队伍编号	906020
题号	A

基于数据驱动的城市轨道交通网络优化研究

摘 要

随着我国城市轨道交通建设的快速发展，轨道交通信息化水平的不断提高，轨道交通各系统获取的数据量飞速增长且规模日渐庞大，如何基于海量数据进行分析并发挥这些海量数据价值逐渐成为研究特点。

本文主要研究数据驱动下的城市轨道交通网络优化策略，研究北京市乘客出行特征，还原乘客准确出行信息以及乘客最优路径选择，并制定乘客限流方案和限流措施，进而提高城市轨道交通线路规划、运输组织协调，改进城市轨道交通的服务水平。

问题一中：根据北京市某时段部分城市轨道交通线网数据，首先采用SPSS软件对数据进行统计分析，并对异常数据进行处理，其次对乘客出行特征（客流时间分布、客流空间分布和客流时空分布）进行分析，从出行时段分布、出行时长分布，线路客流分布、车站客流的分布、车站客流乘降差分布，不同时间段各线路的进出站客流分布不同角度研究，并从中得出一系列结论，为北京市城市轨道交通规划、运输组织、限流方案等提供参考依据。

问题二中：首先基于图论的相关知识建立相邻车站走向时间的带权邻接矩阵用于还原乘客出行信息以及乘客最优路径选择，在乘客出行信息还原模型中，建立含有限制条件的 KSP 模型及算法，并以乘客#2 为例详细介绍路径还原过程，得到乘客#2 的完整出行路径为：西红门→宣武门→北京站（4 号线→2 号线），列车编号为 419093→213001。

在乘客最优路径选择中，引入路径阻抗，路径伸展系数建立改进的 Dijkstra 模型及算法，为验证该模型的准确性和合理性，验证天安门西站→关庄站的最优路径选择，求解结果与北京地铁官方网站路线方案查询结果完全吻合，最优时间计算为 43.26min，与查询时间基本一致；最后，考虑该时刻 5 号线的拥挤情况，此时最优方案共耗时 46.92min，小于考虑拥挤时原先方案所耗时的 52.03min，进一步验证该模型的的科学性和运算的准确性。

问题三中：在分析大客流条件下乘客候车延误影响因素的基础上，考虑列车周期性运行特性，并抽象乘客到达规律符合均匀分布，以各站乘客均衡候车为优化目标，构建线路层车站间的单目标多约束协同限流模型，在不限制限流车站个数下，基于 Matlab 非线性规划 fmincon 函数进行模型求解，在限制限流车站个数下，采用基于改进模拟退火算法进行模型求解结果表明，选择双桥、管庄进行限流，双桥的减少率最高，双桥、管庄进行限流时累计减少率达到 17.3%。通过与北京八通线已有的限流车站以及从计算结果的趋势性对比体现了模型的有效性和合理性。

问题四中：基于问题一客流乘客出行特征（客流时间分布、客流空间分布和客流时空分布）的分析数据以及问题三限流模型和限流方案，提出采用动态限流方案、限流车站的选取、限流时段的选择、限流强度等多种措施对城市轨道交通多站协同客流控制问题进行研究的方法。基于对北京市城市轨道交通客流限流控制策略研究，有效解决目前研究的限流方案难以根据动态客流变化做出相应的调整的问题，进一步丰富城市轨道交通客流需求管理的理论和方法。

关键词： 城市轨道交通； 数据驱动； 网络路径优化； 协同限流

目录

一、问题的提出 1

二、基本假设 1

三、符号说明 2

四、问题分析 3

4.1 问题 1 的分析 3

4.2 问题 2 的分析 3

4.3 问题 3 的分析 3

4.4 问题 4 的分析 4

五、问题 1 的模型建立与求解 4

5.1 问题分析 4

5.2 数据处理 4

5.2.1 数据统计 4

5.2.2 异常数据处理方法 4

5.2.3 剔除异常数据值 5

5.3 乘客出行特征分析 6

5.3.1 客流时间分布 6

5.3.2 客流空间分布 7

5.3.3 客流时空分布 9

六、问题 2 的模型建立与求解 10

6.1 问题分析 10

6.2 模型分析 10

6.2 模型建立 11

6.2.1 乘客出行信息准确还原模型 12

6.2.2 乘客最优路径选择模型 13

6.3 模型的求解 13

6.3.1 还原模型求解 13

6.3.2 选择模型求解 14

6.4 结果的计算与分析 14

6.4.1 邻接矩阵的建立 14

6.4.2 乘客出行信息准确还原结果 15

6.4.3 乘客最优路径选择结果 18

七、问题 3 的模型建立与求解 19
7.1 问题分析 19
7.1.1 背景分析 19
7.1.2 数据准备 20
7.2 模型建立 22
7.2.1 不限制限流车站个数模型 22
7.2.2 八通线限制限流车站个数 23
7.3 模型求解 25
7.4 结果分析与验证 26
7.4.1 不限制限流车站个数 26
7.4.2 限制限流车站个数 26
7.4.3 两个限流效果最好的车站 27
八、问题 4 的分析与求解 28
8.1 问题分析 28
8.2 具体限流措施 28
九、模型的评价与推广 29
9.1 模型的评价 29
9.2 模型的推广 29
十、参考文献 30
附录 31
附录 1：求解问题一各数据 31
附录 2：求解问题二的 MATLAB 程序 33
附录 2-1：乘客出行信息准确还原模型程序 33
附录 2-2：乘客最优路径选择模型程序 38
附录 3：求解问题三的 MATLAB 程序 39
附录 3.1: 优化模型 MATLAB 程序 39
附录 3.2：改进的模拟退火算法 MATLAB 程序 41

一、 问题的提出

截至 2018 年 12 月 31 日，中国内地累计共有 35 座城市建成并投运城市轨道交通，里程共计 5766.6 公里。进入“十三五”以来，三年累计新增运营线路长度为 2148.7 公里，年均新增线路长度为 716.2 公里（2018 中国城市轨道交通协会快报）。根据 2018 年中国内地城轨交通运营线路长度排名前 5 的各大城市。以北京市为例，其轨道交通覆盖 11 个市辖区，运营里程约 714 公里，共设车站 391 座，开通里程居中国第二位。此外，据 2017 年统计，北京城市轨道交通年乘客量全年达到 45.3 亿人次，日均客流为 1241.1 万人次，单日客运量最高达 1327.46 万人次。可见，城市轨道交通已成为大城市居民出行的主要载体，也是城市发展的重要支撑。

目前，北上广等城市轨道交通客流量大，乘客出行的 O-D 数据缤繁复杂，以北京城市轨道交通网络为例，现需要你对给定的历史数据进行以下分析和评估：

问题 1：附件 1 给出了北京市某时段部分城市轨道交通线网的乘客 O-D 数据，附件 2 为基础信息数据，附件 3 为该时段的列车运行图数据。依据北京城市轨道交通线网图（附件 4），试分析基于以上数据的乘客出行特征，包括出行时段分布、出行距离分布、出行时长分布等。

问题 2：基于问题 1 的路径选择结果，设计一套算法还原乘客出行的准确信息，即乘客在何时何站搭乘何辆地铁列车（如有换乘，需计算）并在何时何地出站，完成其一次完整的地铁出行，并完整填写表 2（计算乘客编号为 2、7、19、31、41、71、83、89、101、113、2845、124801、140610、164834、193196、223919、275403、286898、314976、315621 的完整出行线路）。另外，设计一套智能算法，以辅助并优化乘客的在轨道交通路网中的路径选择，如通过优化路径可缩短行程、减少拥挤等。

问题 3：假设地铁八通线每列列车容量为 1428 人，列车座位数为 256 座，限流时段长度可根据需要任选，且以 7：00 为首班列车发车时刻，在减小列车超载现象的基础上，尽可能缩短乘客出行时间（包括出行时间和滞留时间），并以此为目标建立城市轨道交通单一线路乘客限流模型。对模型求解后给出具体限流措施以改进八通线的服务水平，具体包括：

问题 3.1：若八通线不限制限流车站个数，试分析限流前后的总出行时间、平均出行时间对比，结果如表 3 所示。

问题 3.2：若八通线限制限流车站个数（分别取限流车站数为 1-5 个车站），试分析限流前后的总出行时间、平均出行时间对比，结果如表 4 所示。

问题 3.3：根据以上分析结果，举例说明八通线两个限流效果最好的车站，并阐述原因。

问题 4：结合问题 1-3，给出具体限流措施以改进城市轨道交通的服务水平，如动态限流方案、限流车站的选取、限流时段的选择、限流强度等。

二、基本假设

问题一假设：

（1）列车按计划时刻表运行，无晚点或其他突发事件发生；

（2）假设进站客流为同一控制时间段内均匀到达，状态稳定；

（3）假设线网乘客 O-D 数据、线网基础信息表、列车运行图数据真实有效；

（4）在总限流时段内，车站进站口及站台的客流，都符合均匀到达规律，视为均匀分布；

（5）已知高峰时段客流需求量及起讫点（下称 OD 点），考虑到高峰时段客流组成多

为通勤客流，其结构稳定，可通过历史数据分析获得较高精度的 OD 客流信息。

问题二假设：

（1）乘客可正常上车，不受下车客流的逆向影响，不故意绕路，故意不下车；

（2）进站口到达乘客仅有被限流和进入站台两种状态，不存在放弃出行需求的行为；

（3）列车发车间隔时间固定，且严格按照计划时刻表运行，在实际运营中，在不发生故障的情况下，城市轨道交通列车准点率极高，可假定为按照计划时刻表运行。

问题三假设：

（1）乘客不存在主观留乘行为，当列车能力未达饱和，乘客可继续上车时，站台乘客不会选择等候下一趟列车；

（2）仅研究单向的列车运行过程，由于城市轨道交通常态化限流时段的客流潮汐特征明显，因此本文仅研究单一方向上的协同限流策略，模型输入参数如未进行特殊说明，均为处理后的单方向输入参数；

（3）进站乘客进入站台后，仅可通过上车接受运输服务离开其出发车站，不会再次返回其出发车站进站口离站；

（4）客流需求总量保持不变，即计算客流不考虑转移到其他交通方式；

（5）简化乘客在车站付费区的走行过程。不考虑进出站通道对乘客的影响，模型主要从线路层面考虑协同限流策略，因此忽略站内基础设施的影响，即经过进站口的乘客，在限流过后，可直接到达车站站台，简化乘客站内走行时间；

（6）在实际运营中，出站客流能快速离开车站，模型忽略其对站台及进站口客流的影响。

三、符号说明

本章构建模型所使用符号，可分为四类：系统编号及时间、乘客人数、车站属性、列车属性。具体符号说明，如表 3-1 所示：

表 3-1 符号说明

符号	意义
s	为线路单方向上车站顺序编号，s=1，2，3，…，*S*
r	为线路单方向上列车顺序编号，r=1，2，3，…，*R*
t	为各限流时段顺序编号，t=1，2，3…，*TE*
Δt	为每一个限流时段持续时间，单位：*min*
R_s^{at}（t）	为 t 时刻，车站 s 的乘客到达率，单位：人/*min*
Q_s^t	为车站 *s* 在时间粒度 *T* 内的总进站量，单位：人
$P_{i,\ s}$	为从车站 *i* 到车站 *s* 的乘客占 *i* 站上车总乘客的比例
ρ_s	车站 s 总进站量中，选择本文研究方向出行乘客所占比例
γ	为进站客流量放大系数，%
f	为线路上列车发车间隔时间，单位：*min*
ΔT_s^{s+1}	为列车在车站 *s* 和 *s+1* 的区间运行时分，单位：*min*
Q_S^P	车站 *s* 的站台可容纳的最大聚集人数能力，单位：人
T_{delay}	为乘客总出行延误时间，单位：人 *min*
T_{ql}	为因进站口限流导致乘客延误时间，单位：人 *min*
T_{qr}	为因站台留乘导致乘客延误时间，单位：人 *min*

θ_1	表示因进站口限流导致乘客延误的延误时间惩罚系数
θ_2	表示因留乘导致乘客延误的延误时间惩罚系数
q_s^o（t）	为在 t 时段内，新到达车站 s 进站口等候乘车的乘客人数，单位：人
q_s^k（t）	为在 t 时段内，车站 s 进站口等候进站的聚集乘客人数，单位：人
q_s^l（t）	为在 t 时段内，车站 s 进站口被限制未能进入车站的乘客人数，单位：人
q_s^e（t）	为在 t 时段内，车站 s 进站口进入车站站台的乘客人数，单位：人 min
q_s^p（t）	为在 t 时段内，车站 s 站台的候车乘客人数，单位：人 min
q_s^a（t）	为在 t 时段内，车站 s 站台下车的乘客人数，单位：人 min
q_s^b（t）	为在 t 时段内，在车站 s 站台候车乘客中的上车人数，单位：人 min
q_s^d（t）	为在 t 时段结束时，车站 s 站台候车乘客中的留乘人数，单位：人 min
q_r^{in}（t）	为在 t 时段结束时，列车 r 内的载客人数，单位：人 min
q_r^{av}（t）	为在 t 时段结束时，列车 r 可允许的上车人数，单位：人 min
$L_{s,\,r}$（t）	为在 t 时段开始时，列车 r 是否位于车站 s 内，为 0-1 变量
C	列车额定载客人数，单位：人
η^{max}	最大列车满载率系数，%

四、 问题分析

4.1 问题 1 的分析

城市轨道交通运营客流是动态变化的，对客流调查数据进行统计分析，可以了解客流在时间、空间上的动态变化规律；同时对既有线路的运营客流特征分析，也能为后续实施线路或者其他城市的规划路网提供参考数据，从而为其线网规模的控制、基建工程和设备采用与布置以及运输组织等诸多方面提供参考。在进行乘客出行特征分析之前，需对海量数据进行统计收集，深入研究分析的各数据间的关系和含义，并对异常数据进行处理。根据轨道交通路网层次将客流特征分为车站客流特征、线路客流特征和网络客流特征三类，将北京市城市轨道交通乘客出行分别按照时间、空间、时空进行分析。

