13,1) & \alpha_(13,2) & \alpha_(13,3) & \dots & \alpha_(13,30) \end{bmatrix} \tag{9}$$

该矩阵各元素具有约束条件。当$0 < \alpha \neq 1$，若不采取限流措施，则$\alpha = 1$采取限流措施时，就会导致有一些相对较少的站的乘客上不了地铁。这显然不是优解。为了在实际情况中更方便轨道交通部门管理以及方案的实施，我们将同一站点在限流时段的限流比设置为相同，不同站点的限流比不同。即在限流时段，同一列的元素值相等。

限流主要是为了防止高峰期乘客的盲目乘车行为，造成在空间维度上的堵塞加剧，时间维度上的人均堵塞时间变长，由于在一段时间内乘车乘客数量一定，因此我们将目标函数设定为所有乘客由于限流和堵塞造成的延迟时间。

我们的优化目标是延迟时间的最小化。即为

$$\min T_{delay} \tag{10}$$

延迟时间主要由两部分组成，我们先把它称为T_1，T_2，我们再此建立一个简单的式子：

$$T_{delay} = \phi_1 T_1 + \phi_2 T_2 \tag{11}$$

其中T_1指的是在进站口外排队等候时间；T_2指的是在站台等待乘车的时间。此时，我们又可以定义T_1和T_2这两个函数。在定义这两个变量之前，还要定义以下的几个变量。$x_n(i,j)$指的是j时段i站新到站乘客。这个无法计算，只能通过调查与统计分析得出。$x_o(i,j)$指的是j时段在i站外等候的人。

$$x_o(i,j_1) = x_n(i,j_1) + x_o(i,j_1-1) - \alpha S \tag{12}$$

$x_w(i,j)$指的是j时段在i站内等候的人。

$$x_w(i,j_2) = x_n(i,j_2) + x_w(i,j_2-1) - \alpha S \tag{13}$$

其中的j_1和j_2又不相同，j_1指的是我们以每10分钟进行一次抽样。j_2指的是同一线路的两个班次的地铁的间隔时间。$X_n(i,j)$指的是一个关于x_n的13×30的特征矩阵。

$$X_n = \begin{bmatrix} x_n(1,1) & x_n(1,2) & x_n(1,3) & \dots & x_n(1,30) \\ x_n(2,1) & x_n(2,2) & x_n(2,3) & \dots & x_n(2,30) \\ x_n(3,1) & x_n(3,2) & x_n(3,3) & \dots & x_n(3,30) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n(13,1) & x_n(13,2) & x_n(13,3) & \dots & x_n(13,30) \end{bmatrix} \tag{14}$$

$X_o(i,j)$指的是一个关于x_o的13×30的特征矩阵。

$$X_o = \begin{bmatrix} x_o(1,1) & x_o(1,2) & x_o(1,3) & \dots & x_o(1,30) \\ x_o(2,1) & x_o(2,2) & x_o(2,3) & \dots & x_o(2,30) \\ x_o(3,1) & x_o(3,2) & x_o(3,3) & \dots & x_o(3,30) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_o(13,1) & x_o(13,2) & x_o(13,3) & \dots & x_o(13,30) \end{bmatrix} \tag{15}$$

$X_w(i,j)$指的是一个关于x_w的13×30的特征矩阵。

$$X_w = \begin{bmatrix} x_w(1,1) & x_w(1,2) & x_w(1,3) & \dots & x_w(1,30) \\ x_w(2,1) & x_w(2,2) & x_w(2,3) & \dots & x_w(2,30) \\ x_w(3,1) & x_w(3,2) & x_w(3,3) & \dots & x_w(3,30) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_w(13,1) & x_w(13,2) & x_w(13,3) & \dots & x_w(13,30) \end{bmatrix} \tag{16}$$

$$T_1 = \sum_{j=1}^{n} \sum_{i=1}^{n} x_o(i,j) \Delta t_1 \tag{17}$$

$$T_2 = \sum_{j=1}^{n} \sum_{i=1}^{n} x_w(i,j) \Delta t_2 \tag{18}$$

本题的约束条件均为线性约束，该模型的本质是一个优化问题，基于凸优化和对偶理论，我们对问题进行对偶变换，将有条件约束问题转化为无约束问题。

7.4 八通线两个限流效果最好的车站

根据我们在第二问优化路径算法中得出的“多多多”，“多少多”，“少少少”模型，我们选取连续几个进站客流都很大的站点进行限流，以保证乘客安全及避免列车严重超载；还对进站客流很小的站点进行限流，以期留出更多的空位给之后客流量大的站点的乘客，这将大量缩短后方拥堵站点的等待时间，从而减少整条线路总的等待时间。当前列车驶离本站时人数等于在前一站点时的人数减去本站下车人数加上本站上车人数，这是一个时滞系统。如下式：

$$x[n] = x[n-1] + on[n] - off[n] \tag{19}$$

本站上车人数就是我们要限流限制的人数。

下车人数可以根据问题三中算出的在八通线不同站点上车的乘客在此站下车的概率乘以在不同站点上车的人数得出。如下式：

$$off[n] = on[n-1] \times B[n-1,n] + on[n-2] \times B[n-2,n] + \ldots = \sum_{k=0}^{n-1} on[k] \times B[i,n] \tag{20}$$

经过不断迭代，可得出列车驶入本站时的可容纳量。

根据一二问的统计，在高峰时期，即早上七点到九点时，八通线进站客流绝大多数是土桥驶向四惠方向的乘客，符合我们在问题1中得出的从郊区前往市区上班的结论。进站客流最多的站点是土桥站，其次为梨园，双桥，果园，通州北苑，管庄，传媒大学，九棵树，临河里，八里桥，进站最少的为高碑店。

Figure 36: 八通线示意图

限流效果最好的两个车站是九棵树和果园。因为土桥、梨园是客流最大的两个车站，而果园是客流量第三大的车站，九棵树客流相对较小，典型的“多少多“模型。虽然限流九棵树少量增加了九棵树的等候时间，但是九棵树本来客流少，所以限流不会造成过大滞留，而在九棵树因限流留出的列车空位可以使九棵树后的客流大站果园站的滞留时间大大减少。该方案符合少量增加客流小车站的等候时间，大量减少客流大的车站的滞留时间的模型。

要想加快缓解交通滞留情况，我们可以通州北苑站加开空车，因为通州北苑站地处八通线中段，通州北苑站也是大站，列车驶过通州北苑后进站客流相对没有之前大，可以缓解通州北苑之后因来车满载而引起的滞留情况。

本模型的求解属于典型的长期约束优化（long-term constrained optimization），可通过动态规划和随机优化的方法实现，随机优化包括遗传算法、模拟退火算法、隐式马尔科夫模型（HMM）等。[10] 但由于由于时间不够充裕以及本优化模型的参数规模太大，我们没能在能够在规定时间前内求出最优解。但是，如果能提供更加充足的时间，我们一定能做出完整的方p案以及结果分析。

8 问题4的求解

8.1 动态限流方案

基于前三问的结果，我们引入了限流阈值，用于评估整条线路的运行情况。我们假设X_i为某个列车的总人数；P为该列车总共能容纳的乘客数。W_1 表示地铁内的载客率。

$$W_1 = \sum_{i=1}^{n} \frac{X_i}{P} \tag{21}$$

再引入变量w_i，表示每个站的拥堵人数。W_2则为整条地铁线路内的拥堵人数。

$$W_2 = \sum_{i=1}^{n} w_i \tag{22}$$

我们特别关注在不同优化方案中启动限流方案时的这两个指标。并将不同方案的这两个指

标进行横向对比，以发现其中的规律，在最优解时的这两个指标当作轨道交通的限流阈值，即当单线指标达到阈值时，启动限流方案并为今后的决策提供更灵活合理的参考依据。

动态限流方案则为根据实时监控的数据和平时运行的数据进行预测，基于前面的多站点协同组合模型，我们将站点分为健康，亚健康和病态三种状态。若线路已经触发限流阈值时，此时若有较大客流持续涌入，那么很快就会导致严重的拥堵情况发生。因此要开始启动限流方案，以应对可能产生的运输压力，当压力得到缓解，指标下降到阈值以下，再解除限流方案。第三题中求出的最优解和整条线路的运行指标作为限流时段选取的依据。

8.2 限流车站的选取

对象首先是客流量大的站点，才具有足够的限流空间。基于前面的分析，为了防止病态连续站点的累积效应，必须在中间选择限流效果最佳的站点进行较大规模限流，而其它邻近站点也要辅以小规模的乘客引导。

基于第二问的站点组合模型，我们已经总结出以下几种最高效的限流选择。

限流的强度可随着整条线路的指标浮动而动态调整，当拥堵指标超过阈值，且不断攀升的时候，加大前面各限流站点的限流强度。若指标突然出现较大波动，突然有大量乘客涌入交通系统，那么直接导致病态运行情况的发生，那么就启动更高级的限流方案。

当出现严重堵塞接近瘫痪的站点时，我们可以通过一些具体的调度措施，迅速释放要到达该站的列车容量。具体可采用交替限流互补等策略，可以起到相当于排队的效果，使后续车站的乘客也可以陆陆续续得到消化。这班该站只下不上，下一班让下一站只下不上中间站点大限流，边缘站点小限流等。

	进站人数	出站人数
站点1	少	多
站点2	多	少
站点3	多	少

Table 9: 常见的少-多-多情况

在中间站点下车乘客数量少，列车运输能力迅速达到饱和，使下游站点乘客无法被消化，乘客滞留过多，甚至出现运营瘫痪的情况。

到达此状态时，站点3已经属于病态，假设此时站点3仍然涌入一大批人，导致地铁站内严重拥堵。假设此状态形成后，采取交替限流方法：站点2暂停放行乘客进入车站，下一班列车经过站点2时，采取只下不上的策略，此时列车内中的容纳乘客空间都留给站点3用于消化拥堵的乘客。当站点2也滞留了超过一定量的乘客后，轮到站点2乘客允许上车，而站点3乘客只下不上。这样实现了在空间上的公平共享地铁资源的目的。

9 模型的评价

9.1 模型的优点

1. 第一问中，我们先从总体出发。了解了北京的地铁的总体架构和北京的城市规划。之后，我们利用频率分布直方图，排列图和热度图，全面而直观的展现了北京群众搭乘地铁的出行状况，并基于此，结合北京城市各功能区，准确地分析出北京市民或游客的出行特点。兼具科学性和人文关怀。

2. 第二问中，我们采用了传统的迪杰斯特拉（Dijkstra）算法极进行路径规划。在出行时间的基础上，还引入了拥挤程度、中转站两个因素的权值进行多元线性回归，以最小生成树的方式智能地进行路径规划，使路径更实时。具有深刻的理论基础和更好的实用性。

3. 第三问采用时滞迭代模型分析对某一站点限流在各站点间的互相影响及乘客滞留的总体情况，找出使整条线路出行时间最短的方案，具有总体性和前瞻性。虽然结果没有算出，但提供了一套值得继续研究的思路。

9.2 模型的不足

1. 关于地铁间隔的分析：在求解本题时没有意识到地铁有两个方向。理所当然地认为地铁一个站只有一个方向。虽然后来意识到了并修正了相关的程序代码。但是建立的思想可能存在基于原始思想的地方。这是需要改进的地方。

2. 由于时间有限，我们并没有完整的建立第三问模型，但是我们建立起了完整的结构及建模的思路和方向。若有足够时间，我们能建立出更完善的数学模型，对地铁限流提出建设性方案。

3. 由于提供的数据支持还不够大，以及对北京的了解，尤其是北京地铁规划不熟悉。所以我们只能通过高德地图和给出的数据来了解北京地铁的相关情况并进行数据处理与分析。由于北京站点共有328个，最近还在不断地增加。我们无法对北京地铁的所有站点都进行分析。并且由于附件只给了上午的数据，我们只能分析其早高峰时段的情况，无法分析其他时期的情况。

9.3 敏感性分析

该题有几点可以通过敏感性分析来检验其准确性。

1. 关于第一问的分析：我们可以根据地铁的相关数据来预测该地区的主要功能。这一般是都是正确的。上午作为始发站比较多的站多为住宅区，作为终点站比较多的多为CBD，旅游区等。我们之后查询了北京那几个地铁站周边情况，印证了该分析的正确性。

2. 关于路径规划问题的准确性。我们可以通过高德地图查询相关线路的路线规划以验证我们路线规划问题的准确性。我们就以从永安里到光熙门为例。发现和高德地图所示完全一样。验证了我们设计的算法的准确性。p


mm

Figure 38: 从永安里到光熙门（高德地图）

```
315621  永安里to光熙门

take the train 103104 enter time 8:17:45
19永安里 18建国门
change to 2 line
 49朝阳门 50东四十条 51东直门
change to 13 line
 252柳芳 251光熙门
leave the train 1353085 leave time 8:39:45
total 6
```

Figure 39: 从永安里到光熙门（我们的程序）

10 模型的应用

10.1 模型的可改进之处

1. 在开始解决这道题时，没有考虑到地铁线路的方向性。可改进之处是把地铁的方向性看做最先考虑的部分。并对线路进行分类。这样以便于更好地设计算法。

2. 我们完成此题时，由于时间不够充裕，没能把具体的结果计算出来，只设计出了基本思路。如果能够计算出结果，可能还需要进行一系列的修正。

3. 我们所做的优化模型是基于定在原本地铁线路的情况。但是，城市是在不断发展的，地铁的线路也会不断扩建。查阅新闻可知，北京正在新建17条地铁线路。当一条新线路开通后，在方便了沿线居民的同时使得调度问题更为复杂。如果模型能够预测未来新增线路之后的通行情况更好。

10.2 模型的推广

首先，在第二题中我们使用了智能算法为乘客规划地铁行程。相比现在已有的导航软件，还可以考虑到拥挤程度、减少换乘等其他方面。如果能够根据乘客的个人喜好设定不同的权值，能够成为理想的地铁导航指南。具有实时性、准确性和实用性。

我们建立的基于北京地铁客流量分析与限流控制的模型也可以的应用到全国其他城市的轨道交通运营管理方案中。并且可以指导其他城市的轨道交通建设和规划，只要将城市

划分出各个功能区，如商区，居住区，交通枢纽，CBD ，风景旅游区等，便可根据我们建立的模型规划可以减少拥堵同时满足出行的地铁线路。

不仅仅是轨道交通，只要改变一下约束条件，或者把更多的因素纳入考虑范围内，便可对道路交通的客运高峰进行分析，并对市民错峰出行缓解交通压力提出建设性意见。第二问还可以用于实施开车出行的导航路径的规划，综合考虑路径长短和拥堵情况的影响，使出行总时长达到最短。第三问的构思能对城市道路交通限流如单双号、封路等措施的实施提出良好的方案。不仅如此，该模型甚至还可以运用到城市转移等社会问题中，缓解一线城市的中心压力。

该模型作为一个动态模型，对交通工程、城市规划、社会学等均有裨益。

11 参考文献

[1]维基百科，地铁，https://zh.wikipedia.org/wiki/\%E5\%9C\%B0\%E9\%90\%B5，2019/4/13

[2]维基百科，北京地铁，https://zh.wikipedia.org/wiki/\%E5\%8C\%97\%E4\%BA\%AC\%E5\%9C\%B0\%E9\%93\%81，2019/4/13

[3]高德开发平台，坐标拾取系统，https://lbs.amap.com/console/show/picker，2019/4/12

[4]北京地铁线路图，北京地铁官方网站https://map.bjsubway.com/，2019/4/12

[5]携程旅行，北京市https://you.ctrip.com/sight/beijing1.html，2019/4/12

[6]高德地图https://www.amap.com/，2019/4/12

[7]百度百科，地铁限流https://baike.baidu.com/item/μ/10249049?fr=aladdin\#2，2019/4/13

[8]百度百科，北京地铁八通线https://baike.baidu.com/item/\%E5\%8C\%97\%E4\%BA\%AC\%E5\%9C\%B0\%E9\%93\%81\%E5\%85\%AB\%E9\%80\%9A\%E7\%BA\%BF?fromtitle=\%E5\%85\%AB\%E9\%80\%9A\%E7\%BA\%BF&fromid=5461430，2019/4/14

[9]北京晚报，八通线几乎全线限流https://m.hexun.com/news/2011-04-06/128518188.html，2019/4/14

[10] Tianyi Chen, Qing Ling,Georgios B. Giannakis, Learn-and-Adapt Stochastic Dual Gradients for Network Resource Allocation, IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, VOL. 5, NO. 4, 1941, DECEMBER 2018

12 附录

12.1 使用的软件

1. 文字处理软件：LaTeX。
2. 编程软件：C++ 和MATLAB R2018b。
3. 数据处理软件：画图，Excel，Origin 2018，Tableau 2018.3 和MATLAB R2018b。
4. 编程软件：MATLAB R2018b。

12.2 相关的代码

MATLAB代码：

1. a3.m

```
A = xlsread("subway.xlsx");
S = size(A)
A(:,6) = A(:,5) - A(:,4);
xlswrite('subway2.xlsx',A);
for i = 1:S(1)-1
    if A(i+1,2) == A(i,2)
        A(i,7) = A(i,3);
        A(i,8) = A(i+1,3);
        A(i,9) = A(i+1,4) - A(i,5);
    else
        continue
    end
end
xlswrite('subway3.xlsx',A)
X = zeros(328)
Y = zeros(328)
Z = zeros(328)
for i = 1:328
    for j = 1:328
        Z(i,j) = inf;
    end
end
```

```
for k = 1:328
    Z(k,k) = 0;
end
for i = 1:S(1)
    if A(i,7) ~= 0
        X(A(i,7),A(i,8)) = X(A(i,7),A(i,8)) + A(i,9);
        Y(A(i,7),A(i,8)) = Y(A(i,7),A(i,8)) + 1;
    end
end
for i = 1:328
    for j = 1:328
        if X(i,j) ~= 0
            Z(i,j) = X(i,j) / Y(i,j);
        end
    end
end
xlswrite('subway4.xlsx',Z)
```