4.2 问题 2 的分析

问题 2 主要包括两个问题：第一：还原乘客一次出行的准确信息，即乘客在何时何站搭乘何辆地铁列车（如有换乘，需计算）并在何时何地出站，由于乘客进站位置，出站位置以及进站时间和和出站时间是确定的，路径还原即为在限制条件下的多解问题。第二：建立智能算法辅助并优化乘客在轨道交通路网中的路径选择，即为选择耗时时间最短路径下的最优解问题，通过选择最优路径缩短行程、提高乘客换乘效率、优化站内客流并减少拥挤。

4.3 问题 3 的分析

由于本文主要研究单线协同限流策略，重点分析线路上各车站限流人数与限流时间的关系，因此，本文的决策变量为车站限流人数，车站进站口聚集人数中，因受到限流影响而未能进入车站站台的人数。城市轨道交通运营方迫于站台能力和列车能力的限

制，当线路客流量较大时，分别在各站进站口采取一定措施，限制部分乘客的进站，从而对站内候车人数进行一定规模的控制，使整条线路的运营达到最优状态。

基于以上分析，所建模型的目标函数为最小化乘客总出行延误时间乘客总出行延误时间应为在总限流时段内全线各站乘客出行延误时间的总和，为一维数值形式。因此，可通过数学方法，制定使全线乘客总出行延误时间最小化的单线协同限流策略。

4.4 问题 4 的分析

问题 4 要求给出具体限流措施以改进城市轨道交通的服务水平，针对城市轨道交通系统具有的动态时变特点，提出了采用动态时域优化对城市轨道交通多站协同客流控制问题进行研究的方法。基于动态时域的客流控制策略，实现对限流方案的持续优化，有效解决了目前研究的限流方案难以根据动态客流变化做出相应的调整的问题。这种通过持续动态地对既有的限流方案进行动态评价、及时调整和优化思想，考虑了城市轨道交通系统的客流动态性和全局性，实现了客流不均衡条件的动态削峰及路网能力高效利用。同时，也进一步丰富了城市轨道交通客流需求管理的理论和方法[1, 2]。

五、 问题 1 的模型建立与求解

5.1 问题分析

通过附件 1 北京市某时段部分城市轨道交通线网的乘客 O-D 数据，附件 2 为基础信息数据，附件 3 为该时段的列车运行图数据。在进行乘客出行特征分析之前，需对海量数据进行统计收集，利用 SPSS 统计软件对异常数据进行处理。基于数据驱动的乘客出行特征研究，主要包括乘客出行客流时间分布、客流空间分布和客流时空分布，客流时间分布主要分析出行时段分布、出行时长分布，客流空间分布主要分析线路客流分布、车站客流的分布、车站客流差分布，客流时空分布主要分析不同时间段各线路的进出站客流分布。

5.2 数据处理

5.2.1 数据统计

运用 SPSS 统计软件对附件 2 数据分析，可知北京城市轨道交通网络共计 17 条线路，合计 328 个车站，其中有 49 个换乘站（包含连接 3 条线路的换乘站 3 个，连接 2 条线路的换乘站 46 个），对附件 1 北京市某时段部分城市轨道交通线网的乘客 O-D 数据统计分析，可知该时段共有 1048575 人乘坐地铁。

5.2.2 异常数据处理方法

在数据统计中可以发现其中含有多处异常数据，对异常数据的处理方法主要包括简单统计分析，3δ 原则，箱型图分析等方法。

（1）简单统计分析

对属性值进行一个描述性的统计，从而查看哪些值是不合理的。比如对年龄这个属性进行规约：年龄的区间在[0：200]，如果样本中的年龄值不再该区间范围内，则表示该样本的年龄属性属于异常值。

（2）3δ 原则

当数据服从正态分布：根据正态分布的定义可知，距离平均值 3δ 之外的概率为 $P(|x-\mu|>3\delta) \le 0.003$，这属于极小概率事件，在默认情况下我们可以认定，距离超过

平均值3δ的样本是不存在的。因此，当样本距离平均值大于3δ，则认定该样本为异常值。

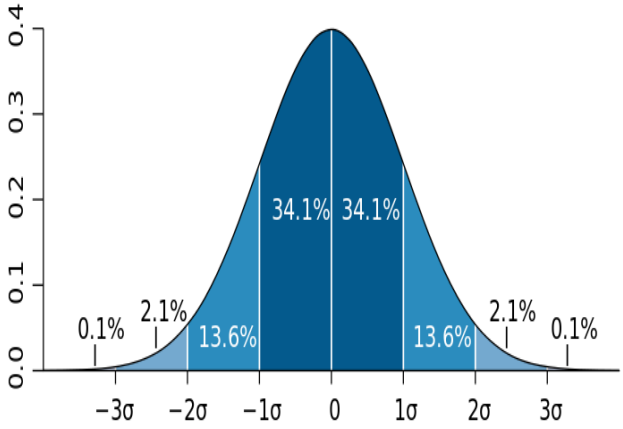


图 5-1 正态分布图

当数据不服从正态分布：当数据不服从正态分布，可以通过远离平均距离多少倍的标准差来判定，多少倍的取值需要根据经验和实际情况来决定。

（3）箱型图分析

箱型图提供了一个识别异常值的标准，即大于或小于箱型图设定的上下界的数值即为异常值，箱型图如下图所示：

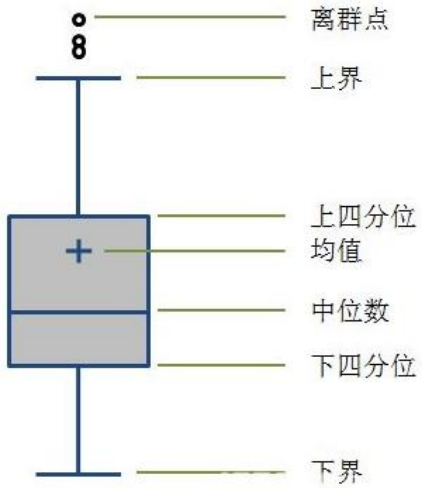


图 5-2 箱型图

上四分位我们设为 U，表示的是所有样本中只有 1/4 的数值大于 U；同理，下四分位我们设为 L，表示的是所有样本中只有 1/4 的数值小于 L。设上四分位与下四分位的插值为 IQR，即：IQR=U-L，那么，上界为 U+1.5IQR ，下界为：L - 1.5IQR。箱型图选取异常值比较客观，在识别异常值方面有一定的优越性。

5.2.3 剔除异常数据值

基于上述异常数据处理方法对数据进行分析，从中得到一定数量数据异常的乘客，下面分别举例说明不同异常原因的乘客出现信息。

（1） 乘客#357641，异常原因：出站刷卡时间小于进站刷卡时间。

表 5-1 乘客#357641 出行信息

乘客编号	始发车站	车站名称	线路编号	目的地车站	车站名称	线路编号	进站刷卡时刻	出站刷卡时刻	时间差值
357641	45	前门	2	49	朝阳门	2	8：25：00	8：19：04	**-0：05：56**

（2）乘客# 423669，异常原因：进站刷卡时间与出站刷卡时间的时间间隔过小。

表 5-2 乘客# 423669 出行信息

乘客编号	始发车站	车站名称	线路编号	目的地车站	车站名称	线路编号	进站刷卡时刻	出站刷卡时刻	时间差值
423669	55	东直门	11	167	育知路	3	10：52：00	10：52：15	**0：00：15**

（3）乘客# 686728，异常原因：进站刷卡时间与出站刷卡时间的时间间隔太大。

表 5-3 乘客# 686728 出行信息

乘客编号	始发车站	车站名称	线路编号	目的地车站	车站名称	线路编号	进站刷卡时刻	出站刷卡时刻	时间差值
686728	215	巴沟	10	65	海淀黄庄	4	2：05：00	7：37：09	**5：32：09**

（4）乘客# 267413，异常原因：始发车站为无效车站。

表 5-4 乘客# 267413 出行信息

乘客编号	始发车站	车站名称	线路编号	目的地车站	车站名称	线路编号	进站刷卡时刻	出站刷卡时刻	时间差值
267413	**4294967295**	—	—	239	大钟寺	13	7：54：00	7：54：44	——

5.3 乘客出行特征分析

5.3.1 客流时间分布

（1）出行时段分布

通过分析乘客出行时间分布，可以更好的确定城市轨道交通出入口、通道等设备容量，是计算全日行车计划和车辆配备计划的基础，出行时段分布主要包括进站出行时段分布和出站出行时段分布，通过对附件 1 数据分析可知，99.9%的乘客均是在 4：00：00-12：00：00 这个时间段进站，99.4%的乘客均是在 4：00：00-13：00：00 这个时间段进站，因此在研究出行时段分布时，以进站（出站）时刻的时间段（取 1 小时为间隔时间段）为横坐标，以相应于各时间段的进站（出站）人数为纵坐标，运用 SPSS 统计软件得到进站出行时段分布和出站出行时段分布如图 5-3 和图 5-4 所示。

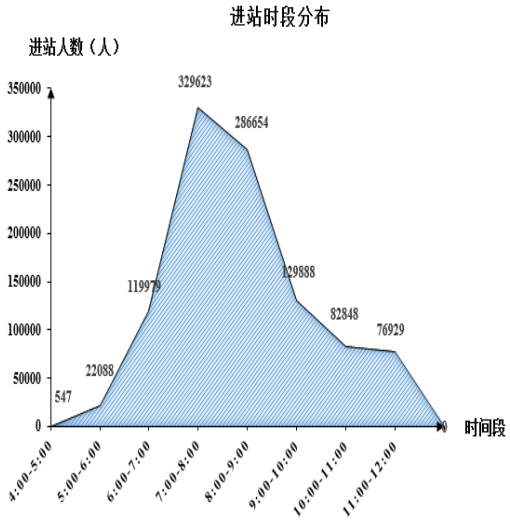


图 5-3 进站时段分布

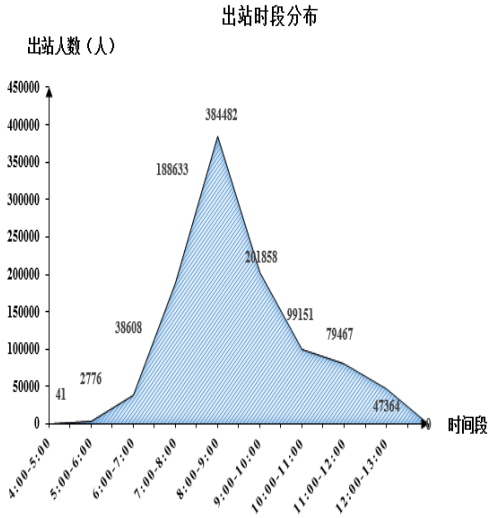


图 5-4 出站时段分布

通过上图可知，对于进站时段分布，7：00-8：00 时间段进站乘客最多，共计 329623 人，占总进站人数的 31.44%，人数次之的为 8：00-9：00 时间段，乘客进站人数占总人数的 27.34%，由于 7：00-9：00 为出行高峰，因此其进站时段分布准确合理。而对于出站时段分布，8：00-9：00 时间段出站乘客最多，共计 384482 人，占总出站人数的 36.89%。

（2）出行时长分布

出行时长分布主要研究乘客乘坐城市轨道交通所耗费时间长短，出行时长不确定性对计算拥堵成本、帮助出行者制定出行规划具有重要意义。本模型基于 SPSS 统计软件将出行时长分为 5 类情况，即耗时小于 30 分钟、在 30 分钟至 1 小时之间、在 1 小时至 1 小时 30 分钟之间、在 1 小时 30 分钟至 2 小时之间以及大于 2 小时等不同情况。同时还研究在早高峰时间段（即 7：00-9：00）乘客出行时长分布情况，所得结果如图 1-5 和图 1-6 所示。

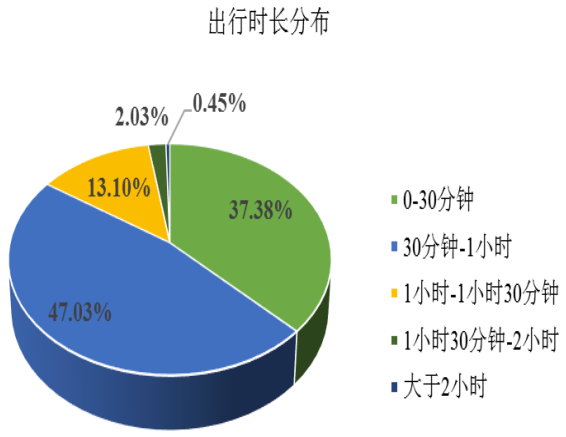


图 5-5 整体出行时长分布

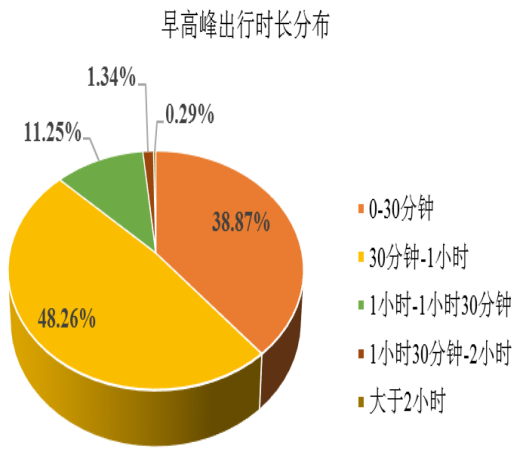


图 5-6 早高峰出行时长分布

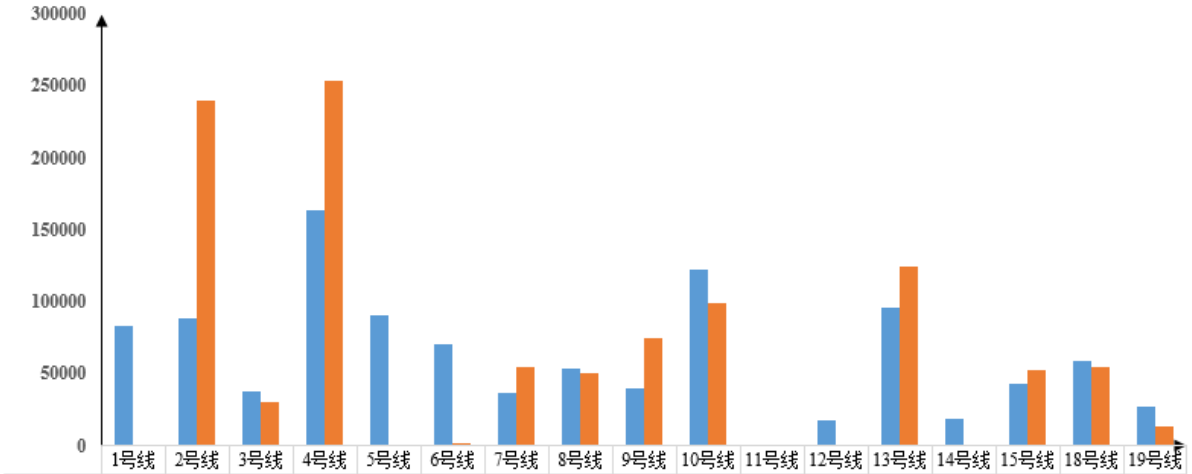
通过上图可知，整体出行时长分布与早高峰出行时长分布各耗时时间分布情况基本一致，均为出行耗时在 30 分钟至 1 小时之间所占比例最大，耗时小于 30 分钟所占比例次之，耗时大于 2 小时所占比例最小。