2. a4.m

```
clear all
% 发车时间间隔问题
A = xlsread("subway.xlsx");
%% 构造一堆数组
% 一号线1: 1-23
line1_start_1 = [];
line1_reach_1 = [];
line1_start_23 = [];
line1_reach_23 = [];
% 二号线2: 37-54
% 二号线是一个环线，但我们以37号站作为起点站，54号线作为终点站。
line2_start_37 = [];
line2_reach_37 = [];
line2_start_54 = [];
line2_reach_54 = [];
% 八通线3: 24-36
line3_start_24 = [];
line3_reach_24 = [];
line3_start_36 = [];
```

```
line3_reach_36 = [];
% 四号线4: 59-93
line4_start_59 = [];
line4_reach_59 = [];
line4_start_93 = [];
line4_reach_93 = [];
% 五号线5: 94-116
line5_start_94 = [];
line5_reach_94 = [];
line5_start_116 = [];
line5_reach_116 = [];
% 六号线6: 131-150,300-307
% 六号线有两段
line6_start_131 = [];
line6_reach_131 = [];
line6_start_150 = [];
line6_reach_150 = [];
line6_start_300 = [];
line6_reach_300 = [];
line6_start_307 = [];
line6_reach_307 = [];
% 七号线7: 308-328
line7_start_308 = [];
line7_reach_308 = [];
line7_start_328 = [];
line7_reach_328 = [];
% 八号线8: 151-169
line8_start_151 = [];
line8_reach_151 = [];
line8_start_169 = [];
line8_reach_169 = [];
% 九号线9: 170-181
line9_start_170 = [];
line9_reach_170 = [];
line9_start_181 = [];
line9_reach_181 = [];
% 十号线10: 193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
line10_start_193 = [];
```

```
line10_reach_193 = [];
line10_start_237 = [];
line10_reach_237 = [];
% 机场线11: 55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
line11_start_55 = [];
line11_reach_55 = [];
line11_start_58 = [];
line11_reach_58 = [];
% 亦庄线12: 117-130
line12_start_117 = [];
line12_reach_117 = [];
line12_start_130 = [];
line12_reach_130 = [];
% 13号线13: 238-253
line13_start_238 = [];
line13_reach_238 = [];
line13_start_253 = [];
line13_reach_253 = [];
% 14号线14: 261-267,288-299
% 六号线有两段
line14_start_261 = [];
line14_reach_261 = [];
line14_start_267 = [];
line14_reach_267 = [];
line14_start_288 = [];
line14_reach_288 = [];
line14_start_299 = [];
line14_reach_299 = [];
% 15号线15: 268-287
line15_start_268 = [];
line15_reach_268 = [];
line15_start_287 = [];
line15_reach_287 = [];
% 昌平线18: 254-260
line18_start_254 = [];
line18_reach_254 = [];
line18_start_260 = [];
line18_reach_260 = [];
```

```
% 房山线19: 182-192
line19_start_182 = [];
line19_reach_182 = [];
line19_start_192 = [];
line19_reach_192 = [];
S = size(A);
%% 还要构造关于时间间隔的数组
% 一号线1: 1-23
line1_gap_1 = [];
line1_gap_23 = [];
%% 开始循环
for i = 2:S(1)-1
%% 处理一号线
    % 一号线的发车
    if A(i,3) == 23
        line1_reach_23 = horzcat(line1_reach_23,A(i,4));
        line1_start_23 = horzcat(line1_start_23,A(i,5));
    % 一号线的到站
    elseif A(i,3) == 1
        line1_reach_1 = horzcat(line1_reach_23,A(i,4));
        line1_start_1 = horzcat(line1_start_23,A(i,5));
%% 处理二号线
    % 二号线的发车
    elseif A(i,3) == 37
        line2_reach_37 = horzcat(line2_reach_37,A(i,4));
        line2_start_37 = horzcat(line2_start_37,A(i,5));
    % 二号线的到站
    elseif A(i,3) == 54
        line2_reach_54 = horzcat(line2_reach_54,A(i,4));
        line2_start_54 = horzcat(line2_start_54,A(i,5));
%% 处理八通线
    % 八通线的发车
    elseif A(i,3) == 24
        line3_reach_24 = horzcat(line3_reach_24,A(i,4));
        line3_start_24 = horzcat(line3_start_24,A(i,5));
    % 八通线的到站
    elseif A(i,3) == 36
        line3_reach_36 = horzcat(line3_reach_36,A(i,4));
        line3_start_36 = horzcat(line3_start_36,A(i,5));
```

```
%% 处理四号线
    % 四号线的发车
    elseif A(i,3) == 59
        line4_reach_59 = horzcat(line4_reach_59,A(i,4));
        line4_start_59 = horzcat(line4_start_59,A(i,5));
    % 四号线的到站
    elseif A(i,3) == 93
        line4_reach_93 = horzcat(line4_reach_93,A(i,4));
        line4_start_93 = horzcat(line4_start_93,A(i,5));
%% 处理五号线
    % 五号线的发车
    elseif A(i,3) == 94
        line5_reach_94 = horzcat(line5_reach_94,A(i,4));
        line5_start_94 = horzcat(line5_start_94,A(i,5));
    % 五号线的到站
    elseif A(i,3) == 116
        line5_reach_116 = horzcat(line5_reach_116,A(i,4));
        line5_start_116 = horzcat(line5_start_116,A(i,5));
%% 处理六号线
    % 六号线的131号
    elseif A(i,3) == 131
        line6_reach_131 = horzcat(line6_reach_131,A(i,4));
        line6_start_131 = horzcat(line6_start_131,A(i,5));
    % 六号线的150号
    elseif A(i,3) == 150
        line6_reach_150 = horzcat(line6_reach_150,A(i,4));
        line6_start_150 = horzcat(line6_start_150,A(i,5));
    % 六号线的300号
    elseif A(i,3) == 300
        line6_reach_300 = horzcat(line6_reach_300,A(i,4));
        line6_start_300 = horzcat(line6_start_300,A(i,5));
    % 六号线的307号
    elseif A(i,3) == 307
        line6_reach_307 = horzcat(line6_reach_307,A(i,4));
        line6_start_307 = horzcat(line6_start_307,A(i,5));
%% 处理七号线
    % 七号线的308号
    elseif A(i,3) == 308
        line7_reach_308 = horzcat(line7_reach_308,A(i,4));
```

```
        line7_start_308 = horzcat(line7_start_308,A(i,5));
    % 七号线的328号
    elseif A(i,3) == 328
        line7_reach_328 = horzcat(line7_reach_328,A(i,4));
        line7_start_328 = horzcat(line7_start_328,A(i,5));
%% 处理八号线
    % 八号线的151号
    elseif A(i,3) == 151
        line8_reach_151 = horzcat(line8_reach_151,A(i,4));
        line8_start_151 = horzcat(line8_start_151,A(i,5));
    % 八号线的169号
    elseif A(i,3) == 169
        line8_reach_169 = horzcat(line8_reach_169,A(i,4));
        line8_start_169 = horzcat(line8_start_169,A(i,5));
%% 处理九号线
    % 九号线的170号
    elseif A(i,3) == 170
        line9_reach_170 = horzcat(line9_reach_170,A(i,4));
        line9_start_170 = horzcat(line9_start_170,A(i,5));
    % 九号线的181号
    elseif A(i,3) == 181
        line9_reach_181 = horzcat(line9_reach_181,A(i,4));
        line9_start_181 = horzcat(line9_start_181,A(i,5));
%% 处理十号线
    % 十号线的193号
    elseif A(i,3) == 193
        line10_reach_193 = horzcat(line10_reach_193,A(i,4));
        line10_start_193 = horzcat(line10_start_193,A(i,5));
    % 十号线的237号
    elseif A(i,3) == 237
        line10_reach_237 = horzcat(line10_reach_237,A(i,4));
        line10_start_237 = horzcat(line10_start_237,A(i,5));
%% 处理机场线
    % 机场线的55号
    elseif A(i,3) == 55
        line11_reach_55 = horzcat(line11_reach_55,A(i,4));
        line11_start_55 = horzcat(line11_start_55,A(i,5));
    % 机场线的58号
    elseif A(i,3) == 58
```

```
        line11_reach_58 = horzcat(line11_reach_58,A(i,4));
        line11_start_58 = horzcat(line11_start_58,A(i,5));
%% 处理亦庄线
    % 亦庄线的117号
    elseif A(i,3) == 117
        line12_reach_117 = horzcat(line12_reach_117,A(i,4));
        line12_start_117 = horzcat(line12_start_117,A(i,5));
    % 亦庄线的130号
    elseif A(i,3) == 130
        line12_reach_130 = horzcat(line12_reach_130,A(i,4));
        line12_start_130 = horzcat(line12_start_130,A(i,5));
%% 处理十三号线
    % 十三号线的238号
    elseif A(i,3) == 238
        line13_start_238 = horzcat(line13_start_238,A(i,4));
        line13_reach_238 = horzcat(line13_reach_238,A(i,5));
    % 十三号线的253号
    elseif A(i,3) == 253
        line13_reach_253 = horzcat(line13_reach_253,A(i,4));
        line13_start_253 = horzcat(line13_start_253,A(i,5));
%% 处理十四号线
    % 十四号线的261号
    elseif A(i,3) == 261
        line14_reach_261 = horzcat(line14_reach_261,A(i,4));
        line14_start_261 = horzcat(line14_start_261,A(i,5));
    % 十四号线的267号
    elseif A(i,3) == 267
        line14_reach_267 = horzcat(line14_reach_267,A(i,4));
        line14_start_267 = horzcat(line14_start_267,A(i,5));
    % 十四号线的288号
    elseif A(i,3) == 288
        line14_reach_288 = horzcat(line14_reach_288,A(i,4));
        line14_start_288 = horzcat(line14_start_288,A(i,5));
    % 十四号线的289号
    elseif A(i,3) == 299
        line14_reach_299 = horzcat(line14_reach_299,A(i,4));
        line14_start_299 = horzcat(line14_start_299,A(i,5));
%% 处理十五号线
    % 十五号线的268号
```

```
    elseif A(i,3) == 268
        line15_reach_268 = horzcat(line15_reach_268,A(i,4));
        line15_start_268 = horzcat(line15_start_268,A(i,5));
    % 十五号线的287号
    elseif A(i,3) == 287
        line15_reach_287 = horzcat(line15_reach_287,A(i,4));
        line15_start_287 = horzcat(line15_start_287,A(i,5));
%% 处理昌平号线
    % 昌平线的254号
    elseif A(i,3) == 254
        line18_reach_254 = horzcat(line18_reach_254,A(i,4));
        line18_start_254 = horzcat(line18_start_254,A(i,5));
    % 昌平线的260号
    elseif A(i,3) == 260
        line18_reach_260 = horzcat(line18_reach_260,A(i,4));
        line18_start_260 = horzcat(line18_start_260,A(i,5));
%% 处理房山线
    % 房山线的182号
    elseif A(i,3) == 182
        line19_reach_182 = horzcat(line19_reach_182,A(i,4));
        line19_start_182 = horzcat(line19_start_182,A(i,5));
    % 房山线的192号
    elseif A(i,3) == 192
        line19_reach_192 = horzcat(line19_reach_192,A(i,4));
        line19_start_192 = horzcat(line19_start_192,A(i,5));
    end
end
%% 由于时间没有按顺序，所以给这些时间排个序。
% 以便于之后的数据分析。
% 一号线1：1-23
line1_start_1 = sort(line1_start_1);
line1_reach_1 = sort(line1_reach_1);
line1_start_23 = sort(line1_start_23);
line1_reach_23 = sort(line1_reach_23);
% 二号线2：37-54
line2_start_37 = sort(line2_start_37);
line2_reach_37 = sort(line2_reach_37);
line2_start_54 = sort(line2_start_54);
line2_reach_54 = sort(line2_reach_54);
```

```
% 八通线3: 24-36
line3_start_24 = sort(line3_start_24);
line3_reach_24 = sort(line3_reach_24);
line3_start_36 = sort(line3_start_36);
line3_reach_36 = sort(line3_reach_36);
% 四号线4: 59-93
line4_start_59 = sort(line4_start_59);
line4_reach_59 = sort(line4_reach_59);
line4_start_93 = sort(line4_start_93);
line4_reach_93 = sort(line4_reach_93);
% 五号线5: 94-116
line5_start_94 = sort(line5_start_94);
line5_reach_94 = sort(line5_reach_94);
line5_start_116 = sort(line5_start_116);
line5_reach_116 = sort(line5_reach_116);
% 六号线6: 131-150,300-307
% 六号线有两段
line6_start_131 = sort(line6_start_131);
line6_reach_131 = sort(line6_reach_131);
line6_start_150 = sort(line6_start_150);
line6_reach_150 = sort(line6_reach_150);
line6_start_300 = sort(line6_start_300);
line6_reach_300 = sort(line6_reach_300);
line6_start_307 = sort(line6_start_307);
line6_reach_307 = sort(line6_reach_307);
% 七号线7: 308-328
line7_start_308 = sort(line7_start_308);
line7_reach_308 = sort(line7_reach_308);
line7_start_328 = sort(line7_start_328);
line7_reach_328 = sort(line7_reach_328);
% 八号线8: 151-169
line8_start_151 = sort(line8_start_151);
line8_reach_151 = sort(line8_reach_151);
line8_start_169 = sort(line8_start_169);
line8_reach_169 = sort(line8_reach_169);
% 九号线9: 170-181
line9_start_170 = sort(line9_start_170);
line9_reach_170 = sort(line9_reach_170);
line9_start_181 = sort(line9_start_181);
```

```
line9_reach_181 = sort(line9_reach_181);
% 十号线10: 193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
line10_start_193 = sort(line10_start_193);
line10_reach_193 = sort(line10_reach_193);
line10_start_237 = sort(line10_start_237);
line10_reach_237 = sort(line10_reach_237);
% 机场线11: 55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
line11_start_55 = sort(line11_start_55);
line11_reach_55 = sort(line11_reach_55);
line11_start_58 = sort(line11_start_58);
line11_reach_58 = sort(line11_reach_58);
% 亦庄线12: 117-130
line12_start_117 = sort(line12_start_117);
line12_reach_117 = sort(line12_reach_117);
line12_start_130 = sort(line12_start_130);
line12_reach_130 = sort(line12_reach_130);
% 13号线13: 238-253
line13_start_238 = sort(line13_start_238);
line13_reach_238 = sort(line13_reach_238);
line13_start_253 = sort(line13_start_253);
line13_reach_253 = sort(line13_reach_253);
% 14号线14: 261-267,288-299
% 六号线有两段
line14_start_261 = sort(line14_start_261);
line14_reach_261 = sort(line14_reach_261);
line14_start_267 = sort(line14_start_267);
line14_reach_267 = sort(line14_reach_267);
line14_start_288 = sort(line14_start_288);
line14_reach_288 = sort(line14_reach_288);
line14_start_299 = sort(line14_start_299);
line14_reach_299 = sort(line14_reach_299);
% 15号线15: 268-287
line6_start_300 = sort(line6_start_300);
line6_reach_300 = sort(line6_reach_300);
line6_start_307 = sort(line6_start_307);
line6_reach_307 = sort(line6_reach_307);
% 昌平线18: 254-260
```

```
line18_start_254 = sort(line18_start_254);
line18_reach_254 = sort(line18_reach_254);
line18_start_260 = sort(line18_start_260);
line18_reach_260 = sort(line18_reach_260);
% 房山线19: 182-192
line19_start_182 = sort(line19_start_182);
line19_reach_182 = sort(line19_reach_182);
line19_start_192 = sort(line19_start_192);
line19_reach_192 = sort(line19_reach_192);
%% 保存很多个一维数组
% 一号线1: 1-23
xlswrite('line1_start_1.xlsx', line1_start_1);
xlswrite('line1_reach_1.xlsx', line1_reach_1);
xlswrite('line1_start_23.xlsx', line1_start_23);
xlswrite('line1_reach_23.xlsx', line1_reach_23);
% 二号线2: 37-54
xlswrite('line2_start_37.xlsx', line2_start_37);
xlswrite('line2_reach_37.xlsx', line2_reach_37);
xlswrite('line2_start_54.xlsx', line2_start_54);
xlswrite('line2_reach_54.xlsx', line2_reach_54);
% 八通线3: 24-36
xlswrite('line3_start_24.xlsx', line3_start_24);
xlswrite('line3_reach_24.xlsx', line3_reach_24);
xlswrite('line3_start_36.xlsx', line3_start_36);
xlswrite('line3_reach_36.xlsx', line3_reach_36);
% 四号线4: 59-93
xlswrite('line4_start_59.xlsx', line4_start_59);
xlswrite('line4_reach_59.xlsx', line4_reach_59);
xlswrite('line4_start_93.xlsx', line4_start_93);
xlswrite('line4_reach_93.xlsx', line4_reach_93);
% 五号线5: 94-116
xlswrite('line5_start_94.xlsx', line5_start_94);
xlswrite('line5_reach_94.xlsx', line5_reach_94);
xlswrite('line5_start_116.xlsx', line5_start_116);
xlswrite('line5_reach_116.xlsx', line5_reach_116);
% 六号线6: 131-150,300-307
% 六号线有两段
xlswrite('line6_start_131.xlsx', line6_start_131);
xlswrite('line6_reach_131.xlsx', line6_reach_131);
```

```
xlswrite('line6_start_150.xlsx', line6_start_150);
xlswrite('line6_reach_150.xlsx', line6_reach_150);
xlswrite('line6_start_300.xlsx', line6_start_300);
xlswrite('line6_reach_300.xlsx', line6_reach_300);
xlswrite('line6_start_307.xlsx', line6_start_307);
xlswrite('line6_reach_307.xlsx', line6_reach_307);
% 七号线7：308-328
xlswrite('line7_start_308.xlsx', line7_start_308);
xlswrite('line7_reach_308.xlsx', line7_reach_308);
xlswrite('line7_start_328.xlsx', line7_start_328);
xlswrite('line7_reach_328.xlsx', line7_reach_328);
% 八号线8：151-169
xlswrite('line8_start_151.xlsx', line8_start_151);
xlswrite('line8_reach_151.xlsx', line8_reach_151);
xlswrite('line8_start_169.xlsx', line8_start_169);
xlswrite('line8_reach_169.xlsx', line8_reach_169);
% 九号线9：170-181
xlswrite('line9_start_170.xlsx', line9_start_170);
xlswrite('line9_reach_170.xlsx', line9_reach_170);
xlswrite('line9_start_181.xlsx', line9_start_181);
xlswrite('line9_reach_181.xlsx', line9_reach_181);
% 十号线10：193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
xlswrite('line10_start_193.xlsx', line10_start_193);
xlswrite('line10_reach_193.xlsx', line10_reach_193);
xlswrite('line10_start_237.xlsx', line10_start_237);
xlswrite('line10_reach_237.xlsx', line10_reach_237);
% 机场线11：55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
xlswrite('line11_start_55.xlsx', line11_start_55);
xlswrite('line11_reach_55.xlsx', line11_reach_55);
xlswrite('line11_start_58.xlsx', line11_start_58);
xlswrite('line11_reach_58.xlsx', line11_reach_58);
% 亦庄线12：117-130
xlswrite('line12_start_117.xlsx', line12_start_117);
xlswrite('line12_reach_117.xlsx', line12_reach_117);
xlswrite('line12_start_130.xlsx', line12_start_130);
xlswrite('line12_reach_130.xlsx', line12_reach_130);
% 13号线13：238-253
```

```
xlswrite('line13_start_238.xlsx', line13_start_238);
xlswrite('line13_reach_238.xlsx', line13_reach_238);
xlswrite('line13_start_253.xlsx', line13_start_253);
xlswrite('line13_reach_253.xlsx', line13_reach_253);
% 14号线14: 261-267,288-299
% 六号线有两段
xlswrite('line14_start_261.xlsx', line14_start_261);
xlswrite('line14_reach_261.xlsx', line14_reach_261);
xlswrite('line14_start_267.xlsx', line14_start_267);
xlswrite('line14_reach_267.xlsx', line14_reach_267);
xlswrite('line14_start_288.xlsx', line14_start_288);
xlswrite('line14_reach_288.xlsx', line14_reach_288);
xlswrite('line14_start_299.xlsx', line14_start_299);
xlswrite('line14_reach_299.xlsx', line14_reach_299);
% 15号线15: 268-287
xlswrite('line15_start_268.xlsx', line15_start_268);
xlswrite('line15_reach_268.xlsx', line15_reach_268);
xlswrite('line15_start_287.xlsx', line15_start_287);
xlswrite('line15_reach_287.xlsx', line15_reach_287);
% 昌平线18: 254-260
xlswrite('line18_start_254.xlsx', line18_start_254);
xlswrite('line18_reach_254.xlsx', line18_reach_254);
xlswrite('line18_start_260.xlsx', line18_start_260);
xlswrite('line18_reach_260.xlsx', line18_reach_260);
% 房山线19: 182-192
xlswrite('line19_start_182.xlsx', line19_start_182);
xlswrite('line19_reach_182.xlsx', line19_reach_182);
xlswrite('line19_start_192.xlsx', line19_start_192);
xlswrite('line19_reach_192.xlsx', line19_reach_192);
%% 再给这个一维数组做个简单的数据处理。
```

3. a5.m

```
%% 一些简单的数据处理
%% 寻找每条地铁线路的最早和最晚时间
P = []
% 一号线1: 1-23
[EARLY1,LATE1] = REQUIRE('line1_start_1.xlsx');
[EARLY2,LATE2] = REQUIRE('line1_reach_1.xlsx');
```

```
[EARLY3,LATE3] = REQUIRE('line1_start_23.xlsx');
[EARLY4,LATE4] = REQUIRE('line1_reach_23.xlsx');
P = vertcat(P,[1,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 二号线2: 37-54
[EARLY1,LATE1] = REQUIRE('line2_start_37.xlsx');
[EARLY2,LATE2] = REQUIRE('line2_reach_37.xlsx');
[EARLY3,LATE3] = REQUIRE('line2_start_54.xlsx');
[EARLY4,LATE4] = REQUIRE('line2_reach_54.xlsx');
P = vertcat(P,[2,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 八通线3: 24-36
[EARLY1,LATE1] = REQUIRE('line3_start_24.xlsx');
[EARLY2,LATE2] = REQUIRE('line3_reach_24.xlsx');
[EARLY3,LATE3] = REQUIRE('line3_start_36.xlsx');
[EARLY4,LATE4] = REQUIRE('line3_reach_36.xlsx');
P = vertcat(P,[3,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 四号线4: 59-93
[EARLY1,LATE1] = REQUIRE('line4_start_59.xlsx');
[EARLY2,LATE2] = REQUIRE('line4_reach_59.xlsx');
[EARLY3,LATE3] = REQUIRE('line4_start_93.xlsx');
[EARLY4,LATE4] = REQUIRE('line4_reach_93.xlsx');
P = vertcat(P,[4,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 五号线5: 94-116
[EARLY1,LATE1] = REQUIRE('line5_start_94.xlsx');
[EARLY2,LATE2] = REQUIRE('line5_reach_94.xlsx');
[EARLY3,LATE3] = REQUIRE('line5_start_116.xlsx');
[EARLY4,LATE4] = REQUIRE('line5_reach_116.xlsx');
P = vertcat(P,[5,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 六号线6: 131-150,300-307
% 六号线有两段
[EARLY1,LATE1] = REQUIRE('line6_start_131.xlsx');
[EARLY2,LATE2] = REQUIRE('line6_reach_131.xlsx');
[EARLY3,LATE3] = REQUIRE('line6_start_150.xlsx');
[EARLY4,LATE4] = REQUIRE('line6_reach_150.xlsx');
P = vertcat(P,[6,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
[EARLY1,LATE1] = REQUIRE('line6_start_300.xlsx');
[EARLY2,LATE2] = REQUIRE('line6_reach_300.xlsx');
[EARLY3,LATE3] = REQUIRE('line6_start_307.xlsx');
[EARLY4,LATE4] = REQUIRE('line6_reach_307.xlsx');
P = vertcat(P,[6,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
```

```
% 七号线7：308-328
[EARLY1,LATE1] = REQUIRE('line7_start_308.xlsx');
[EARLY2,LATE2] = REQUIRE('line7_reach_308.xlsx');
[EARLY3,LATE3] = REQUIRE('line7_start_328.xlsx');
[EARLY4,LATE4] = REQUIRE('line7_reach_328.xlsx');
P = vertcat(P,[7,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 八号线8：151-169
[EARLY1,LATE1] = REQUIRE('line8_start_151.xlsx');
[EARLY2,LATE2] = REQUIRE('line8_reach_151.xlsx');
[EARLY3,LATE3] = REQUIRE('line8_start_169.xlsx');
[EARLY4,LATE4] = REQUIRE('line8_reach_169.xlsx');
P = vertcat(P,[8,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 九号线9：170-181
[EARLY1,LATE1] = REQUIRE('line9_start_170.xlsx');
[EARLY2,LATE2] = REQUIRE('line9_reach_170.xlsx');
[EARLY3,LATE3] = REQUIRE('line9_start_181.xlsx');
[EARLY4,LATE4] = REQUIRE('line9_reach_181.xlsx');
P = vertcat(P,[9,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 十号线10：193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
[EARLY1,LATE1] = REQUIRE('line10_start_193.xlsx');
[EARLY2,LATE2] = REQUIRE('line10_reach_193.xlsx');
[EARLY3,LATE3] = REQUIRE('line10_start_237.xlsx');
[EARLY4,LATE4] = REQUIRE('line10_reach_237.xlsx');
P = vertcat(P,[10,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 机场线11：55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
[EARLY1,LATE1] = REQUIRE('line11_start_55.xlsx');
[EARLY2,LATE2] = REQUIRE('line11_reach_55.xlsx');
[EARLY3,LATE3] = REQUIRE('line11_start_58.xlsx');
[EARLY4,LATE4] = REQUIRE('line11_reach_58.xlsx');
P = vertcat(P,[11,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 亦庄线12：117-130
[EARLY1,LATE1] = REQUIRE('line12_start_117.xlsx');
[EARLY2,LATE2] = REQUIRE('line12_reach_117.xlsx');
[EARLY3,LATE3] = REQUIRE('line12_start_130.xlsx');
[EARLY4,LATE4] = REQUIRE('line12_reach_130.xlsx');
P = vertcat(P,[12,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 13号线13：238-253
```

```
[EARLY1,LATE1] = REQUIRE('line13_start_238.xlsx');
[EARLY2,LATE2] = REQUIRE('line13_reach_238.xlsx');
[EARLY3,LATE3] = REQUIRE('line13_start_253.xlsx');
[EARLY4,LATE4] = REQUIRE('line13_reach_253.xlsx');
P = vertcat(P,[13,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 14号线14: 261-267,288-299
% 六号线有两段
[EARLY1,LATE1] = REQUIRE('line14_start_261.xlsx');
[EARLY2,LATE2] = REQUIRE('line14_reach_261.xlsx');
[EARLY3,LATE3] = REQUIRE('line14_start_267.xlsx');
[EARLY4,LATE4] = REQUIRE('line14_reach_267.xlsx');
P = vertcat(P,[14,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
[EARLY1,LATE1] = REQUIRE('line14_start_288.xlsx');
[EARLY2,LATE2] = REQUIRE('line14_reach_288.xlsx');
[EARLY3,LATE3] = REQUIRE('line14_start_299.xlsx');
[EARLY4,LATE4] = REQUIRE('line14_reach_299.xlsx');
P = vertcat(P,[14,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 15号线15: 268-287
[EARLY1,LATE1] = REQUIRE('line15_start_268.xlsx');
[EARLY2,LATE2] = REQUIRE('line15_reach_268.xlsx');
[EARLY3,LATE3] = REQUIRE('line15_start_287.xlsx');
[EARLY4,LATE4] = REQUIRE('line15_reach_287.xlsx');
P = vertcat(P,[15,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 昌平线18: 254-260
[EARLY1,LATE1] = REQUIRE('line18_start_254.xlsx');
[EARLY2,LATE2] = REQUIRE('line18_reach_254.xlsx');
[EARLY3,LATE3] = REQUIRE('line18_start_260.xlsx');
[EARLY4,LATE4] = REQUIRE('line18_reach_260.xlsx');
P = vertcat(P,[18,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
% 房山线19: 182-192
[EARLY1,LATE1] = REQUIRE('line19_start_182.xlsx');
[EARLY2,LATE2] = REQUIRE('line19_reach_182.xlsx');
[EARLY3,LATE3] = REQUIRE('line19_start_192.xlsx');
[EARLY4,LATE4] = REQUIRE('line19_reach_192.xlsx');
P = vertcat(P,[19,EARLY1,LATE1,EARLY2,LATE2,EARLY3,LATE3,EARLY4,LATE4]);
xlswrite('INFORMATION.xlsx',P)
```

4. REQUIRE.m

```
function [EARLY,LATE] = REQUIRE(line)
    S = xlsread(line);
    EARLY = S(1);
    LATE = S(length(S));
end
```