5.3.2 客流空间分布

（1）线路客流分布

城市轨道交通线网的各条线路因其所在的城市客流走廊带不同、沿线用地性质不

同，使得其客流规模和分布规律各异。通过客流统计数据分析北京市城市轨道交通 17 条线路的客流分布特征，依次统计 17 条线路的进站总人数和出站总人数，其线路客流分布如图 5-7 所示。



	1号线	2号线	3号线	4号线	5号线	6号线	7号线	8号线	9号线	10号线	11号线	12号线	13号线	14号线	15号线	18号线	19号线
进站人数	83451	88264	38291	163887	91156	70172	37069	53323	40193	122148	1	17182	96122	18430	42603	58696	27502
出站人数	1	239758	29888	253140	0	1420	54459	50660	74969	99588	0	0	124880	0	52335	54243	13234

图 5-7 线路客流分布

由图 5-7 可知，进站客流人数排名前 3 的线路依次为 4 号线、10 号线、13 号线，进站客流人数分别为 163887 人、122148 人、96122 人；出站客流人数排名前 3 的线路依次为 4 号线、2 号线、13 号线，出站客流人数分别为 253140 人、239758 人、124880 人。

（2）车站客流的分布

在轨道交通线路上，由于线路行经区域的用地开发性质不同，所覆盖的客流集散点的规模和数量不同，因而出现线路各个车站乘降人数不同，线路单向各个车站的客流存在不均衡现象是不可避免的。利用 SPSS 统计软件对北京城市轨道交通合计 328 个车站进出站客流进行统计，得出进站客流排名前 10 和出站客流排名前 10 的车站如图 5-8 和图 5-9 所示。

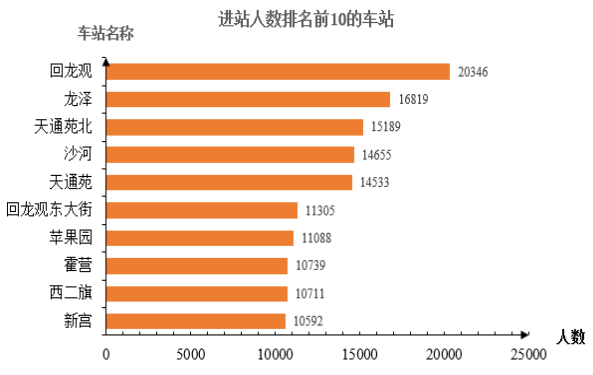


图 5-8 进站客流排名

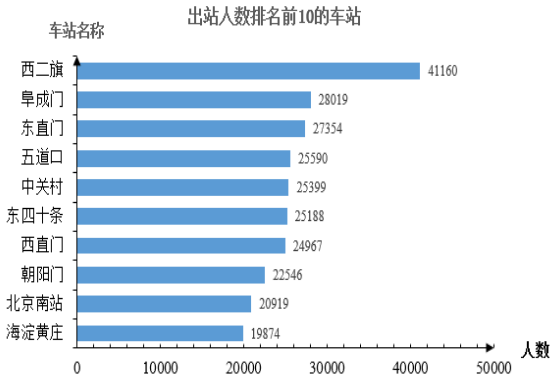


图 5-9 出站客流排名

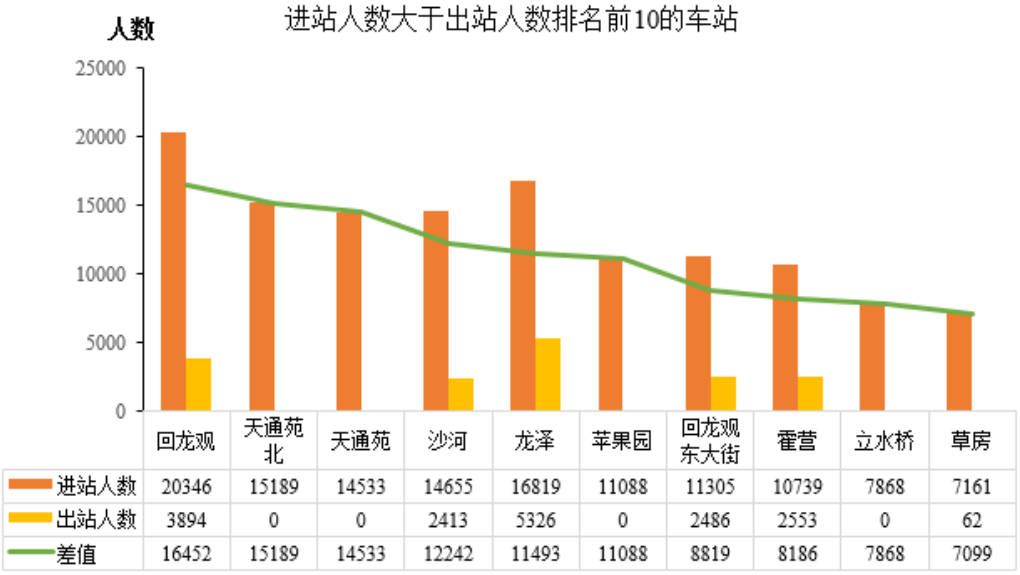
由图 5-8 可知，进站客流排名前 10 的车站依次为回龙观、龙泽、天通苑北、沙河天通苑、回龙观东大街、苹果园、霍营、西二旗、新宫，排名前 10 的车站累计进站人数均超过 1 万人次，进站客流最大的车站为回龙观车站，累计进站客流达 20346 人。

由图 5-9 可知，，出站客流排名前 10 的车站依次为西二旗、阜成门、东直门、五道口、中关村、东四十条、西直门、朝阳门、北京南站、海淀黄庄，排名前 10 的车站累计出站人数除海淀黄庄车站均超过 2 万人次，出站客流最大的车站为西二旗车站，累

计出站客流达 41160 人。

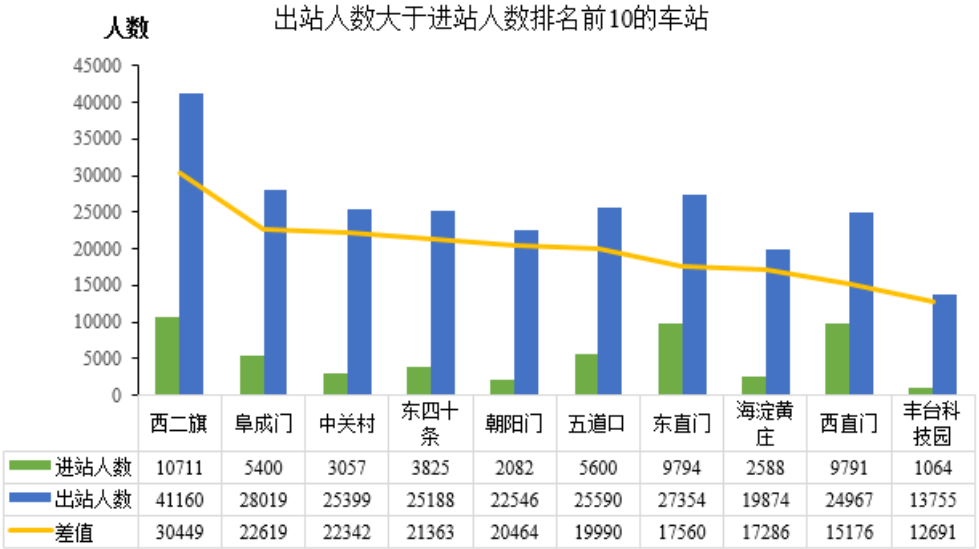
（3）车站客流差分布

轨道交通线路各个车站的进出站客流不均衡，甚至相差悬殊情况并不少见。在不少线路上，全线各站总的乘降量集中在少数几个车站上办理。此外，新的居民住宅区形成规模和新的轨道交通线路投入运营，也会使车站乘降量发生较大的变化及带来不均衡的加剧或新的不均衡。利用 SPSS 统计软件对北京城市轨道交通合计 328 个车站进出站客流进行统计，得到进站客流大于出站客流排名前 10 和出站客流大于进站客流排名前 10 的车站如图 5-10 和图 5-11 所示。



	回龙观	天通苑北	天通苑	沙河	龙泽	苹果园	回龙观东大街	霍营	立水桥	草房
进站人数	20346	15189	14533	14655	16819	11088	11305	10739	7868	7161
出站人数	3894	0	0	2413	5326	0	2486	2553	0	62
差值	16452	15189	14533	12242	11493	11088	8819	8186	7868	7099

图 5-10 进站客流大于出站客流排名前 10 的车站



	西二旗	阜成门	中关村	东四十条	朝阳门	五道口	东直门	海淀黄庄	西直门	丰台科技园
进站人数	10711	5400	3057	3825	2082	5600	9794	2588	9791	1064
出站人数	41160	28019	25399	25188	22546	25590	27354	19874	24967	13755
差值	30449	22619	22342	21363	20464	19990	17560	17286	15176	12691

图 5-11 出站客流大于进站客流排名前 10 的车站

5.3.3 客流时空分布

在 5.3.1 节和 5.3.2 节详细研究北京市城市轨道交通客流时间分布、客流空间分布的基础上，结合客流时间分布和客流空间分布情况，可进一步研究客流时空分布，同时考虑时间和空间因素，研究不同时间段各线路的进出站客流分布情况，用于指导城市轨道交通限流方案的实施，为避免大客流对线路或路网造成过大压力，控制客流分布，降低

运输压力与事故风险，从而保障轨道交通运营安全。利用 SPSS 统计软件不同时间段各线路的进出站客流分布如图 5-12 和图 5-13 所示。

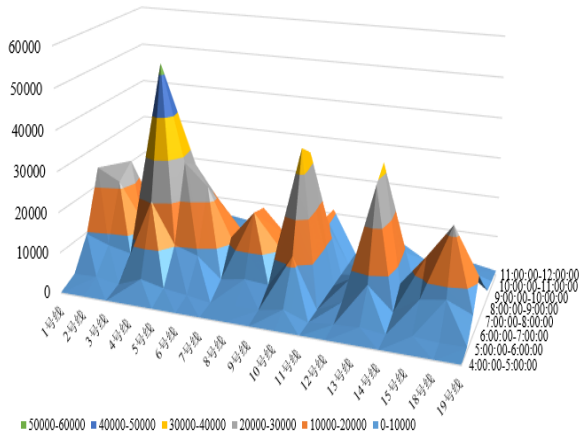


图 5-12 不同时间段各线路的进站客流分布

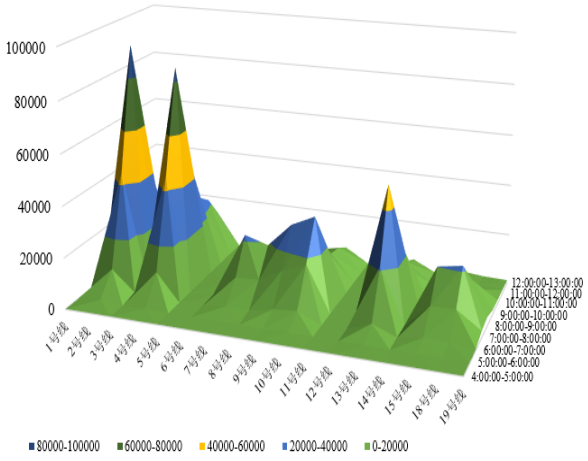


图 5-13 不同时间段各线路的出站客流分布

通过上图可知：进站高峰时间段主要集中在 7：00-9：00 时间段，线路进站最大高峰时段出现在 4 号线 7：00-8：00 这一时间段，该小时内累计进站人数达 52838 人次。出站高峰时间段主要集中在 7：00-10：00 时间段，线路出站最大高峰时段出现在 2 号线 8：00-9：00 这一时间段，该小时内累计出站人数达 92679 人次。

六、 问题 2 的模型建立与求解

6.1 问题分析

网络路径选择是指确定起点到终点的路径，路径的选择和网络的方案多种多样，且路径还原与优化常涉及大量的信息，计算量大且计算复杂，一般需借助优化算法解决该问题，如线性规划、动态规划、Dijkstra、Floyd 、A*、KSP、遗传算法、模拟退火算法等算法，这些算法大多数建立在图论的基础上寻找最优解，通过建立邻接矩阵，设置限制条件确定合理的路径选择。为有效处理轨道交通网络节点，定义有效路段和冗余路段及其判定规则。在乘客出行信息准确还原中，利用 KSP 算法确定满足限制条件的三条最优路径，在考虑时间间隔（包括进站时刻至上车时刻，下车时刻至出站时刻，换乘间隔时间）最合理的情况下选择列车编号。在乘客最优路径选择中，引入路径阻抗，路径伸展系数建立改进的 Dijkstra 算法模型。

6.2 模型分析

为分析研究城市轨道交通网络优化问题，数学上常用图来描述网络的统一工具，任何一个网络都可以看是由一些节点按某种方式连接，构成的一个网络，城市轨道交通网络也不例外。具体网络的抽象图的表示，就是用抽象的节点表示具体网络中的节点，并用节点之间的连线，或者称为边，来表示具体网络结点之间的关系。 一个具体网络可抽象为一个由点集V 和边集E 组成的图$G=(V,E)$。如果任意一点节对间的连线可双向通行，则该网络称为无向网络，否则称为有向网络。如果给每一条边都赋予相应的权值，那么网络就称为加权网络，否则称为无权网络[3]。

描述图的最直观一个方法是用图形表示，但为了计算相关统计数据的方便，往往需要将图以一定方式储存。在计算机中，常常使用矩阵来记录图，对应于不同的图，因其模型复杂度及应用具体问题不同，所以相应的图的表示方法也是多种多样的，如图的邻

接矩阵表示法、图的邻接表表示法、图的邻接多重表表示法等[4]，下面定义邻接矩阵表示法[5]：

定义：若G是一个具有n个节点的图，则G的邻接矩阵A是如下的n阶的矩阵：

$$A[i,j]=\begin{Bmatrix}1,\ \text{若}(V_i,V_j)\text{是图}G\text{的边}\\0,\ \text{若}(V_i,V_j)\text{不是图}G\text{的边}\end{Bmatrix} \tag{6-1}$$

无向图的邻接矩阵一定是对称的，而有向图的邻接矩阵一般是不对称的。

网络一般是带权的图，所以只要将邻接矩阵稍加扩充便可用来表示网络。设网络$G=(V,E,W)$具有n个节点，编号为$1,2,\cdots,n$，描述网络G的带权邻接矩阵为n阶方阵A，其矩阵元素a_{ij}，定义为：

$$a_{ij}=\begin{Bmatrix}w_{ij}，\text{若节点}\ i\ \text{到节点}\ j\ \text{有邻边，}\ w_{ij}\text{为该边边权}\\0,\ \text{若}i=j\\\infty，\text{若节点}i\text{到节点}\ j\ \text{无邻接边}\end{Bmatrix} \tag{6-2}$$

用邻接矩阵法来表示图，需要存储一个$N\times N$的空间来表示节点间的相邻关系，根据边权含义不同可分为不同的路径优化问题，当w_{ij}为距离时则为最短距离路径优化问题，而当w_{ij}为时间时则为最少时间路径优化问题，对于本模型，显然w_{ij}为时间。

目前，Dijkstra、Floyd 、A*、KSP、SPFA 算法均可用来求解网络路径优化的经典智能搜索算法。Floyd 算法可以求解网络中任意两个节点间的最短路径（或最少时间）问题，其基本的思想是借助阻抗矩阵的迭代运算来求解最小效用。相比之下，Dijkstra 算法用于求解网络中任意一个节点到其他任意节点的最短路径（或最少时间）问题，且 Dijkstra 算法在实际应用中更常用也最易实现，通过循环算法同样可以实现求解网络任意两个节点间的最短路径（或最少时间）。KSP 最短路径问题是最短路径问题的一种变形，与最短路径问题不同，KSP 问题的目的是寻找图中起点和终点间的多个备选优化路径，形成最短路径组，以最大程度满足用户对不同路径的选择需求[6]。

6.2 模型建立

（1）网络节点处理

城市轨道交通网络主要由区间和车站两个部分组成，除了轨道交通区间有运行时间阻抗，轨道交通车站也存在停车时间阻抗，因此轨道交通网络每个节点都存在节点阻抗。为了表示车站的停站时间，将每个轨道交通普通车站分拆表示为停车节点和出发节点，其间增加停车弧表示停车时间。除固定的区间运行时间、停车时间之外，换乘步行时间和换乘等待时间也是轨道交通网络阻抗的重要组成部分[7]。对于轨道交通换乘站而言，节点阻抗存在两种情况：

一方面，如果乘客在换乘站不需要换乘，则只经历停车时间，可看作普通车站，通过停车弧对其进行表示；另一方面，如果乘客在换乘站进行了换乘，则存在换乘时间，需要引入换乘弧对其进行描述。处理后的整个城市轨道交通网络可以用“带权有向图”来描述。图 6-1 和图 6-2 分别表示普通车站和换乘车站的网络节点示意图。

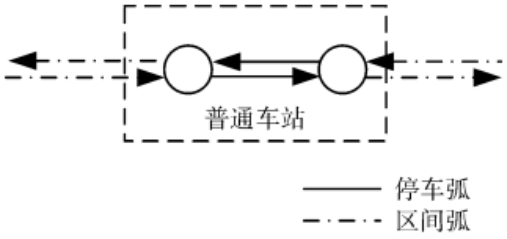


图 6-1 普通车站停车弧示意图

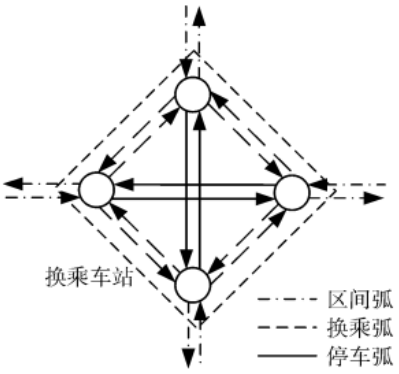


图 6-2 换乘车站换乘弧示意图

（2）路径冗余路段定义

由于城市轨道交通车站节点的特殊处理，有效路径判定规则的部分路径中还包含不符合换乘逻辑的路段。为了说明上述路段的特征，图 6-3 为某个换乘车站网络节点的示意图，虚线框线内的节点归属于同一个车站，其中节点 18、19、74、75 为换乘车站节点，其余为普通车站节点。

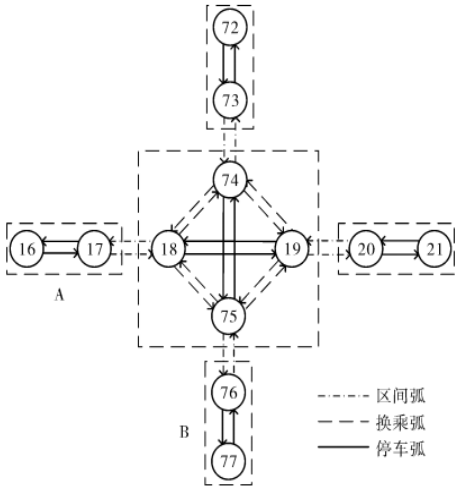


图 6-3 换乘节点网络节点示意图

假如某位乘客从车站 A 上车，在车站 B 下车，其间经过两段区间运行时间和一段换乘时间，可以看出，符合该乘客正确换乘逻辑的路径应该是 17－18－75－76。通过未改进的算法搜索到的相同起终点的路径 17－18－74－75－76、17－18－19－75－76 和 17－18－74－19－75－76。这些路径中均包含了不符合逻辑的停车时间和换乘时间，相应的 18－74－75、18－19－75、18－74－19－75 均可判定为冗余路段。

6.2.1 乘客出行信息准确还原模型

对于复杂的城市轨道交通网络，在一次出行中乘客总是希望选择出行时间最少的路径，当起讫站点间有多条路径可选时，乘客不会考虑全部的路径[1-16]，而是将其中几条作为备选方案，KSP 模型能够有效寻找前 K 条最短路径，其在有多约束限制条件下的路径问题选择中具有非常强的实用性，而乘客出行信息准确还原正是有限制条件的路径选择模型，乘客还原模型限制条件如下[6]：

① 路径 k 中不存在环路，即不重复经过同一个节点；

② 对于起点站：乘客进站时间≤列车发车时刻；对于终点站：列车到达时刻≤乘客出站时间；对于换乘站：前一列车的到达时刻≤后一列车的到达时刻；

$$s.t.\begin{cases} T_j \le T_f \\ T_d \le T_c \\ T_{qd} \le T_{hd} \end{cases} \tag{6-3}$$

③ 总时间：所经过的路径时间总和≤乘客出站时间—乘客进站时间，即

$$\sum t_{ij} \le T_{出} - T_{进} \tag{6-4}$$

式中：t_{ij} 是相邻两车站的走行时间与滞留时间之和，即等于第 j 个车站的发车时刻与第 i 个车站的到达时刻的差值。

6.2.2 乘客最优路径选择模型

对于乘客最优路径选择模型，是基于 6.2.1 模型中无起始站限制以及进出站时间限制的任意两车站最优路径的选择模型，即选择做短走行时间对应的路径模型，通过分析常见的网络路径优化经典智能搜索算法，选择 Dijkstra 算法模型，并引入路径阻抗，路径伸展系数建立改进的 Dijkstra 算法模型[13]，如式（6-5）所示。

$$\min\ t_{ij}$$

$$s.t.\begin{cases} C_k^{rs} \le (1+H_{rs}) \min\limits_i c_i^{rs} \\ T_k \le T_{\max} \end{cases} \tag{6-5}$$

$$H_{rs} = \frac{\max c_{k'}^{rs}}{\min\limits_i c_i^{rs}} - 1 \tag{6-6}$$

式中：t_{ij} 是相邻两车站的走行时间，即等于第 j 个车站的发车时刻与第 i 个车站的到达时刻的差值，H_{rs} 路径伸展系数，C_k^{rs} 为路径 k 的阻抗，$\min\limits_i c_i^{rs}$ 为最短路径阻抗，$\max c_{k'}^{rs}$ 为最大合理路径阻抗。

④ 考虑到换乘次数的增加也会降低乘客的舒适性和走行时间，因此在路径选择的判定规则中增加换乘次数的限制，即路径 k 的换乘次数小于等于网络最大换乘次数限制。

$$H_k \le H_{\max} \tag{6-7}$$

式中：H_k 为路径 k 的换乘次数；$H_{\max}$ 为网络最大换乘次数限制，本模型取 $H_{\max} = 5$。

6.3 模型的求解

6.3.1 还原模型求解

在还原模型中 KSP 搜索寻找前 K 条最少时间路径算法中，并在增加起始站限制以及进出站时间限制步骤后，乘客出现信息准确还原算法具体步骤如下：

Step1：初始化。给相关变量赋初值，输入邻接矩阵，设定参数；

Step2：在网络中应用最短路径算法获得在限制条件下的最短路径；

Step3：如果存在最短路径，那么先删除最短路径中的一条边，然后利用最短路径算法求出另外一条临时最短路径；

Step4：重复 Step3，直到最短路径中的所有边都被删除过一次，比较所有的临时最短路径，次最短路径就是最短的那一条；

Step5：如果要得到第 K 短路径，首先要将前（K-1）短路径中的所有边进行集合配对，每次删除一个边对，其余过程与 Step3 和 Step4：类似，最后将所有得到的临时最短路径进行比较，第 K 短路径就是最短的那一条；

在实际的应用中，因为 K 短路径搜索算法的计算量较大，一般算法都只搜索 $K \leq 3$ 条渐短路径，即假设客流分配的基础为乘客选择概率较高的三条渐短路径。

6.3.2 选择模型求解

基于改进的 Dijkstra 乘客最优路径选择算法具体步骤如下：

Step1：初始化。给相关变量赋初值，输入邻接矩阵，设定参数；

Step2：利用 Dijkstra 算法计算在起始站节点 q 和终点站节点 z 之间满足限制条件的最短路径阻抗；

Step3：从当前节点 i 出发，遍历与节点 i 相邻的节点，例如节点 j，如果从起始站对应节点 q 出发沿着该路径的阻抗小于或等于设定阈值并且满足换乘限制条件，则令节点 j 为当前点，转下一步，否则转 Step 7；

Step4：循环 n-1 次扩展结点，在状态为未扩展的顶点中选择时间最小的顶点 u，并将他的状态设为已扩展。对于每个与顶点 u 相邻的顶点 v，执行时间更新操作 Relax（u，v）。在此操作中，如果 $dist\ [u] + w\ [u, v] < dist\ [v]$，就把 $dist[v]$ 的值更新成 $dist[u]+w[u,v]$ 的值。更新后由源顶点 s 到顶点 v 的最短路径上，v 的前一个顶点为 u。

Step5：判断节点 j 是否为终节点 z，如果不是则转入 Step 2，否则进行下一步；

Step6：判断该路径是否包含冗余路段，如果是则忽略该路径并转入 Step 2，否则进行下一步；

Step7：搜索前驱节点序列并输出路径；

Step8： 退回上一层，若未退回到根节点则转入 Step 2。

6.4 结果的计算与分析

6.4.1 邻接矩阵的建立

问题 2 是求解城市轨道交通网络路径优化问题，根据已知附件数据中乘客的进出站（附件 1 线网乘客 O-D 数据）和时间列车的到发时刻（附件 3 列车运行图数据），所有在乘客出行信息还原模型和网络最优路径选择模型中是考虑时间问题的求解模型，因此网络 G 的为带权邻接矩阵为 n（n=328）阶方阵 T，边权为 t_{ij}，t_{ij} 是相邻两车站的走行时间与滞留时间之和，通过列车运行图数据可求解不同线路在任意两车站在时间，由于城市轨道交通车辆均是追踪发车，因此 t_{ij} 取所有车辆经过两车站的时间期望值，为考虑乘客的换乘时间 t_h，在附件 3 数据资料统计基础上得到期望换乘时间为 3.27min，因此本模型 t_h=3.27，表 6-1 给出八通线合计 13 个车站的邻接矩阵 T。