5. STATISTICS.m

```
function X = STATISTICS(line)
    S = xlsread(line);
    % 5点前 5点到5点半 5点半到6点 6点到6点半 6点半到7点 7点到7点半 7 点半到8点 8 点到8点半 8点半到9点
    % 9点到9点半 9点半到10点 10点到10点半 10点半到11点 11点到11点半 11 点半到12 点
    X = zeros(1,15);
    for i = 1:length(S)
        if S(i) < 5/24
            X(1) = X(1) + 1;
        elseif S(i) < 5.5/24
            X(2) = X(2) + 1;
        elseif S(i) < 6/24
            X(3) = X(3) + 1;
        elseif S(i) < 6.5/24
            X(4) = X(4) + 1;
        elseif S(i) < 7/24
            X(5) = X(5) + 1;
        elseif S(i) < 7.5/24
            X(6) = X(6) + 1;
        elseif S(i) < 8/24
            X(7) = X(7) + 1;
        elseif S(i) < 8.5/24
            X(8) = X(8) + 1;
        elseif S(i) < 9/24
            X(9) = X(9) + 1;
        elseif S(i) < 9.5/24
            X(10) = X(10) + 1;
        elseif S(i) < 10/24
            X(11) = X(11) + 1;
        elseif S(i) < 10.5/24
```

```
            X(12) = X(12) + 1;
        elseif S(i) < 11/24
            X(13) = X(13) + 1;
        elseif S(i) < 11.5/24
            X(14) = X(14) + 1;
        elseif S(i) < 12/24
            X(15) = X(15) + 1;
        end
    end
end
```

6. STATISTICS2.m

```
function [SS,Y] = STATISTICS2(line)
    S = xlsread(line);
    SS = S(1:length(S)-1)
    Y = zeros(1,length(S)-1);
    for i = 1:length(S)-1
        Y(i) = S(i+1) - S(i);
        if Y(i) < 0
            Y(i) = Y(i-1)
        end
    end
end
```

7. STATISTICS3.m

```
function [SS,Y] = STATISTICS3(line)
    S = xlsread(line);
    SS = S(1:length(S)-1)
    Y = zeros(1,length(S)-1);
    for i = 1:length(S)-1
        Y(i) = S(i+1) - S(i);
        if Y(i) < 2/1440
            SS(i) = -1
            Y(i) = -1
        end
    end
```

```
    SS(SS==-1)=[];
    Y(Y==-1)=[];
end
```

8. LINE_STATISTICS.m

```
%% 地铁班次密集情况分线图
X = [5:.5:12]
% 一号线1: 1-23
X1 = STATISTICS('line1_start_1.xlsx');
X2 = STATISTICS('line1_start_23.xlsx');
P = plot(X,X1,X,X2);
title('Line1')
legend('Start from 1','Start from 23')
saveas(gcf,'line1','png')
% 二号线2: 37-54
X1 = STATISTICS('line2_start_37.xlsx');
X2 = STATISTICS('line2_start_54.xlsx');
P = plot(X,X1,X,X2);
title('Line2')
legend('Start from 37','Start from 54')
saveas(gcf,'line2','png')
% 八通线3: 24-36
X1 = STATISTICS('line3_start_24.xlsx');
X2 = STATISTICS('line3_start_36.xlsx');
P = plot(X,X1,X,X2);
title('Line3')
legend('Start from 24','Start from 36')
saveas(gcf,'line3','png')
% 四号线4: 59-93
X1 = STATISTICS('line4_start_59.xlsx');
X2 = STATISTICS('line4_start_93.xlsx');
P = plot(X,X1,X,X2);
title('Line4')
legend('Start from 59','Start from 93')
saveas(gcf,'line4','png')
% 五号线5: 94-116
X1 = STATISTICS('line5_start_94.xlsx');
X2 = STATISTICS('line5_start_116.xlsx');
```

```
P = plot(X,X1,X,X2);
title('Line5')
legend('Start from 94','Start from 116')
saveas(gcf,'line5','png')
% 六号线6: 131-150,300-307
% 六号线有两段
X1 = STATISTICS('line6_start_131.xlsx');
X2 = STATISTICS('line6_start_150.xlsx');
P = plot(X,X1,X,X2);
title('Line6A')
legend('Start from 131','Start from 150')
saveas(gcf,'line6A','png')
X1 = STATISTICS('line6_start_300.xlsx');
X2 = STATISTICS('line6_start_307.xlsx');
P = plot(X,X1,X,X2);
title('Line6B')
legend('Start from 24','Start from 36')
saveas(gcf,'line6B','png')
% 七号线7: 308-328
X1 = STATISTICS('line7_start_308.xlsx');
X2 = STATISTICS('line7_start_328.xlsx');
P = plot(X,X1,X,X2);
title('Line7')
legend('Start from 308','Start from 328')
saveas(gcf,'line7','png')
% 八号线8: 151-169
X1 = STATISTICS('line8_start_151.xlsx');
X2 = STATISTICS('line8_start_169.xlsx');
P = plot(X,X1,X,X2);
title('Line8')
legend('Start from 151','Start from 169')
saveas(gcf,'line8','png')
% 九号线9: 170-181
X1 = STATISTICS('line9_start_170.xlsx');
X2 = STATISTICS('line9_start_181.xlsx');
P = plot(X,X1,X,X2);
title('Line9')
legend('Start from 170','Start from 181')
saveas(gcf,'line9','png')
```

```
% 十号线10: 193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
X1 = STATISTICS('line10_start_193.xlsx');
X2 = STATISTICS('line10_start_237.xlsx');
P = plot(X,X1,X,X2);
title('Line10')
legend('Start from 193','Start from 237')
saveas(gcf,'line10','png')
% 机场线11: 55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
X1 = STATISTICS('line11_start_55.xlsx');
X2 = STATISTICS('line11_start_58.xlsx');
P = plot(X,X1,X,X2);
title('Line11')
legend('Start from 55','Start from 58')
saveas(gcf,'line11','png')
% 亦庄线12: 117-130
X1 = STATISTICS('line12_start_117.xlsx');
X2 = STATISTICS('line12_start_130.xlsx');
P = plot(X,X1,X,X2);
title('Line12')
legend('Start from 117','Start from 130')
saveas(gcf,'line12','png')
% 13号线13: 238-253
X1 = STATISTICS('line13_start_238.xlsx');
X2 = STATISTICS('line13_start_253.xlsx');
P = plot(X,X1,X,X2);
title('Line13')
legend('Start from 238','Start from 253')
saveas(gcf,'line13','png')
% 14号线14: 261-267,288-299
% 六号线有两段
X1 = STATISTICS('line14_start_261.xlsx');
X2 = STATISTICS('line14_start_267.xlsx');
P = plot(X,X1,X,X2);
title('Line14A')
legend('Start from 24','Start from 36')
saveas(gcf,'line14A','png')
X1 = STATISTICS('line14_start_288.xlsx');
```

```
X2 = STATISTICS('line14_start_299.xlsx');
P = plot(X,X1,X,X2);
title('Line14B')
legend('Start from 24','Start from 36')
saveas(gcf,'line14B','png');
% 15号线15: 268-287
X1 = STATISTICS('line15_start_268.xlsx');
X2 = STATISTICS('line15_start_287.xlsx');
P = plot(X,X1,X,X2);
title('Line15')
legend('Start from 268','Start from 287')
saveas(gcf,'line15','png');
% 昌平线18: 254-260
X1 = STATISTICS('line18_start_254.xlsx');
X2 = STATISTICS('line18_start_260.xlsx');
P = plot(X,X1,X,X2);
title('Line18')
legend('Start from 254','Start from 260')
saveas(gcf,'line18','png');
% 房山线19: 182-192
X1 = STATISTICS('line19_start_182.xlsx');
X2 = STATISTICS('line19_start_192.xlsx');
P = plot(X,X1,X,X2);
title('Line19')
legend('Start from 182','Start from 192')
saveas(gcf,'line19','png');
```

9. LINE_STATISTICS_2.m

```
%% 地铁班次密集情况分线图
% 一号线1: 1-23
[SS1,Y1] = STATISTICS2('line1_start_1.xlsx');
[SS2,Y2] = STATISTICS2('line1_start_23.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line1')
legend('Start from 1','Start from 23')
saveas(gcf,'line1S','png')
% 二号线2: 37-54
[SS1,Y1] = STATISTICS2('line2_start_37.xlsx');
```

```
[SS2,Y2] = STATISTICS2('line2_start_54.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line2')
legend('Start from 37','Start from 54')
saveas(gcf,'line2S','png')
% 八通线3: 24-36
[SS1,Y1] = STATISTICS2('line3_start_24.xlsx');
[SS2,Y2] = STATISTICS2('line3_start_36.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line3')
legend('Start from 24','Start from 36')
saveas(gcf,'line3S','png')
% 四号线4: 59-93
[SS1,Y1] = STATISTICS2('line4_start_59.xlsx');
[SS2,Y2] = STATISTICS2('line4_start_93.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line4')
legend('Start from 59','Start from 93')
saveas(gcf,'line4S','png')
% 五号线5: 94-116
[SS1,Y1] = STATISTICS2('line5_start_94.xlsx');
[SS2,Y2] = STATISTICS2('line5_start_116.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line5')
legend('Start from 94','Start from 116')
saveas(gcf,'line5S','png')
% 六号线6: 131-150,300-307
% 六号线有两段
[SS1,Y1] = STATISTICS2('line6_start_131.xlsx');
[SS2,Y2] = STATISTICS2('line6_start_150.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line6A')
legend('Start from 131','Start from 150')
saveas(gcf,'line6AS','png')
[SS1,Y1] = STATISTICS2('line6_start_300.xlsx');
[SS2,Y2] = STATISTICS2('line6_start_307.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line6B')
legend('Start from 24','Start from 36')
```

```
saveas(gcf,'line6BS','png')
% 七号线7: 308-328
[SS1,Y1] = STATISTICS2('line7_start_308.xlsx');
[SS2,Y2] = STATISTICS2('line7_start_328.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line7')
legend('Start from 308','Start from 328')
saveas(gcf,'line7S','png')
% 八号线8: 151-169
[SS1,Y1] = STATISTICS2('line8_start_151.xlsx');
[SS2,Y2] = STATISTICS2('line8_start_169.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line8')
legend('Start from 151','Start from 169')
saveas(gcf,'line8S','png')
% 九号线9: 170-181
[SS1,Y1] = STATISTICS2('line9_start_170.xlsx');
[SS2,Y2] = STATISTICS2('line9_start_181.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line9')
legend('Start from 170','Start from 181')
saveas(gcf,'line9S','png')
% 十号线10: 193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
[SS1,Y1] = STATISTICS2('line10_start_193.xlsx');
[SS2,Y2] = STATISTICS2('line10_start_237.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line10')
legend('Start from 193','Start from 237')
saveas(gcf,'line10S','png')
% 机场线11: 55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
[SS1,Y1] = STATISTICS2('line11_start_55.xlsx');
[SS2,Y2] = STATISTICS2('line11_start_58.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line11')
legend('Start from 55','Start from 58')
saveas(gcf,'line11S','png')
% 亦庄线12: 117-130
```

```
[SS1,Y1] = STATISTICS2('line12_start_117.xlsx');
[SS2,Y2] = STATISTICS2('line12_start_130.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line12')
legend('Start from 117','Start from 130')
saveas(gcf,'line12S','png')
% 13号线13: 238-253
[SS1,Y1] = STATISTICS2('line13_start_238.xlsx');
[SS2,Y2] = STATISTICS2('line13_start_253.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line13')
legend('Start from 238','Start from 253')
saveas(gcf,'line13S','png')
% 14号线14: 261-267,288-299
% 六号线有两段
[SS1,Y1] = STATISTICS2('line14_start_261.xlsx');
[SS2,Y2] = STATISTICS2('line14_start_267.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line14A')
legend('Start from 24','Start from 36')
saveas(gcf,'line14AS','png')
[SS1,Y1] = STATISTICS2('line14_start_288.xlsx');
[SS2,Y2] = STATISTICS2('line14_start_299.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line14B')
legend('Start from 24','Start from 36')
saveas(gcf,'line14BS','png');
% 15号线15: 268-287
[SS1,Y1] = STATISTICS2('line15_start_268.xlsx');
[SS2,Y2] = STATISTICS2('line15_start_287.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line15')
legend('Start from 268','Start from 287')
saveas(gcf,'line15S','png');
% 昌平线18: 254-260
[SS1,Y1] = STATISTICS2('line18_start_254.xlsx');
[SS2,Y2] = STATISTICS2('line18_start_260.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line18')
```

```
legend('Start from 254','Start from 260')
saveas(gcf,'line18S','png');
% 房山线19: 182-192
[SS1,Y1] = STATISTICS2('line19_start_182.xlsx');
[SS2,Y2] = STATISTICS2('line19_start_192.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line19')
legend('Start from 182','Start from 192')
saveas(gcf,'line19S','png');
```

10. LINE_STATISTICS_3.m

```
%% 地铁班次密集情况分线图
% 一号线1: 1-23
[SS1,Y1] = STATISTICS3('line1_start_1.xlsx');
[SS2,Y2] = STATISTICS3('line1_start_23.xlsx');
xx1 = SS1(1):0.5:SS1(length(SS1));
yy1 = spline(SS1,Y1,xx1)
xx2 = SS1(1):0.5:SS2(length(SS2));
yy2 = spline(SS2,Y2,xx2)
scatter(SS1,Y1);hold on;scatter(SS2,Y2);plot(xx1,yy1,xx2,yy2);hold off;
title('Line1')
legend('Start from 1','Start from 23')
saveas(gcf,'line1SS','png')
% 二号线2: 37-54
[SS1,Y1] = STATISTICS3('line2_start_37.xlsx');
[SS2,Y2] = STATISTICS3('line2_start_54.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line2')
legend('Start from 37','Start from 54')
saveas(gcf,'line2SS','png')
% 八通线3: 24-36
[SS1,Y1] = STATISTICS3('line3_start_24.xlsx');
[SS2,Y2] = STATISTICS3('line3_start_36.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line3')
legend('Start from 24','Start from 36')
saveas(gcf,'line3SS','png')
% 四号线4: 59-93
```

```
[SS1,Y1] = STATISTICS3('line4_start_59.xlsx');
[SS2,Y2] = STATISTICS3('line4_start_93.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line4')
legend('Start from 59','Start from 93')
saveas(gcf,'line4SS','png')
% 五号线5: 94-116
[SS1,Y1] = STATISTICS3('line5_start_94.xlsx');
[SS2,Y2] = STATISTICS3('line5_start_116.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line5')
legend('Start from 94','Start from 116')
saveas(gcf,'line5SS','png')
% 六号线6: 131-150,300-307
% 六号线有两段
[SS1,Y1] = STATISTICS3('line6_start_131.xlsx');
[SS2,Y2] = STATISTICS3('line6_start_150.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line6A')
legend('Start from 131','Start from 150')
saveas(gcf,'line6ASS','png')
[SS1,Y1] = STATISTICS3('line6_start_300.xlsx');
[SS2,Y2] = STATISTICS3('line6_start_307.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line6B')
legend('Start from 24','Start from 36')
saveas(gcf,'line6BSS','png')
% 七号线7: 308-328
[SS1,Y1] = STATISTICS3('line7_start_308.xlsx');
[SS2,Y2] = STATISTICS3('line7_start_328.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line7')
legend('Start from 308','Start from 328')
saveas(gcf,'line7SS','png')
% 八号线8: 151-169
[SS1,Y1] = STATISTICS3('line8_start_151.xlsx');
[SS2,Y2] = STATISTICS3('line8_start_169.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line8')
```

```
legend('Start from 151','Start from 169')
saveas(gcf,'line8SS','png')
% 九号线9: 170-181
[SS1,Y1] = STATISTICS3('line9_start_170.xlsx');
[SS2,Y2] = STATISTICS3('line9_start_181.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line9')
legend('Start from 170','Start from 181')
saveas(gcf,'line9SS','png')
% 十号线10: 193-237
% 十号线是一个环线，但我们以193号站作为起点站，237号线作为终点站。
[SS1,Y1] = STATISTICS3('line10_start_193.xlsx');
[SS2,Y2] = STATISTICS3('line10_start_237.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line10')
legend('Start from 193','Start from 237')
saveas(gcf,'line10SS','png')
% 机场线11: 55-58
% 机场线是一个环线，但我们以55号站作为起点站，58号线作为终点站。
[SS1,Y1] = STATISTICS3('line11_start_55.xlsx');
[SS2,Y2] = STATISTICS3('line11_start_58.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line11')
legend('Start from 55','Start from 58')
saveas(gcf,'line11SS','png')
% 亦庄线12: 117-130
[SS1,Y1] = STATISTICS3('line12_start_117.xlsx');
[SS2,Y2] = STATISTICS3('line12_start_130.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line12')
legend('Start from 117','Start from 130')
saveas(gcf,'line12SS','png')
% 13号线13: 238-253
[SS1,Y1] = STATISTICS3('line13_start_238.xlsx');
[SS2,Y2] = STATISTICS3('line13_start_253.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line13')
legend('Start from 238','Start from 253')
saveas(gcf,'line13SS','png')
```

```
% 14号线14: 261-267,288-299
% 六号线有两段
[SS1,Y1] = STATISTICS3('line14_start_261.xlsx');
[SS2,Y2] = STATISTICS3('line14_start_267.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line14A')
legend('Start from 24','Start from 36')
saveas(gcf,'line14ASS','png')
[SS1,Y1] = STATISTICS3('line14_start_288.xlsx');
[SS2,Y2] = STATISTICS3('line14_start_299.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line14B')
legend('Start from 24','Start from 36')
saveas(gcf,'line14BSS','png');
% 15号线15: 268-287
[SS1,Y1] = STATISTICS3('line15_start_268.xlsx');
[SS2,Y2] = STATISTICS3('line15_start_287.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line15')
legend('Start from 268','Start from 287')
saveas(gcf,'line15SS','png');
% 昌平线18: 254-260
[SS1,Y1] = STATISTICS3('line18_start_254.xlsx');
[SS2,Y2] = STATISTICS3('line18_start_260.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line18')
legend('Start from 254','Start from 260')
saveas(gcf,'line18SS','png');
% 房山线19: 182-192
[SS1,Y1] = STATISTICS3('line19_start_182.xlsx');
[SS2,Y2] = STATISTICS3('line19_start_192.xlsx');
scatter(SS1,Y1);hold on;scatter(SS2,Y2);hold off;
title('Line19')
legend('Start from 182','Start from 192')
saveas(gcf,'line19SS','png');
```

11. batong.m

```
%真实进站时间为4:48-12: 00, 用0.2-0.5表示,共432分钟, 即共选取了432个时间
```

```
点，有432列
%每行代表一个站点，如第一行为站点25，第二行为站点26，第12行代表站点36
A=zeros(12,432);
for i=26:37
    for j=1:length(station)          %真实车站值为j+25
        if i==station(j)+1
            time=intime(j)*432./0.3-288;
            time=floor(time);
            A(i-25,time)=A(i-25,time)+1;
        end
    end
end
```

12. DISTANCE.m

```
clear all
A = xlsread('station final.xlsx');
B = zeros(328);
for i = 1:328
    for j = 1:328
        B(i,j) = 1000/9*(abs(A(i,4)-A(j,4))+abs(A(i,5)-A(j,5)));
    end
end
xlswrite('distance.xlsx',B)
```

13. LINEAR_DISTANCE.m

```
clear all
A = xlsread('station final.xlsx');
B = zeros(328);
for i = 1:328
    for j = 1:328
        B(i,j) = 1000/9*(abs(A(i,4)-A(j,4))+abs(A(i,5)-A(j,5)));
    end
end
xlswrite('distance.xlsx',B)
```

14. alpha.m

```
Bbefore9=zeros(1,328);      %前半上午每个站点的进站人数
Bafter9=zeros(1,328);       %后半上午每个站点的进站人数
BUSYb=zeros(1,328);       %9点前的拥堵程度
BUSYa=zeros(1,328);       %9点后的拥堵程度

for i=1:length(before9)
    Bbefore9(before9(i))=before9num(i);
end

for i=1:length(after9)
    Bafter9(after9(i))=after9num(i);
end

for m=1:328
    if  Bbefore9(m)>9000
        BUSYb(m)=4;
    elseif  Bbefore9(m)>6000
        BUSYb(m)=3;
    elseif  Bbefore9(m)>2000
        BUSYb(m)=2;
    else BUSYb(m)=1;
    end
end
for m=1:328
    if  Bafter9(m)>9000
        BUSYa(m)=4;
    elseif  Bafter9(m)>6000
        BUSYa(m)=3;
    elseif  Bafter9(m)>2000
        BUSYa(m)=2;
    else BUSYa(m)=1;
    end
end
BUSY=[BUSYb;BUSYa];
```

15. batonginmatrixA.m

```
%真实进站时间为7:00-12：00，用0.2917-0.5表示,共5小时，统计每十分钟进站人
数，共30 行
%每行代表一个站点，如第一行为站点25，第二行为站点26，第12行代表站点36
A=zeros(12,300);
for i=26:37                              %每分钟
    for j=1:length(stationbt)            %真实车站值为j+25
        if i==stationbt(j)+1
            time=intimebt(j)*300./0.2083-420;
            time=ceil(time);
            A(i-25,time)=A(i-25,time)+1;
        end
    end
end
B=zeros(12,30);
for i=1:12                    %间隔十分钟
    for j=1:10:300
        B(i,fix(j/10)+1)=B(i,fix(j/10)+1)+A(i,j)+A(i,j+1)+A(i,j+2)+A(i,j+3)+A(i,j+4)
    end
end
xlswrite('C:\Users\11936\Desktop\matrixA.xlsx',B);
```

16. matrixB.m

```
A=zeros(1,14);
matrixb=zeros(14,14);
for i=1:14
    A(i)=sum(matrixBv(i,:));
end
for i=1:14
    for j=1:14
        matrixb(i,j)=matrixBv(i,j)./A(i);
    end
end
xlswrite('C:\Users\11936\Desktop\matrixb.xls',matrixb);
```

C++代码

1. 还原路径.cpp

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <map>
#include <fstream>
using namespace std;
const int infinity = 5000;
map<string,int> m; //由站名到邻接矩阵中的下标映射
map<int,string> show_map; //由下标到站名的映射
map<int,string> chongfu;
class sta
{
public:
    map<string,map<string,string> > le;
};
map<string,sta> mt;
class station
{
public:
    vector<int> indx;
    vector<string> line;//所在线路
    int p; //前驱站点
    bool f; //是否被访问
    int d; //距离相隔站点数
    station(){};
    station(const vector<string> &_line, const int & _p=0, const bool & _f=false, c
    :line(_line),p(_p),f(_f),d(_d){};
};
int time2num(string a)
{
    int count = a.size();
    int js = (a[count-4]-'0'+10*(a[count-5]-'0'));
    if(count-7==0)
    {
        js += 60*a[count-7];
    }
    else
```

```
        js += 60*(a[count-7]-'0'+10*(a[count-8]-'0'));
    return js;
}
bool compa(string s1,string s2)
{
    int a = time2num(s1);
    int b = time2num(s2);
    if(b-a<5&&b>a+2)
        return true;
    return false;

}
bool compa2(string s1,string s2)
{
    int a = time2num(s1);
    int b = time2num(s2);
    if(a-b<5&&a>b+2)
        return true;
    return false;

}
string next_line(vector<string> a,vector<string> b);//是否与先前路径为同
一条路线
void set_distance(vector<vector<int> > v,int source,int end_station,string begin_ti
贪心算法求最小路径
{
    int count[13]={0};
    int *distance = new int[v.size()]{}; //distance数组
    string current_line;
    station *vertex = new station[v.size()]; //vertex数组

    for(int i=0;i<v.size();i++) //初始化vertex数组
    {
        vertex[i].f = false;
        vertex[i].p = source;
        vertex[i].d = infinity;///将所有站点的前驱结点都先设为起始点，方
便第一次选择操作，距离设为无穷大
    }
    vertex[source].f=true;  // 把起点放入vertex集合
```