表 6-1 八通线邻接矩阵

相邻车站	四惠	四惠东	高碑店	传媒大学	双桥	管庄	八里桥	通州北苑	果园	九棵树	梨园	临河里	土桥
四惠	0	3.83	0	0	0	0	0	0	0	0	0	0	0
四惠东	3.83	0	3.00	0	0	0	0	0	0	0	0	0	0
高碑店	0	3.00	0	3.67	0	0	0	0	0	0	0	0	0
传媒大学	0	0	3.67	0	4.00	0	0	0	0	0	0	0	0
双桥	0	0	0	4.00	0	3.83	0	0	0	0	0	0	0
管庄	0	0	0	0	3.83	0	3.67	0	0	0	0	0	0
八里桥	0	0	0	0	0	3.67	0	3.50	0	0	0	0	0
通州北苑	0	0	0	0	0	0	3.50	0	3.17	0	0	0	0
果园	0	0	0	0	0	0	0	3.17	0	2.50	0	0	0
九棵树	0	0	0	0	0	0	0	0	2.50	0	2.83	0	0
梨园	0	0	0	0	0	0	0	0	0	2.83	0	2.67	0
临河里	0	0	0	0	0	0	0	0	0	0	2.67	0	2.33
土桥	0	0	0	0	0	0	0	0	0	0	0	2.33	0

6.4.2 乘客出行信息准确还原结果

根据乘客出行信息准确还原模型以及还原模型求解步骤，可得到乘客编号为 2、7、19、31、41、71、83、89、101、113、2845、124801、140610、164834、193196、223919、275403、286898、314976、315621 的完整出行线路。下面以乘客#2 为例详细介绍具体求解步骤：

表 6-2 乘客#2 已知信息表

乘客编号	始发车站（编号）	对应线路	目的地车站（编号）	对应线路	进站刷卡时刻	出站刷卡时刻
#2	西红门（84）	4 号线	北京站（47）	2 号线	8：24：00	9：14：24

根据 KSP 算法求解出时间依次最少的三条路径如表 6-3 所示：

表 6-3 乘客#2 最短 3 条路径

路径	路线信息
路径 1	西红门→宣武门→北京站（4 号线→2 号线）
路径 2	西红门→西单→建国门→北京站（4 号线→1 号线→2 号线）
路径 3	西红门→角门西站→宋家庄家→崇文门站→北京站（4 号线→10 号线→5 号线→2 号线）

在根据附件 3 列车运行图数据利用限制约束条件找到在乘客#进站刷卡与出站刷卡时刻的有效路径以及列车编号，对于路径 1：西红门→宣武门→北京站（4 号线→2 号线），通过模型求解可找到 4 号线（西红门→宣武门）共有 4 列班车，2 号线（宣武门→北京站）共有 6 列班车，其详细表见表 6-4 所示。

表 6-4 路径 1 有效列车编号

对应线路	列车编号	车站名称	列车到达时刻	对应线路	列车编号	车站名称	列车到达时刻
4 号线（西红门→宣武门）				2 号线（宣武门→北京站）	211141	宣武门	8：53：26
						北京站	9：02：39
	419091	西红门	8：28：32		211143	宣武门	8：55：56
		宣武门	8：51：52			北京站	9：05：09
	419093	西红门	8：33：41		211145	宣武门	8：57：56
		宣武门	8：57：01			北京站	9：07：09
	419095	西红门	8：39：41		211147	宣武门	8：59：56
		宣武门	9：03：01			北京站	9：09：09
	419097	西红门	8：45：41		213001	宣武门	9：01：56
		宣武门	9：09：01			北京站	9：11：09
					211149	宣武门	9：03：56
						北京站	9：13：09

在根据对应换乘站宣武门站，必须满足前一列车的到达时刻≤后一列车的到达时刻，有效列车标号下的有效路径共计 12 条，（依次为 419091→211141，419091→211143，419091 → 211145， 419091 → 211147， 419091 → 213001， 419091 → 211149，419093 → 211145， 419093 → 211147， 419093 → 213001， 419093 → 211149，419093→211149，419095→211149）。按照相同步骤继续求解路径 2，路径 3 对应的有效列车编号，最后考虑时间间隔（包括进站时刻至上车时刻，下车时刻至出站时刻，换乘间隔时间）最合理的情况下选择最优路径，即 419093→213001，经过以上分析，准确还原出乘客#2 的完整出行线路。

依次求解出余下乘客的完整出行线路见表 6-5 所示。（其中乘客#113 号与乘客#286898 无有效出行路径，乘客#101 因 14 号线七里庄站暂缓通，无有效路径。）

表 6-5 乘客出行准确数据

乘客	车站	进站时间	列车 1	换乘站 1	列车 2	换乘站 2	列车 3	换乘站 3	列车 4	换乘站 4	列车 5	车站	出站时间
#2	84	8：24：00	419093	宣武门	213001	—	—	—	—	—	—	47	9：14：24
#7	163	6：58：00	808008	鼓楼大街	—	—	—	—	—	—	—	54	7：29：23
#19	251	8：47：00	1349070	东直门	212106	—	—	—	—	—	—	45	9：22：29
#31	95	7：30：00	500096	雍和宫	211071	—	—	—	—	—	—	38	8：16：03
#41	1	8：17：00	106125	建国门	211141	—	—	—	—	—	—	52	9：17：12
#71	94	9：02：00	500178	崇文门	211171	—	—	—	—	—	—	47	9：52：04
#83	246	8：33：00	1353067	西直门	211157	—	—	—	—	—	—	45	9：20：24
#89	145	9：25：00	601066	车公庄	—	—	—	—	—	—	—	39	9：54：23
#101	262	10：07：00	因 14 号线七里庄站暂缓通，无有效路径									47	11：04：03
#113	307	5：59：00	无有效路径									47	6：56：24
#2845	93	7：25：00	419069	宣武门	211111	—	—	—	—	—	—	50	8：45：20
#124801	59	7：52：00	414060	海淀黄庄	1035038	芍药居	1353081	望京西				268	8：40：42
#140610	166	11：01：00	809144	霍营	1349152	—	—	—	—	—	—	268	11：50：50
#164834	11	8：17：00	106101	东单	501123	大屯路东	1519065	—	—	—	—	279	9：42：49
#193196	34	7：22：00	366062	四惠	103096	国贸	1039005	芍药居	1353083	望京西	1519059	276	9：20：42
#223919	229	6：46：00	1041033	西直门	418025	西直门	212104	—	—	—	—	37	8：48：25
#275403	150	8：32：00	600060	平安里	419099	西直门	1349120	—	—	—	—	241	9：40：23
#286898	86	11：14：00	无有效路径									244	12：40：06
#314976	3	7：23：00	106069	复兴门	208032	西直门	1349076	—	—	—	—	241	8：20：49
#315621	19	8：14：00	103104	建国门	211101	东直门	1353085	—	—	—	—	251	8：42：51

备注：乘客#113 号与乘客#286898 无有效出行路径，乘客#101 因 14 号线七里庄站暂缓通，无有效出行路径。

6.4.3 乘客最优路径选择结果

根据基于改进的 Dijkstra 乘客最优路径选择模型和算法可求解任意两车站之间的最优路径，为验证该模型的准确性和合理性，选取任意两站计算耗时时间最短的最优路径，根据该结果与北京地铁官方网站路线方案查询结果相比较。同时，在选择最短时间路径中考虑该时刻的拥挤因素，分析在拥挤情况下的路径选择结果。

验证方案：选取从天安门西站（对应于 1 号线，车站编号为 14 号）至关庄站（对应于 15 号线，车站编号为 281 号），基于 6.2.2 节中的乘客最优路径选择模型和 6.3.2 节模型求解步骤计算结果如表 6-6 所示。

表 6-6 天安门西站→关庄站最优线路方案

车站编号	14	15	16	**17**	**110**	109	108	107	106	105	104	103	102	101	**100**	**282**	281
车站名称	天安门西	天安门东	王府井	**东单**	**东单**	灯市口	东四	张自忠路	北新桥	雍和宫	和平里北街	和平西桥	惠新西街南口	惠新西街北口	**大屯路东**	**大屯路东**	关庄
对应线路	1	1	1	**1**	**5**	5	5	5	5	5	5	5	5	5	**5**	**15**	15

该最优方案显示：天安门西站→关庄站共途经 14 站；经过 3 条线路，分别为 1 号线，5 号线，15 号线；需换乘 2 次，分别在东单站、大屯路东站换乘；且该最优方案共耗时 43.2600min。

对比北京地铁官方网站路线方案查询结果，在查询栏搜索起终站站名，得到查询结果如图 6-4 所示，线路走行图见图 6-5 所示，通过该结果可看出线路走行方案与本文建立的模型求解结果完全吻合，显示耗时时间约 43min，与求解出来的 43.2600min 也基本一致。

图 6-4 查询结果图

图 6-5 线路走行图

当考虑该时刻的拥挤情况，设地铁 5 号线此时拥挤程度较高，如乘客选择 5 号线去

往关庄站，将可能因为拥挤状况而延误走行时间，而此时如果选择另一路线方案，其最终走行时间可能更优于上述方案。

为有效量化此时地铁 5 号线的拥挤程度，通过对地铁 5 号线所对应的车站的邻接矩阵 T 进行更改，增加两车站之间的耗时时间 t_{ij}，两车间之间的耗时时间 t_{ij} 增加 30s，t_h 增加一倍换乘时间，则原来方案在此时走向共计需耗时 43.26+5.5+3.27=52.03min，根据以上模型计算考虑拥挤情况的天安门西站→关庄站最优线路方案结果见表 6-7 所示。

表 6-7 考虑拥挤情况的天安门西站→关庄站最优线路方案

车站编号	14	15	16	17	18	48	49	50	51	253	252	251	250	249	268	281
车站名称	天安门西	天安门东	王府井	东单	建国门	建国门	朝阳门	东四十条	东直门	东直门	柳芳	光熙门	芍药居	望京西	望京西	关庄
对应线路	1	1	1	1	1	2	2	2	2	13	13	13	13	13	15	15

通过表 6-7 可知，此时天安门西站→关庄站共途经 13 站；经过 4 条线路，分别为 1 号线，2 号线，13 号线，15 号线；需换乘 3 次，分别在建国门站、东直门站、望京西站换乘；计算得到考虑拥挤情况该最优方案共耗时 46.9200min，显然方案是小于 52.03min，从而进一步验证了该模型的的科学性和运算的准确性。

七、问题 3 的模型建立与求解

7.1 问题分析

7.1.1 背景分析

北京地铁八通线（以下简称“八通线”）是北京地铁的运营线路之一，属城市轨道交通系统，于 2003 年 12 月 27 日全线开通，连接北京中心城区及东部的通州新城，线路标识色为红色，示意图如图 7-1 所示。

八通线线路呈东西走向，全长 18.964km，最高时速为 70km/h，采用的是固定 6 节编组，标准 B 型列车，上行方向为四惠站至土桥站，运营时间为 06：00~23：50，下行方向为土桥站至四惠站，运营时间为 5：20~23：10。全线共设 13 个车站，其中四惠站和四惠东站为与 1 号线的换乘车站，站台形式为岛式，其他车站均为单线运营，站台形式均为侧式[17]。

图 7-1 北京地铁八通线线路示意图

本文北京地铁八通线的实际客流数据为依据，研究范围为早高峰 7：00~9：00 时段，共 120min，主要包括各站进站口客流量和乘客出行数据。

7.1.2 数据准备

（1）各站进站口客流量数据

对八通线实际 OD 数据进行统计分析，可得以八通线下行方向全线各站进站，客流量数（四惠作为下行方向终点站，不予考虑），见表 7-1：

表 7 -1 全时段八通线进站量表

车站编号	控制时段		
	7 点以前	7：00~9：00	9：00 以后
四惠	655	2918	1901
四惠东	681	2607	1198
高碑店	185	551	429
传媒大学	678	2176	880
双桥	1018	2663	1165
管庄	877	2239	999
八里桥	229	873	262
通州北苑	669	2407	1205
果园	706	2544	848
九棵树	411	1354	599
梨园	1115	2689	1108
临河里	515	1340	427
土桥	1371	3447	1110

根据表 7-1 绘制各车站各时间段出行情况曲面图，如图 7-2 所示。

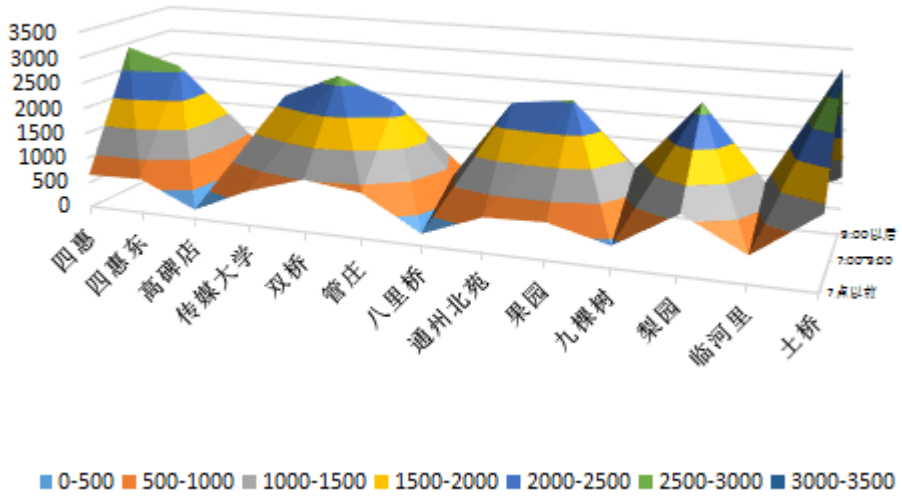


图 7-2 八通线各车站各时间段出行情况曲面图

根据北京地铁八通线的实际客流数据为依据，出行高峰时段为早上 7：00~9：00 时段，因此本文北京地铁八通线的实际客流数据为依据，研究范围为早高峰 7：00~9：00 时段，共 120min。

对八通线实际 OD 数据进行统计分析，可得以八通线下行方向全线早上 7：00~9：00 时段各站进站，客流量数据（控制时段取为 30min），见表 7-2：

表 7-2　7：00~9：00 时段八通线进站量表

车站编号	控制时段			
	7：00~7：30	7：30~8：00	8：00~8：30	8：30~9：00
四惠	641	866	838	573
四惠东	555	828	736	488
高碑店	116	157	150	128
传媒大学	627	738	450	361
双桥	806	816	566	475
管庄	520	736	495	488
八里桥	243	263	208	159
通州北苑	589	722	601	495
果园	734	736	640	434
九棵树	352	423	358	221
梨园	776	787	645	481
临河里	383	440	329	188
土桥	1067	955	875	550

根据表 7-2 绘制各车站各时间段出行情况曲面图，如图 7-3 所示。

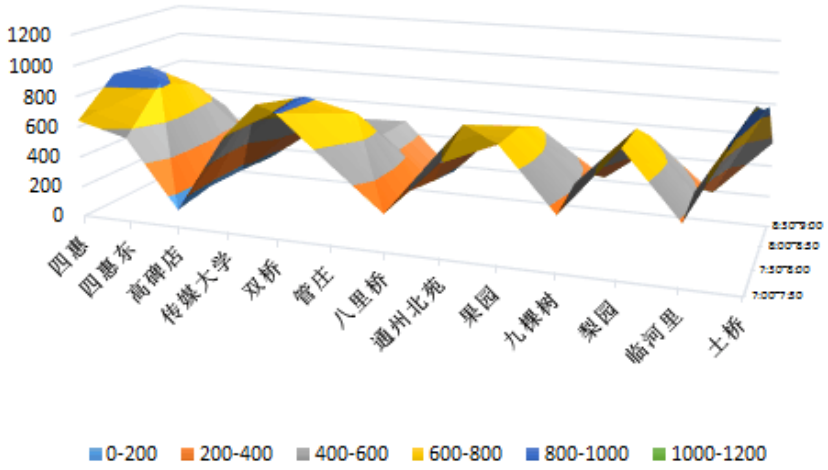


图 7-3 八通线各车站早上 7：00~9：00 时段出行情况曲面图

（2）各站出站口客流量数据

对八通线实际 OD 数据进行统计分析，可得以八通线下行方向全线各站早上 7：00~9：00 时段出站客流量数据，见表 7-3：

表 7-3 7：00~9：00 时段八通线出站量表

车站编号	控制时段			
	7：00~7：30	7：30~8：00	8：00~8：30	8：30~9：00
四惠	0	0	0	0
四惠东	0	0	0	0
高碑店	257	449	817	1269
传媒大学	171	314	349	402
双桥	284	400	602	800
管庄	232	279	380	439
八里桥	73	58	78	68

通州北苑	241	294	291	392
果园	106	111	129	132
九棵树	101	160	318	300
梨园	174	292	290	289
临河里	66	189	183	108
土桥	217	285	372	413

以四惠→四惠东→高碑店→传媒大学→双桥→管庄→八里桥→通州北苑→果园→ 九棵树→梨园→临河里→土桥方向为研究方向。乘客下车率=前站上车人数/本站下车人数，根据表 7-2，表 7-3 的数据可得八通线各站的乘客下车率，结果见表 7-4 乘客下车率表。

表 7-4 乘客下车率

车站	四惠	四惠东	高碑店	传媒大学	双桥	管庄	八里桥	通州北苑	果园	九棵树	梨园	临河里	土桥
下车率	0.00	0.00	0.55	0.21	0.27	0.13	0.02	0.09	0.03	0.05	0.05	0.02	0.06

本文所建单线协同限流策略优化模型主要分析车站初始客流输入和客流在车站与列车间的转移过程两个部分，因此约束条件主要围绕以下四个方面进行描述，分别为进站口乘客出行行为约束、列车位置信息约束、站台乘客上下车行为约束和单线多站服务系统能力约束[10]。城市轨道交通大客流呈现非均衡、非稳态、非线性特征，直接影响乘客服务水平和列车运行可靠性，是车站客运组织管理中的重要环节[8, 9]。在分析大客流条件下乘客候车延误影响因素的基础上，考虑列车周期性运行特性，并抽象乘客到达规律符合均匀分布，以各站乘客均衡候车为优化目标，构建线路层车站间的单目标多约束协同限流模型，并基于 Matlab 非线性规划 fmincon 函数进行求解。

7.2 模型建立

7.2.1 不限制限流车站个数模型

$$\min\left(\frac{\sum_{k=1}^{i}\mathrm{D}_{k1}-\sum_{k=1}^{i-1}S_{k1}}{S_{i1}}+\frac{\sum_{k=1}^{i}D_{k2}-\sum_{k=1}^{i-1}\mathrm{S}_{k2}}{\mathrm{S}_{i2}}+\cdots+\frac{\sum_{k=1}^{i}D_{kj}-\sum_{k=1}^{i-1}S_{kj}}{S_{ij}}\right) \tag{7-1}$$

$$\text{s.t}\quad .\ Q_{ij}=\begin{cases}C-\sum_{j=1}^{j}S_{ij}(1-\alpha_j)\cdots(1-\alpha_2)(1-\alpha_1) & j\geq 2\\ C & j=1\end{cases} \tag{7-2}$$

$$\lambda_{ij}=\begin{cases}\dfrac{D_{ij}+(D_{(i-1)j}-S_{(i-1)j}\times n_i)}{H_i} & i\geq 2\\ \dfrac{D_{ij}}{H_{ij}} & i=1\end{cases} \tag{7-3}$$

$$n_i=\begin{cases}\left\lfloor\dfrac{H_i\times i}{T}\right\rfloor-\sum_{k=1}^{i=1}\left\lfloor\dfrac{H_k\times k}{T}\right\rfloor & i\geq 2\\ \left\lfloor\dfrac{H_i\times i}{T}\right\rfloor & i=1\end{cases} \tag{7-4}$$

$$P_{ij}=\begin{cases}(P_{(i-1)j}+\lambda_{ij}-S_{ij})/n & i\in(2,i)\\ \lambda_{ij}T/n & i=1\end{cases} \tag{7-5}$$

$$S_{i,j}\le\min(Q_{ij},\ P_{ij}) \tag{7-6}$$

其中，定义影响候车时间的相关参数如下：

Hi 为第 *i* 个控制时间段时长；

Dij 为第 *i* 个控制时间段第 *j* 站到达的乘客总人量；

Sij 为第 *i* 个控制时间段第 *j* 站各次列车允许的上车人数；

z 为客流分解点（即列车通过该客流分解点后运能快速释放）；

Qij 为列车在第 *i* 个控制时间段第 *j* 站各次列车的剩余容纳能力（单位：人）；

C 为列车定员人数；

αj 为乘客在第 *j* 站下车率（单位：人）；

Pij 为第 *i* 个控制时间段内首次列车到站时第 *j* 站的候车人数；

λij 为第 *i* 个控制时间段内第 *j* 站客流均匀到达率（ 单位：人）。

T 为列车发车时间间隔（单位：min），下面以各控制时间段 （第 *i* 周期）全线 *j* 站等待时间最小且均衡为目标，构建多站协同限流模型，如式（7-2） 至式（7-5）

式（7-2）表示列车在各控制时间段内各次列车在各站剩余能力，是上车客流人数的限制条件，列车总承载量为列车定员人数与满载系数的乘积；

式（7-3）表示第 *j* 站在第 *i* 个控制时间段内客流平均到达率；

式（7-4）表示第 *j* 站在第 *i* 个控制时间段内可开行的列车对数；

式（7-5）表示各控制时间段内首次列车到站时各站的候车人数；

式（7-6）为最佳上车人数的约束条件，该值既应满足客流需求上限约束，即上车人数不能超过当前候车人数，同时又满足列车剩余能力的上限，即不能超过列车可容纳量。

7.2.2 八通线限制限流车站个数

（1）决策变量与目标函数

具体计算公式如式 7-7，7-8，7-9 所示。由于在实际运营中，乘客对于因限流和留乘所造成的出行延误较为敏感，当乘客在进站口因限流延误时间或者在站台因留乘延误时间过长时，会极大的影响城市轨道交通车站运营安全，降低乘客出行满意程度，因此，对乘客因限流延误时间和乘客因留乘延误时间设定延误时间惩罚系数。

$$T_{ql}=\sum_{t=1}^{TE}\sum_{s=1}^{S-1}\left[q_s^{l}(t)\cdot\frac{\Delta t}{2}\right] \tag{7-7}$$

$$T_{qr}=\sum_{t=1}^{TE}\sum_{s=1}^{S-1}\left[q_s^{d}(t)\cdot f\right] \tag{7-8}$$

$$T_{delay}=\left[\theta_1 T_{ql}+\theta_2 T_{qr}\right]=\sum_{t=1}^{TE}\sum_{s=1}^{S-1}\theta_1\left[q_s^l(t)\cdot\frac{\Delta t}{2}\right]+\sum_{t=1}^{TE}\sum_{s=1}^{S-1}\theta_2\left[q_s^d(t)\cdot f\right] \tag{7-9}$$

（2）模型约束条件

①进站口乘客出行行为约束

$$q_s^o(t)=R_s^{\alpha r}(t)\cdot\Delta t \tag{7-10}$$

$$q_s^h(t)=q_s^o(t)+q_s^t(t-1) \tag{7-11}$$

$$q_s^e(t)=q_s^h(t)-q_s^t(t) \tag{7-12}$$

式 7-10 为在 t 时段内，进站口到达的客流人数约束，表示乘客已经到达车站进站口等待，为乘客出行需求输入；

式 7-11 为在 t 时段内车站进站口聚集人数约束，用 t 时段内新到达的乘客人数与 $t-1$ 时段内的限流人数之和表示；

式 7-12 为能够进入车站站台的乘客人数约束，用进站口在 t 时段内的聚集人数与当前限流人数的差值表示，为车站进站口与车站站台的链接环节。

②列车位置信息约束

在模型限流时段不断推进的过程中，列车在线路上运行，不能保证恰好在一个限流时段内有列车进入车站，触发乘客上下车行为。

$$L_{s,r}(t)=\begin{cases}0\text{（列车r 未处于车站s 内）}\\1\text{（列车r 处于车站s 内）}\end{cases} \tag{7-13}$$

③站台乘客上下车行为约束

$$q_s^p(t)=\begin{cases}q_s^p(t-1)+q_s^e(t),\ \sum_{t=1}^{R}L_{s,r}(t)=0\\q_s^d(t),\ \sum_{r=1}^{R}L_{s,r}(t)=1\end{cases} \tag{7-14}$$

$$q_s^b(t)=\sum_{r=1}^{R}\sum_{r=1}^{s-1}q_s^b(t)\cdot P_{l,s}\cdot L_{s,r}(t) \tag{7-15}$$

$$q_s^b(t)=\begin{cases}\sum_{r=1}^{R}L_{s,r}(t)\cdot\left[q_s^p(t-1)+q_s^e(t)\right],\ q_s^p(t-1)+q_s^e(t)\le q_r^{av}(t-1)+q_s^a(t)\\\sum_{r=1}^{R}L_{s,r}(t)\cdot\left[q_s^{av}(t-1)+q_s^a(t)\right],\ q_s^p(t-1)+q_s^e(t)\ge q_r^{av}(t-1)+q_s^a(t)\end{cases} \tag{7-16}$$

$$q_s^d(t)=\sum_{r=1}^{R}L_{s,r}(t)\cdot\left[q_s^p(t-1)+q_s^e(t)-q_s^b(t)\right] \tag{7-17}$$

$$q_s^l(t)\le q_s^o(t)+q_s^l(t-1) \tag{7-18}$$

$$q_r^{av}(t)=\eta^{\max}\cdot C-q_r^m(t) \tag{7-19}$$

式 7-14 是对 t 时段结束后站台人数的计算，分为是否有列车进站两种情况分别计算；

式 7-15 是对乘客下车过程的描述；

式 7-16 是对乘客上车过程的描述，对站台候车人数与列车载客能力的关系分情况处理，上车人数取二者中的较小值；

式 7-17 是对乘客留乘现象的描述，是对 t 时段站台候车人数与上车人数的关系进行分析；

式 7-18 是对 t 时段结束后列晕载客人数的计算；

式 7-19 是对 t 时段结束后列车剩余载客能力的计算，为列车最大载客人数与列车当前载客人数的差值。

④单线多站服务系统能力约束

$$q_s^b(t) \le q_s^p(t) \tag{7-20}$$

$$q_s^p(t) \le Q_s^p \tag{7-21}$$

$$q_s^l(t) \le q_s^o(t) + q_s^l(t-1) \tag{7-22}$$

$$q_s^b(t) \le q_s^p(t) \tag{7-23}$$

式 7-20 表示列车能力约束，列车内载客人数不能超过列车最大载客人数；

式 7-21 表示站台能力约束，站台候车人数不能超过站台允许的最大聚集人数能力；

式 7-22 表示限流人数不能超过进站口聚集人数；

式 7-23 表示站台上车人数不能超过站台候车人数。

7.3 模型求解

（1）不限制限流车站个数模型

目标非线性数学模型，约束条件为线性不等式，可用 Matlab 的非线性规划 fmincon 函数进行求解，逐次递推得到各分时间段内各站最佳上车人数以及客流平均等待周期、平均控流率等指标． 其中，控流率为车站在控制时段内不能满足的客流需求与实际客流需求的比值。

（2）若八通线限制限流车站个数

本文构建模型具有输入参数多，包括进站口客流出行需求、乘客出行矩阵等；中间变量变化快，包括站台人数、上下车人数、留乘人数等均随着限流时段的推进不断变化；输出结果为各限流时段内线路各站的限流人数，为输出 $S \times TE$ 的结果矩阵，当车站个数较多，总限流时段较长时，导致模型求解规模巨大的特点，很难直接使用数学方法求得模型准确的最优解，因此，本文选取能在大规模解空间内快速搜索最优解的改进模拟退火算法进行模型求解。

算法步骤：

Step1：在未进行协同限流的情况下，乘客进站，分别计算该限流时段内，列车在各站出发之前的站台聚集人数，并判断该聚集人数是否超过车站站台允许的最大聚集人数能力。如果超过站台最大聚集人数，则需在进站口采取相应措施，制定阻止超过最大聚集人数的部分乘客流入站台的初步限流策略，以保证车站站台能力的限制。