```
    vertex[source].d=0;
    for(int i=0;i<v.size();i++)
    {

        distance[i]= v[source][i];  //  更新distance
    }
    int j = source; // j是distance最小的点的编号。source是为了避免出错赋
的初值
    for(int z = 1;z<v.size();z++)  //  z是计数变量，把除起点外的点放入vertex集
合
    {
        int min = infinity;
        for(int i=0;i<v.size();i++)  //  找未放入集合的点中，distance最
小的点
        {
            if(!vertex[i].f)
            {
                if(distance[i]<=min)
                {
                    min = distance[i];
                    j = i;
                }
            }
        }
        vertex[j].f = true;//标记为查找到
        vertex[j].d=min;
        current_line = next_line(sys[j].line,sys[vertex[j].p].line);
        //cout << m[j] << " "<<min<<" "<<current_line<<endl;
        if(min==infinity)   // 如果min==infinity，说明j点出度为0，可以跳
过下面的步骤
            continue;
        for(int i=0;i<v[j].size();i++)  // 更新j点的邻近点的distance
        {
            if((!vertex[i].f)&&v[j][i]!=infinity)
            {
                int wei;
                if(current_line!=next_line(sys[i].line,sys[j].line))
                    {
                        wei = v[j][i]+3;
```

```
                }
            else
                wei = v[j][i];

            if(distance[i]>min+wei)
            {
                distance[i] = min+wei;
                vertex[i].p=j;
            }
         }
    }
} //把所有点都放入了vertex集合
string entertime,linenum;
int nz = 0;
current_line = next_line(sys[end_station].line,sys[vertex[end_station].p].line)
cout<<m[end_station]<<"to" <<m[source]<<endl;
map<string,string>::iterator it = mt[m[end_station]].le[current_line].begin();
for(;it!=mt[m[end_station]].le[current_line].end();it++)
{
    if(compa(begin_time,it->first))
       {
           entertime = it->first;
           linenum = it->second;
       }
}
cout << "\ntake the train "<< linenum<<" enter time "<< entertime<<endl;

cout << sys[end_station].indx[0]<< m[end_station];
while(end_station!=source)
{
    nz++;
    cout << " "<<sys[vertex[end_station].p].indx[0]<<m[vertex[end_station].p];
    end_station = vertex[end_station].p;
    if(end_station!=source&&current_line!=next_line(sys[end_station].line,sys[ve
    {
        current_line = next_line(sys[end_station].line,sys[vertex[end_station].p
下一站非现在所在路线，则换乘
        cout <<"\nchange to "<<current_line<<" line\n";
```

```
        }
    }
    for(it = mt[m[end_station]].le[current_line].begin();it!=mt[m[end_station]].le[
    {
        if(compa2(end_time,it->first))
           {
               entertime = it->first;
               linenum = it->second;
           }
    }
    cout << "\nleave the train "<< linenum<<" leave time "<< entertime<<endl;
    cout << "total "<<nz<<endl; //输出总需乘坐站数
    cout << endl<<endl;
}
string next_line(vector<string> a,vector<string> b)///通过当前line与当前
站点与下一站点的公共line是否相同判定是否换乘，并返回换乘线
{
    for(int i=0;i<a.size();i++)
    {
        for(int j=0;j<b.size();j++)
        {
            if(a[i]==b[j])
            {
                return a[i];
            }
        }
    }
    return "";

}


void create_time_table()
{
    ifstream in;
    in.open("3.txt");
    string line;
    int stn;
    string num,begin_time,end_time;
```

```
    in>> line>>num>>stn>>begin_time>>end_time;
    cout << line<<num<< stn<<begin_time;
    while(!in.eof())
    {
        /*if(mt.find(stn)==mt.end()) // 当前m中没有这个站点
        {
            mt[stn]=new sta();//创建新的站点
        }*/
            mt[chongfu[stn]].le[line][begin_time]=num;
            in>> line>>num>>stn>>begin_time>>end_time;
    }
    cout << mt[chongfu[stn]].le[line][begin_time];
    cout << "good\n";
    in.close();
}
void create_adjacency_list(vector<vector<int> > &table,map<string,int> &m,map<int,s
{
    ifstream in;
    in.open("1.txt");
    string station_name;//站名
    string current_line="0"; //当前所在路线
    int index=0; //下标
    string r; //读入线路
    int last_index; // 记录前一站点下标
    station temp;
    int num;
    in>>num>>station_name>>r;
    while(!in.eof())
    {
        cf[num]=station_name;
        if(m.find(station_name)==m.end()) // 当前m中没有这个站点
        {
            m[station_name]=index;//创建新的站点
            show_map[index]=station_name;
            station temp;
            temp.line.push_back(r);//加入所在路线
            temp.indx.push_back(num);
            v.push_back(temp);//加入站点名
```

```
            vector<int> tv;     // tv是temp vector，用于建立邻接表，与前
一站点发生联结，主要用于解决中转站
            if(current_line==r)//如果在同一条线上，则将当前站和前一站建
立相连的边，体现在邻接表上
            {
                tv.push_back(last_index);
                table.push_back(tv);
                table[last_index].push_back(index);
            }
            else table.push_back(tv);
            last_index=index;
            index++;
        }
        else // 如果不在一条线上，则不与读取的前一站建立边
        {
            v[m[station_name]].line.push_back(r);
            if(current_line==r)
            {
                table[m[station_name]].push_back(last_index);
                table[last_index].push_back(m[station_name]);
                last_index = m[station_name];
            }
        }
        /*cout << m[station_name]<< station_name;
        for(int i=0;i<v[m[station_name]].line.size();i++)
            cout <<v[m[station_name]].line[i];
        cout <<endl;*/
        current_line = r;//调整当前路线
        in>>num>>station_name>>r;
    }
    in.close();
}

void create_adjacency_matrix(vector<vector<int> > &table,vector<vector<int> > &matri
生成的邻接矩阵做邻接表，方便贪心算法设计
{
    for(int i=0;i<table.size();i++)
    {
        for(int j=0;j<table.size();j++)
```

```
        {
            matrix[i].push_back(infinity);
        }
    }
    for(int i=0;i<table.size();i++)
    {
        for(int j=0;j<table[i].size();j++)
            matrix[i][table[i][j]]=1;     // 每条边的默认weight是1
    }
}
int main()
{
    //输入文件流
    //数据，保存格式为：站点名，路线号
    vector<station > v;//保存所有站点信息
    vector<vector<int> > table;  //邻接表
    create_adjacency_list(table,m,show_map,chongfu,v); //建立邻接表
    vector<vector<int> > matrix;  //邻接矩阵
    matrix.resize(table.size());//建立邻接矩阵
    create_time_table();
    create_adjacency_matrix(table,matrix);
    string begin,end;
    int b,e;
    string begin_time,end_time;
    cout << "\n";//输入出发点和终点
    ifstream in;
    in.open("5.txt");
    int num;
    while( in>> num>> b >> e>>begin_time>>end_time)
    {
        cout << num <<"\t";
        begin = chongfu[b];
        end = chongfu[e];
        if(m.find(begin)==m.end()||m.find(end)==m.end())
        {
            cout << "The station isn't exist. The line haven't open.\n Please input
            continue;
        }
    set_distance(matrix,m[end],m[begin],begin_time,end_time,show_map,v);
```

```
    cout << "\n";
    }
    in.close();
}
```

2. 地铁智能规划.cpp

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <map>
#include <fstream>
using namespace std;
const int infinity = 5000;
int bt[]={22,23,26,27,28,29,30,31,32,33,34,35,36};
vector<int> batong(bt,bt+13);

class station
{
public:
    vector<int> indx;
    vector<string> line;//所在线路
    int p; //前驱站点
    bool f; //是否被访问
    int d; //距离相隔站点数
    station(){};
    station(const vector<string> &_line, const int & _p=0, const bool & _f=false, co
    :line(_line),p(_p),f(_f),d(_d){};
};

string next_line(vector<string> a,vector<string> b);//是否与先前路径为同
一条路线
void set_distance(vector<vector<int> > v,int source,int end_station,map<int,string>&
贪心算法求最小路径
{
    int count[13]={0};
    int *distance = new int[v.size()]{}; //distance数组
    string current_line;
    station *vertex = new station[v.size()]; //vertex数组
```

```
    for(int i=0;i<v.size();i++) //初始化vertex数组
    {
        vertex[i].f = false;
        vertex[i].p = source;
        vertex[i].d = infinity;///将所有站点的前驱结点都先设为起始点，方
便第一次选择操作，距离设为无穷大
    }
    vertex[source].f=true;  // 把起点放入vertex集合
    vertex[source].d=0;
    for(int i=0;i<v.size();i++)
    {

        distance[i]= v[source][i];  //  更新distance
    }
    int j = source; // j是distance最小的点的编号。source是为了避免出错赋
的初值
    for(int z = 1;z<v.size();z++)  //  z是计数变量，把除起点外的点放入vertex集
合
    {
        int min = infinity;
        for(int i=0;i<v.size();i++)  //  找未放入集合的点中，distance最
小的点
        {
            if(!vertex[i].f)
            {
                if(distance[i]<=min)
                {
                    min = distance[i];
                    j = i;
                }

            }
        }
        vertex[j].f = true;//标记为查找到
        vertex[j].d=min;
        current_line = next_line(sys[j].line,sys[vertex[j].p].line);
        //cout << m[j] << " "<<min<<" "<<current_line<<endl;
        if(min==infinity)   // 如果min==infinity，说明j点出度为0，可以跳
```

过下面的步骤

```
            continue;
        for(int i=0;i<v[j].size();i++)  // 更新j点的邻近点的distance
        {
            if((!vertex[i].f)&&v[j][i]!=infinity)
            {
                int wei;
                if(current_line!=next_line(sys[i].line,sys[j].line))
                    {wei = v[j][i]+3;
                    }
                else
                    wei = v[j][i];

                if(distance[i]>min+wei)
                {
                    distance[i] = min+wei;
                    vertex[i].p=j;
                }
            }
        }
    } //把所有点都放入了vertex集合
    int nz = 0;
    current_line = next_line(sys[end_station].line,sys[vertex[end_station].p].line)
    cout << sys[end_station].indx[0]<< m[end_station];
    while(end_station!=source)
    {
        nz++;
        cout << " "<<sys[vertex[end_station].p].indx[0]<<m[vertex[end_station].p];
        end_station = vertex[end_station].p;
        if(end_station!=source&&current_line!=next_line(sys[end_station].line,sys[ve
        {
            current_line = next_line(sys[end_station].line,sys[vertex[end_station].p
```