Step2：在实行初步限流策略的基础上，对各车站限流人数做随机扰动，得到一种新的单线协同限流策略并将其输入单线多站服务系统中。

Step3：列车到达某车站，进行乘客上下车过程。列车载客能力大于车站站台聚集人数时，乘客全部上车；列车载客能力小于车站站台聚集人数时，乘客未能全部上车，出现留乘现象。

Step4：计算目标函数值，并以此为依据，判断是否接受 Step2 中对单线协同限流策略的调整。

Step5：不断重复 Step2，Step3，Step4，直至目标函数值的变化范围小于精度要求 5‰。或达到最大迭代次数后，停止计算，此时得到的单线协同限流策略，即为算法所得最优

单线协同限流策略。

7.4 结果分析与验证

根据 Matlab 计算结果得，早上 7：00~9：00 时段八通线进站限流人数，结果见表 7-5 早上 7：00~9：00 时段八通线进站限流人流量表。

表 7-5 早上 7：00~9：00 时段八通线进站限流人流量表

车站编号	控制时段			
	7：00~7：30	7：30~8：00	8：00~8：30	8：30~9：00
四惠	160	217	210	143
四惠东	239	339	315	212
高碑店	86	129	114	81
传媒大学	131	153	97	82
双桥	260	280	184	151
管庄	213	254	175	162
八里桥	110	149	106	97
通州北苑	144	167	141	111
果园	236	258	220	164
九棵树	170	180	156	98
梨园	192	206	169	121
临河里	193	204	161	110
土桥	251	238	209	126

7.4.1 不限制限流车站个数

若八通线不限制限流车站个数，试分析限流前后的总出行时间、平均出行时间对比，结果见表 7-6 八通线不限制限流车站个数表。

表 7-6 八通线不限制限流车站个数

对比项	总出行时间（min）	平均出行时间（min/人）
限流前	834240	22.82
限流后	272259	11.37
差值	561981	11.45
减少百分比	67.4%	50.2%

7.4.2 限制限流车站个数

若八通线限制限流车站个数（分别取限流车站数为 1-5 个车站），试分析限流前后的总出行时间、平均出行时间对比，结果见表 7-7 所示。

表 7-7 八通线不限制限流车站个数

对比项	限流车站数量：1	双桥	限流车站数量：2	双桥、管庄
	总出行时间（min）	平均出行时间（min/人）	总出行时间（min）	平均出行时间（min/人）
限流前	834240	22.82	834240	22.82
限流后	755543	20.93	689888	19.21
差值	78697	1.89	144352	3.61
百分比	9.4%	8.3%	17.3%	15.8%
对比项	限流车站数量：3	双桥、管庄、八里桥站	限流车站数量：4	双桥、管庄、八里桥站、传媒大学
	总出行时间（min）	平均出行时间（min/人）	总出行时间（min）	平均出行时间（min/人）
限流前	834240	22.82	834240	22.82
限流后	629851	17.71	574553	16.07
差值	204389	5.11	259687	6.75
百分比	24.5%	22.4%	31.1%	29.6%
对比项	限流车站数量：5		双桥、管庄、八里桥站、传媒大学、四惠	
	总出行时间（min）		平均出行时间（min/人）	
限流前	834240		22.82	
限流后	520917		15.43	
差值	313323		7.39	
百分比	37.6%		32.3%	

7.4.3 两个限流效果最好的车站

根据表 7-5、表 7-6 的结果，得到限流车站的个数、减少率与累计减少率结果见表 7-7 所示。

表 7-7 八通线不限制限流车站个数

限流车站	减少率	累计减少率
双桥	0.094	9.4%
双桥、管庄	0.079	17.3%
双桥、管庄、八里桥站	0.072	24.5%
双桥、管庄、八里桥站、传媒大学	0.066	31.13%
双桥、管庄、八里桥站、传媒大学、四惠	0.064	37.56%

根据表 7-7 减少率以及累计减少率结果，选择双桥、管庄进行限流，双桥的减少率最高，双桥、管庄进行限流时累计减少率达到 17.3%。

总限流时段内各站的乘客出行延误时间包括乘客因限流导致总延误时间和因留乘导致总延误时间，模型所制定单线协同限流策略其目的是通过控制线路各站的限流人数，改变乘客到达站台的时间[1, 11]。在车站层面，随着限流人数的增多，乘客因限流导致的延误时间增多，但进站乘客和站台滞留乘客减少，乘客因留乘延误时间相应的减少；在线路层面，随着上游车站限流人数的增多，乘客因限流延误时间增多，但预留的列车剩余能力增多，使下游车站输送乘客人数增多，相应限流和留乘人数减少。因此模型与实际情况具有较好的符合性。

八、 问题 4 的分析与求解

8.1 问题分析

要求给出具体限流措施以改进城市轨道交通的服务水平，基于问题一中乘客出行特征，问题二中乘客最优路径的难以选择，问题三中限流对针对乘客滞留时间的减少，与出行效率的提高，结合城市轨道交通系统具有的动态时变特点，随着路网不断扩大，客流不断增加。基于问题一客流乘客出行特征（客流时间分布、客流空间分布和客流时空分布）的分析数据以及问题三限流模型和限流方案，提出采用动态限流方案、限流车站的选取、限流时段的选择、限流强度等多种措施对城市轨道交通多站协同客流控制问题进行研究的方法。基于对北京市城市轨道交通客流限流控制策略研究，有效解决目前研究的限流方案难以根据动态客流变化做出相应的调整的问题，进一步丰富城市轨道交通客流需求管理的理论和方法。

8.2 具体限流措施

由问题一可得：进站客流人数排名前 3 的线路依次为 4 号线、10 号线、13 号线，进站客流人数分别为 163887 人、122148 人、96122 人；出站客流人数排名前 3 的线路依次为 4 号线、2 号线、13 号线，出站客流人数分别为 253140 人、239758 人、124880 人。进站客流排名前 10 的车站依次为回龙观、龙泽、天通苑北、沙河天通苑、回龙观东大街、苹果园、霍营、西二旗、新宫，排名前 10 的车站累计进站人数均超过 1 万人次，进站客流最大的车站为回龙观车站，累计进站客流达 20346 人；出站客流排名前 10 的车站依次为西二旗、阜成门、东直门、五道口、中关村、东四十条、西直门、朝阳门、北京南站、海淀黄庄，排名前 10 的车站累计出站人数除海淀黄庄车站均超过 2 万人次，出站客流最大的车站为西二旗车站，累计出站客流达 41160 人。并结合问题三的限流模型提出动态限流方案以及动态限流实现方法。

限流策略研究其主要目的是控制客流分布，降低运输压力与事故风险，从而保障地铁安全。常见的主要措施有动态限流方案、限流车站的选取、限流时段的选择、限流强度的设置，主要方式有关闭售票机、闸机、设置栏杆、关闭出入口、关闭换乘通道、关闭车站等，主要应用场合有：

（1）高峰时段，运营方采取设置障碍物，迂回绕行等手段限制单位时间内的客流量，降低车站或线路的拥挤程度。

（2）发生地铁延误或故障时，运营方对相关的换乘通道设置障碍物，放缓或限制换乘客流，以避免更多客流积压在延误或故障的线路上。

（3）大型活动举办时，临近的车站会对参加活动的客流进行预估，并制定相应的限流方案，避免进场散场时的瞬时大客流对车站与车厢造成过大压力。

作为短期应对措施，限流的效果是比较明显的，但是作为长期应对措施时的效果有限。如果条件允许，提高运能、合理分配运力才是行之有效的措施[17]。

北京地铁八通线除东端前两站土桥站、临河里站及西段的高碑店站外，几乎全线限流，其中传媒大学、双桥、管庄、八里桥站限流时间均集中在 7 时至 9 时；通州北苑、果园、九棵树、梨园站的限流时间则在 7 时至 8 时 30 分。双桥、管庄两站早晚高峰都限流。具体结果见表 8-1 北京地铁限流时段及车站表。

表 8-1 北京地铁限流时段及车站表

时段	车站
早高峰 7：30 严重拥挤区段	八通线：双桥到四惠东 6 号线：物资学院到草房区段 黄渠到呼家楼区段 5 号线：小红门到宋家庄区段 昌平线：沙河到西二旗区段 13 号线：西二旗到五道口区段 4 号线和大兴线：高米店南到公益西桥、北京南站宣武门 9 号线：长阳到郭公庄、 丰台南路到丰台东大街 1 号线：八宝山到玉泉路

在北京市 19 个重点换乘车站和大客流冲击严重车站的出入口、站台、换乘通道、列车两端，安装客流感知和图像监控设备。根据车站的客流流量和密度实现地铁站限流的智能化和动态化，而不是将限流时间设定为一个小时或半个小时的固定时间模式，在保证安全运营的前提下让乘客快速进出站。

九、 模型的评价与推广

9.1 模型的评价

本文所建立的城市轨道交通网络优化模型，是基于实际的乘客出行 OD 数据进行研究，利用 KSP 算法能够有效还原乘客准确出行信息，基于改进的 Dijkstra 乘客最优路径选择模型和算法可求解任意两车站之间的最优路径，并通过案例验证天安门西站→关庄站的最优路径，其结果与北京地铁官方网站路线方案查询结果相比较能够较好的反应实际情况。

由于时间精力有限，建立的模型也有一些不足之处，在模型建立过程中进行的大量假设，模型考虑的因素有限，求解过程中采取近似计算，对模型的精确性带来一定的影响。

9.2 模型的推广

基于数据驱动的城市轨道交通网络优化策略，运用不同的模型分析问题，在乘客出行信息中建立 KSP 模型，在乘客最优路径选择中，引入路径阻抗，路径伸展系数建立改进的 Dijkstra 模型，其中每一个模型都可以推广到现实生活中去，这就很好的体现本次数学建模的意义所在。

第一问中的数据处理以及对大数据的分析方法，SPSS 统计分析，乘客出行特征分析等，可以运用到金融、IT 等大数据处理中，以及铁路、公交等乘客客流特征分布。

第二问的乘客信息还原模型以及最优路径模型可以起到指导其他交通乘客出行选择缩短行程、减少拥挤，以及物流方案的选择。

第三问、第四问的限流模型可以推广到各条线路的各个车站，为政府部门和乘客的舒适性提供更好的指导。

十、 参考文献

[1] 周薇. 城市轨道交通有效路径选择的改进 Dial 算法[J]. 西华大学学报(自然科学版). 2013, 32(06): 38-40.
[2] 谢丽平. 城市轨道交通路网多站协同客流控制研究[D]. 北京交通大学, 2018.
[3] 徐涛，丁晓璐，李建伏. K 最短路径算法综述[J]. 计算机工程与设计. 2013, 34(11): 3900-3906.
[4] 张晨，王明根，李宇豪，等. 基于图论和广度优先搜索算法的分酒问题一般解的研究[J]. 数字技术与应用. 2018, 36(04): 38-39.
[5] 刘剑锋，孙福亮，柏赟，等. 城市轨道交通乘客路径选择模型及算法[J]. 交通运输系统工程与信息. 2009, 9(02): 81-86.
[6] 陈坚，王曼，李和平，等. 城市轨道交通网络乘客换乘路径选择行为模型[J]. 交通运输系统工程与信息. 2017, 17(06): 235-241.
[7] 王保山，丁勇，刘海东. 城市轨道交通网络路径生成方法[J]. 计算机工程与应用. 2012, 48(36): 27-30.
[8] 姜曼. 城市轨道交通单线多站协同客流控制研究[D]. 北京交通大学, 2016.
[9] 赵锴. 城市轨道交通单线协同限流策略优化研究[D]. 北京交通大学, 2018.
[10] 黄文慧. 城市轨道交通路网客流传播建模与限流策略研究[D]. 北京交通大学, 2017.
[11] 黄倩. 城市轨道交通线路多车站协同限流方法研究[D]. 西南交通大学, 2017.
[12] 陈志杰. 城市轨道交通线路负荷实时推算模型及控制方法[D]. 北京交通大学, 2018.
[13] 仲飞翔，肖为周，郭文. 轨道交通网络有效路径搜索算法的改进及实现[J]. 西华大学学报(自然科学版). 2019, 38(01): 108-112.
[14] 胡昆. 基于改进蚁群算法的避免拥堵最优路径选择[D]. 西南交通大学, 2018.
[15] 赵延龙，滑楠，于振华. 一种基于 BFS 的数据链站点优化选址算法[J]. 火力与指挥控制. 2018, 43(12): 103-108.
[16] 周玮腾. 拥塞条件下的城市轨道交通网络流量分配演化建模及疏导策略研究[D]. 北京交通大学, 2016.
[17] 周玮腾. 拥塞条件下的城市轨道交通网络流量分配演化建模及疏导策略研究[D]. 北京交通大学, 2016.