下一站非现在所在路线，则换乘

```
            cout <<"\nchange to "<<current_line<<" line\n";

        }
    }
    cout << "\ntotal "<<nz<<endl; //输出总需乘坐站数
    cout << endl<<endl;
```

```
}
string next_line(vector<string> a,vector<string> b)///通过当前line与当前
站点与下一站点的公共line是否相同判定是否换乘，并返回换乘线
{
    for(int i=0;i<a.size();i++)
    {
        for(int j=0;j<b.size();j++)
        {
            if(a[i]==b[j])
            {
                return a[i];
            }
        }
    }
    return "";

}
void create_adjacency_list(vector<vector<int> > &table,map<string,int> &m,map<int,s
{
    ifstream in;
    in.open("1.txt");
    string station_name;//站名
    string current_line="0"; //当前所在路线
    int index=0; //下标
    string r; //读入线路
    int last_index; // 记录前一站点下标
    station temp;
    int num;
    in>>num>>station_name>>r;
    while(!in.eof())
    {
        cf[num]=station_name;
        if(m.find(station_name)==m.end()) // 当前m中没有这个站点
        {
            m[station_name]=index;//创建新的站点
            show_map[index]=station_name;
            station temp;
            temp.line.push_back(r);//加入所在路线
            temp.indx.push_back(num);
```

```
            v.push_back(temp);//加入站点名

            vector<int> tv;     // tv是temp vector，用于建立邻接表，与前
一站点发生联结，主要用于解决中转站
            if(current_line==r)//如果在同一条线上，则将当前站和前一站建
立相连的边，体现在邻接表上
            {
                tv.push_back(last_index);
                table.push_back(tv);
                table[last_index].push_back(index);
            }
            else table.push_back(tv);
            last_index=index;
            index++;
        }
        else // 如果不在一条线上，则不与读取的前一站建立边
        {
            v[m[station_name]].line.push_back(r);
            if(current_line==r)
            {
                table[m[station_name]].push_back(last_index);
                table[last_index].push_back(m[station_name]);
                last_index = m[station_name];
            }
        }
        /*cout << m[station_name]<< station_name;
        for(int i=0;i<v[m[station_name]].line.size();i++)
            cout <<v[m[station_name]].line[i];
        cout <<endl;*/
        current_line = r;//调整当前路线
        in>>num>>station_name>>r;
    }
    in.close();
}

void create_adjacency_matrix(vector<vector<int> > &table,vector<vector<int> > &matri
生成的邻接矩阵做邻接表，方便贪心算法设计
{
    for(int i=0;i<table.size();i++)
```

```
    {
        for(int j=0;j<table.size();j++)
        {
            matrix[i].push_back(infinity);
        }
    }
    for(int i=0;i<table.size();i++)
    {
        for(int j=0;j<table[i].size();j++)
            matrix[i][table[i][j]]=1;     // 每条边的默认weight是1
    }
}
int main()
{
    map<string,int> m; //由站名到邻接矩阵中的下标映射
    map<int,string> show_map; //由下标到站名的映射
    map<int,string> chongfu;
    //输入文件流
    //数据, 保存格式为: 站点名, 路线号
    vector<station > v;//保存所有站点信息
    vector<vector<int> > table;  //邻接表
    create_adjacency_list(table,m,show_map,chongfu,v); //建立邻接表
    vector<vector<int> > matrix;  //邻接矩阵
    matrix.resize(table.size());//建立邻接矩阵
    create_adjacency_matrix(table,matrix);
    string begin,end;
    int b,e;
    cout << "Please input the begining and the end\n";//输入出发点和终点
    while(cin >> begin >> end)
    {
        if(m.find(begin)==m.end()||m.find(end)==m.end())
        {
            cout << "The station isn't exist. The line haven't open.\n Please input
            continue;
        }
    set_distance(matrix,m[end],m[begin],show_map,v);
    cout << "Please input the begining and the end\n";
    }
}
```

3. 更智能地还原路径.cpp

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <map>
#include <fstream>
using namespace std;
const int infinity = 5000;
map<string,int> m; //由站名到邻接矩阵中的下标映射
map<int,string> show_map; //由下标到站名的映射
map<int,string> chongfu;
class sta
{
public:
    map<string,map<string,string> > le;
};
map<string,sta> mt;
class station
{
public:
    vector<int> indx;
    vector<string> line;//所在线路
    int p; //前驱站点
    bool f; //是否被访问
    int d; //距离相隔站点数
    station(){};
    station(const vector<string> &_line, const int & _p=0, const bool & _f=false, c
    :line(_line),p(_p),f(_f),d(_d){};
};
int time2num(string a)
{
    int count = a.size();
    int js = (a[count-4]-'0'+10*(a[count-5]-'0'));
    if(count-7==0)
    {
        js += 60*a[count-7];
    }
```

```
    else
        js += 60*(a[count-7]-'0'+10*(a[count-8]-'0'));
    return js;
}
bool compa(string s1,string s2)
{
    int a = time2num(s1);
    int b = time2num(s2);
    if(b-a<5&&b>a+2)
        return true;
    return false;

}
bool compa2(string s1,string s2)
{
    int a = time2num(s1);
    int b = time2num(s2);
    if(a-b<5&&a>b+2)
        return true;
    return false;

}
string next_line(vector<string> a,vector<string> b);//是否与先前路径为同
一条路线
void set_distance(vector<vector<int> > v,int source,int end_station,string begin_ti
贪心算法求最小路径
{
    int count[13]={0};
    int *distance = new int[v.size()]{}; //distance数组
    string current_line;
    station *vertex = new station[v.size()]; //vertex数组

    for(int i=0;i<v.size();i++) //初始化vertex数组
    {
        vertex[i].f = false;
        vertex[i].p = source;
        vertex[i].d = infinity;///将所有站点的前驱结点都先设为起始点，方
便第一次选择操作，距离设为无穷大
    }
```

```
    vertex[source].f=true;  // 把起点放入vertex集合
    vertex[source].d=0;
    for(int i=0;i<v.size();i++)
    {

        distance[i]= v[source][i];  //  更新distance
    }
    int j = source; // j是distance最小的点的编号。source是为了避免出错赋
的初值
    for(int z = 1;z<v.size();z++)  //  z是计数变量，把除起点外的点放入vertex集
合
    {
        int min = infinity;
        for(int i=0;i<v.size();i++)  //  找未放入集合的点中，distance最
小的点
        {
            if(!vertex[i].f)
            {
                if(distance[i]<=min)
                {
                    min = distance[i];
                    j = i;
                }
            }
        }
        vertex[j].f = true;//标记为查找到
        vertex[j].d=min;
        current_line = next_line(sys[j].line,sys[vertex[j].p].line);
        //cout << m[j] << " "<<min<<" "<<current_line<<endl;
        if(min==infinity)   // 如果min==infinity，说明j点出度为0，可以跳
过下面的步骤
            continue;
        for(int i=0;i<v[j].size();i++)  // 更新j点的邻近点的distance
        {
            if((!vertex[i].f)&&v[j][i]!=infinity)
            {
                int wei;
                if(current_line!=next_line(sys[i].line,sys[j].line))
                    {
```

```
                    wei = v[j][i]+3;
                }
            else
                wei = v[j][i];

            if(distance[i]>min+wei)
            {
                distance[i] = min+wei;
                vertex[i].p=j;
            }
         }
    }
} //把所有点都放入了vertex集合
string entertime,linenum;
int nz = 0;
current_line = next_line(sys[end_station].line,sys[vertex[end_station].p].line)
cout<<m[end_station]<<"to" <<m[source]<<endl;
map<string,string>::iterator it = mt[m[end_station]].le[current_line].begin();
for(;it!=mt[m[end_station]].le[current_line].end();it++)
{
    if(compa(begin_time,it->first))
       {
           entertime = it->first;
           linenum = it->second;
       }
}
cout << "\ntake the train "<< linenum<<" enter time "<< entertime<<endl;

cout << sys[end_station].indx[0]<< m[end_station];
while(end_station!=source)
{
    nz++;
    cout << " "<<sys[vertex[end_station].p].indx[0]<<m[vertex[end_station].p];
    end_station = vertex[end_station].p;
    if(end_station!=source&&current_line!=next_line(sys[end_station].line,sys[ve
    {
        current_line = next_line(sys[end_station].line,sys[vertex[end_station].p
下一站非现在所在路线，则换乘
        cout <<"\nchange to "<<current_line<<" line\n";
```

```
        }
    }
    for(it = mt[m[end_station]].le[current_line].begin();it!=mt[m[end_station]].le[
    {
        if(compa2(end_time,it->first))
           {
               entertime = it->first;
               linenum = it->second;
           }
    }
    cout << "\nleave the train "<< linenum<<" leave time "<< entertime<<endl;
    cout << "total "<<nz<<endl; //输出总需乘坐站数
    cout << endl<<endl;
}
string next_line(vector<string> a,vector<string> b)///通过当前line与当前
站点与下一站点的公共line是否相同判定是否换乘，并返回换乘线
{
    for(int i=0;i<a.size();i++)
    {
        for(int j=0;j<b.size();j++)
        {
            if(a[i]==b[j])
            {
                return a[i];
            }
        }
    }
    return "";

}


void create_time_table()
{
    ifstream in;
    in.open("3.txt");
    string line;
    int stn;
```

```
    string num,begin_time,end_time;
    in>> line>>num>>stn>>begin_time>>end_time;
    while(!in.eof())
    {
        /*if(mt.find(stn)==mt.end()) // 当前m中没有这个站点
        {
            mt[stn]=new sta();//创建新的站点
        }*/
            mt[chongfu[stn]].le[line][begin_time]=num;
            in>> line>>num>>stn>>begin_time>>end_time;
    }
    in.close();
}
void create_adjacency_list(vector<vector<int> > &table,map<string,int> &m,map<int,s
{
    ifstream in;
    in.open("1.txt");
    string station_name;//站名
    string current_line="0"; //当前所在路线
    int index=0; //下标
    string r; //读入线路
    int last_index; // 记录前一站点下标
    station temp;
    int num;
    in>>num>>station_name>>r;
    while(!in.eof())
    {
        cf[num]=station_name;
        if(m.find(station_name)==m.end()) // 当前m中没有这个站点
        {
            m[station_name]=index;//创建新的站点
            show_map[index]=station_name;
            station temp;
            temp.line.push_back(r);//加入所在路线
            temp.indx.push_back(num);
            v.push_back(temp);//加入站点名

            vector<int> tv;     // tv是temp vector，用于建立邻接表，与前
一站点发生联结，主要用于解决中转站
```

```
            if(current_line==r)//如果在同一条线上，则将当前站和前一站建
立相连的边，体现在邻接表上
            {
                tv.push_back(last_index);
                table.push_back(tv);
                table[last_index].push_back(index);
            }
            else table.push_back(tv);
            last_index=index;
            index++;
        }
        else // 如果不在一条线上，则不与读取的前一站建立边
        {
            v[m[station_name]].line.push_back(r);
            if(current_line==r)
            {
                table[m[station_name]].push_back(last_index);
                table[last_index].push_back(m[station_name]);
                last_index = m[station_name];
            }
        }
        /*cout << m[station_name]<< station_name;
        for(int i=0;i<v[m[station_name]].line.size();i++)
            cout <<v[m[station_name]].line[i];
        cout <<endl;*/
        current_line = r;//调整当前路线
        in>>num>>station_name>>r;
    }
    in.close();
}

void create_adjacency_matrix(vector<vector<int> > &table,vector<vector<int> > &matri
生成的邻接矩阵做邻接表，方便贪心算法设计
{
    for(int i=0;i<table.size();i++)
    {
        for(int j=0;j<table.size();j++)
        {
            matrix[i].push_back(infinity);
```

```
        }
    }
    for(int i=0;i<table.size();i++)
    {
        for(int j=0;j<table[i].size();j++)
            matrix[i][table[i][j]]=1;     // 每条边的默认weight是1
    }
}
int main()
{
    //输入文件流
    //数据，保存格式为：站点名，路线号
    vector<station > v;//保存所有站点信息
    vector<vector<int> > table;  //邻接表
    create_adjacency_list(table,m,show_map,chongfu,v); //建立邻接表
    vector<vector<int> > matrix;  //邻接矩阵
    matrix.resize(table.size());//建立邻接矩阵
    create_time_table();
    create_adjacency_matrix(table,matrix);
    string begin,end;
    int b,e;
    string begin_time,end_time;
    cout << "\n";//输入出发点和终点
    ifstream in;
    in.open("5.txt");
    int num;
    while( in>> num>> b >> e>>begin_time>>end_time)
    {
        cout << num <<"\t";
        begin = chongfu[b];
        end = chongfu[e];
        if(m.find(begin)==m.end()||m.find(end)==m.end())
        {
            cout << "The station isn't exist. The line haven't open.\n Please input
            continue;
        }
    set_distance(matrix,m[end],m[begin],begin_time,end_time,show_map,v);
    cout << "\n";
    }
```

```
        getchar();
        in.close();
    }
```