附录

附录 1：求解问题一各数据

表 1 出行时段分布数据

	时间段	进站人数	所占百分比		时间段	出站人数	所占百分比
进站时段分布	4:00-5:00	547	0.05%	出站时段分布	4:00-5:00	41	0.00%
	5:00-6:00	22088	2.11%		5:00-6:00	2776	0.27%
	6:00-7:00	119979	11.44%		6:00-7:00	38608	3.70%
	7:00-8:00	329623	31.44%		7:00-8:00	188633	18.10%
	8:00-9:00	286654	27.34%		8:00-9:00	384482	36.89%
	9:00-10:00	129888	12.39%		9:00-10:00	201858	19.37%
	10:00-11:00	82848	7.90%		10:00-11:00	99151	9.51%
	11:00-12:00	76929	7.34%		11:00-12:00	79467	7.62%
	合计	1048556	99.90%		12:00-13:00	47364	4.54%
					合计	1042339	99.40%

表 2 出行时长分布数据

	时间分钟数	乘客人数	所占百分比		时间分钟数	乘客人数	所占百分比
整体出行时长分布	0-30 分钟	391173	37.38%	早高峰出行时长分布	0-30 分钟	0	38.87%
	30 分钟-1 小时	492147	47.03%		30 分钟-1 小时	296686	48.26%
	1 小时-1 小时 30 分钟	137119	13.10%		1 小时-1 小时 30 分钟	0	11.25%
	1 小时 30 分钟 -2 小时	0	2.03%		1 小时 30 分钟 -2 小时	0	1.34%
	大于 2 小时	3334	0.45%		大于 2 小时	1225	0.29%

表 3 线路客流分布

线路	进站人数	出站人数
1 号线	83451	1
2 号线	88264	239758
3 号线	38291	29888
4 号线	163887	253140
5 号线	91156	0
6 号线	70172	1420
7 号线	37069	54459
8 号线	53323	50660
9 号线	40193	74969
10 号线	122148	99588

11 号线	1	0
12 号线	17182	0
13 号线	96122	124880
14 号线	18430	0
15 号线	42603	52335
18 号线	58696	54243
19 号线	27502	13234

表 3 进站客流排名数据

排名	车站名	所属线路	进站客流人数
1	回龙观	13	20346
2	龙泽	13	16819
3	天通苑北	5	15189
4	沙河	18	14655
5	天通苑	5	14533
6	回龙观东大街	8	11305
7	苹果园	1	11088
8	霍营	8	10739
9	西二旗	18	10711
10	新宫	4	10592

表 4 出站客流排名数据

排名	车站名	所属线路	出站客流人数
1	西二旗	18	41160
2	阜成门	2	28019
3	东直门	2	27354
4	五道口	13	25590
5	中关村	4	25399
6	东四十条	2	25188
7	西直门	2	24967
8	朝阳门	2	22546
9	北京南站	4	20919
10	海淀黄庄	4	19874

表 5 不同时间段各线路的进站客流分布

人数	4:00:00-5:00:00	5:00:00-6:00:00	6:00:00-7:00:00	7:00:00-8:00:00	8:00:00-9:00:00	9:00:00-10:00:00	10:00:00-11:00:00	11:00:00-12:00:00

1 号线	79	2005	9592	24463	20977	9686	8146	8503
2 号线	122	2278	7599	21530	24780	12434	9959	9560
3 号线	0	1264	6352	12568	9033	4359	2566	2148
4 号线	41	3559	20508	52838	40797	19135	13921	13088
5 号线	110	2247	10402	28708	26020	11461	6609	5599
6 号线	8	1505	8505	23369	17942	8662	5346	4835
7 号线	2	646	3532	11378	11029	4745	3074	2663
8 号线	79	804	4576	18506	17278	6738	2829	2513
9 号线	4	924	4089	11321	11090	5310	3791	3664
10 号线	63	2112	12158	35895	32725	16420	11654	11117
11 号线	0	0	0	0	0	0	1	0
12 号线	0	618	2872	5635	3611	1892	1294	1260
13 号线	21	1075	8084	29342	32510	14520	5766	4802
14 号线	0	398	2518	6453	4536	1932	1301	1292
15 号线	2	1145	6578	14009	11124	4392	2775	2568
18 号线	9	684	7591	22371	16933	6392	2506	2210
19 号线	7	823	5019	11214	6234	1799	1304	1102

表 6 不同时间段各线路的出站客流分布

人数	4:00-5:00	5:00-6:00	6:00-7:00	7:00-8:00	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-13:00
1 号线	0	0	0	1	0	0	0	0	0
2 号线	4	1110	9237	44771	92679	45596	21127	16497	8055
3 号线	0	56	1240	4746	8416	5386	3717	3424	2380
4 号线	1	586	10699	45525	85338	47926	26722	22137	12947
5 号线	0	0	0	0	0	0	0	0	0
6 号线	0	1	35	268	516	325	112	112	45
7 号线	1	97	1570	8766	22012	10166	5159	4083	2385
8 号线	14	78	1332	9343	19179	9466	4647	3768	2548
9 号线	0	128	3270	14471	28610	11952	6635	5916	3591
10 号线	16	330	4175	18167	33053	17715	10831	9274	5604
11 号线	0	0	0	0	0	0	0	0	0
12 号线	0	0	0	0	0	0	0	0	0
13 号线	0	241	3822	21384	49218	27818	9935	7222	4559
14 号线	0	0	0	0	0	0	0	0	0
15 号线	0	85	1589	9142	20145	10297	4394	3521	2449
18 号线	1	26	1120	9368	21758	13482	4357	2086	1519
19 号线	4	38	519	2681	3558	1729	1515	1427	1282

附录 2：求解问题二的 MATLAB 程序

附录 2-1：乘客出行信息准确还原模型程序

1. all_paths_gen.m

```
function [] = all_paths_gen(weight_matrix, k, output_file)
len = length(weight_matrix);
```

```
fileID = fopen(output_file,'wt');
fprintf(fileID,'%d nodes \n\n', len);
fclose(fileID);
node_pairs = combnk(1:len,2);
for index = 1: length(node_pairs)
   src = node_pairs(index, 1);
   dst = node_pairs(index, 2);
   gen_k_shortest_path(weight_matrix, src, dst, k, output_file, index);
end
```

2. dijkstra.m

```
function [shortestPath, totalCost] = dijkstra(netCostMatrix, s, d)
n = size(netCostMatrix,1);
for i = 1:n
   % initialize the farthest node to be itself;
   farthestPrevHop(i) = i; % used to compute the RTS/CTS range;
   farthestNextHop(i) = i;
end
% all the nodes are un-visited;
visited(1:n) = false;
distance(1:n) = inf;     % it stores the shortest distance between each node and the source node;
parent(1:n) = 0;
distance(s) = 0;
for i = 1:(n-1),
   temp = [];
   for h = 1:n,
      if ~visited(h)   % in the tree;
         temp=[temp distance(h)];
      else
         temp=[temp inf];
      end
    end;
    [t, u] = min(temp);       % it starts from node with the shortest distance to the source;
    visited(u) = true;         % mark it as visited;
    for v = 1:n,               % for each neighbors of node u;
      if ( ( netCostMatrix(u, v) + distance(u)) < distance(v) )
         distance(v) = distance(u) + netCostMatrix(u, v);  % update the shortest distance when a
shorter shortestPath is found;
         parent(v) = u;      % update its parent;
      end;
    end;
end;
shortestPath = [];
if parent(d) ~= 0    % if there is a shortestPath!
   t = d;
   shortestPath = [d];
   while t ~= s
```

```
            p = parent(t);
            shortestPath = [p shortestPath];
            if netCostMatrix(t, farthestPrevHop(t)) < netCostMatrix(t, p)
               farthestPrevHop(t) = p;
            end;
            if netCostMatrix(p, farthestNextHop(p)) < netCostMatrix(p, t)
               farthestNextHop(p) = t;
            end;
            t = p;
         end;
      end;
      totalCost = distance(d);
```

3. gen_k_shortest_path.m

```
      function [shortestPaths, totalCosts] = gen_k_shortest_path(weight_matrix, src, dst, k, output_file, demand_id)
      fileID = fopen(output_file,'at');
      %------Call kShortestPath------:
      [shortestPaths, totalCosts] = kShortestPath(weight_matrix, src, dst, k);
      %------Display results------:
      if isempty(shortestPaths)
         fprintf(fileID,'No path available between these nodes\n\n');
      else
         for i = 1: length(shortestPaths)
            %fprintf(fileID,'Path # %d: %d %d : ',i, src, dst);
            fprintf(fileID,'%d %d ', src, dst);
            %disp(shortestPaths{i});
            len = length(shortestPaths{i});
            for j = 1: len - 1
               fprintf(fileID, '%d ', shortestPaths{i}(j));
            end
            fprintf(fileID,'%d', shortestPaths{i}(len));
            %fprintf(fileID,'x_%d_%d \n', demand_id, i);
            fprintf(fileID,'\n');
            %fprintf(fileID,'Cost of path %d is %5.2f\n\n',i,totalCosts(i));
         end
      end

      fclose(fileID);
      function [shortestPaths, totalCosts] = kShortestPath(netCostMatrix, source, destination, k_paths)
      if source > size(netCostMatrix,1)
         destination > size(netCostMatrix,1)
         warning('The source or destination node are not part of netCostMatrix');
         shortestPaths=[];
         totalCosts=[];
      else
```

```
k=1;
[path cost] = dijkstra(netCostMatrix, source, destination);
%P is a cell array that holds all the paths found so far:
if isempty(path)
   shortestPaths=[];
   totalCosts=[];
else
   path_number = 1;
   P{path_number,1} = path; P{path_number,2} = cost;
   current_P = path_number;
   %X is a cell array of a subset of P (used by Yen's algorithm below):
   size_X=1;
   X{size_X} = {path_number; path; cost};
   %S path_number x 1
   S(path_number) = path(1); viation vertex is the first node initially

   % K = 1 is the shortest path returned by dijkstra():
   shortestPaths{k} = path ;
   totalCosts(k) = cost;
   while (k < k_paths    &&    size_X ~= 0 )
      %remove P from X
      for i=1:length(X)
         if   X{i}{1} == current_P
            size_X = size_X - 1;
            X(i) = [];lete cell
            break;
         end
      end
      P_ = P{current_P,1}; %P_ is current P, just to make is easier for the notations
      %Find w in (P_,w) in set S, w was the dev vertex used to found P_
      w = S(current_P);
      for i = 1: length(P_)
         if w == P_(i)
            w_index_in_path = i;
         end
      end

      for index_dev_vertex= w_index_in_path: length(P_) - 1    %index_dev_vertex is
index in P_ of deviation vertex
         temp_netCostMatrix = netCostMatrix;
         %------
         %Remove vertices in P before index_dev_vertex and there incident edges
         for i = 1: index_dev_vertex-1
            v = P_(i);
            temp_netCostMatrix(v,:)=inf;
            temp_netCostMatrix(:,v)=inf;
```

```
            end
            %------
            %remove incident edge of v if v is in shortestPaths (K) U P_  with similar sub_path to P_....
            SP_sameSubPath=[];
            index =1;
            SP_sameSubPath{index}=P_;
            for i = 1: length(shortestPaths)
               if length(shortestPaths{i}) >= index_dev_vertex
                  if P_(1:index_dev_vertex) == shortestPaths{i}(1:index_dev_vertex)
                     index = index+1;
                     SP_sameSubPath{index}=shortestPaths{i};
                  end
               end
            end
            v_ = P_(index_dev_vertex);
            for j = 1: length(SP_sameSubPath)
               next = SP_sameSubPath{j}(index_dev_vertex+1);
               temp_netCostMatrix(v_,next)=inf;
            end
            %------

            %get the cost of the sub path before deviation vertex v
            sub_P = P_(1:index_dev_vertex);
            cost_sub_P=0;
            for i = 1: length(sub_P)-1
               cost_sub_P = cost_sub_P + netCostMatrix(sub_P(i),sub_P(i+1));
            end
            [dev_p  c]  =  dijkstra(temp_netCostMatrix,  P_(index_dev_vertex), destination);
            if ~isempty(dev_p)
               path_number = path_number + 1;
               P{path_number,1} = [sub_P(1:end-1) dev_p] ;  %concatenate sub path- to -vertex -to- destination
               P{path_number,2} =  cost_sub_P + c ;

               S(path_number) = P_(index_dev_vertex);

               size_X = size_X + 1;
               X{size_X} = {path_number;  P{path_number,1} ;P{path_number,2} };
            else
               %warning('k=%d, isempty(p)==true!\n',k);
            end
         end
         %Step necessary otherwise if k is bigger than number of possible paths
         %the last results will get repeated !
```

```
        if size_X > 0
           shortestXCost= X{1}{3};   %cost of path
           shortestX= X{1}{1};        %ref number of path
           for i = 2 : size_X
              if   X{i}{3} < shortestXCost
                 shortestX= X{i}{1};
                 shortestXCost= X{i}{3};
              end
           end
           current_P = shortestX;
           %******
           k = k+1;
           shortestPaths{k} = P{current_P,1};
           totalCosts(k) = P{current_P,2};
           %******
        else
           %k = k+1;
        end
     end
  end
end
```

附录 2-2：乘客最优路径选择模型程序

```
function [mydistance,mypath]=mydijkstra(a,sb,db)
% 输入：a—邻接矩阵(aij)是指 i 到 j 之间的时间，可以是有向的
% sb—起点的标号, db—终点的标号
% 输出：mydistance—最短路径的时间, mypath—最短路的路径
n=size(a,1); visited(1:n) = 0;
distance(1:n) = inf; % 保存起点到各顶点的最短时间
distance(sb) = 0; parent(1:n) = 0;
for i = 1: n-1
 temp=distance;
 id1= visited==1; %查找已经标号的点
 temp(id1)=inf; %已标号点的距离换成无穷
 [t, u] = min(temp); %找标号值最小的顶点
 visited(u) = 1; %标记已经标号的顶点
 id2=find(visited==0); %查找未标号的顶点
 for v = id2
 if a(u, v) + distance(u) < distance(v)
 distance(v) = distance(u) + a(u, v); %修改标号值
 parent(v) = u;
 end
 end
end
mypath = [];
if parent(db) ~= 0 %如果存在路!
```

```
  t = db; mypath =[db];
  while t ~= sb
  p = parent(t);
  mypath = [p mypath];
  t = p;
  end
end
mydistance = distance(db);
return
```

附录 3：求解问题三的 MATLAB 程序

附录 3.1: 优化模型 matlab 程序

```
clc,clear
D=[641    866 838 573
555 828 736 488
116 157 150 128
627 738 450 361
806 816 566 475
520 736 495 488
243 263 208 159
589 722 601 495
734 736 640 434
352 423 358 221
776 787 645 481
383 440 329 188
1067       955 875 550
];
[m,n]=size(D);
[s,y]=fmincon('fun1',rand(m,n),[],[],[],[],zeros(m,n),[],'fun2')

function f=fun1(s);
D=[641    866 838 573
555 828 736 488
116 157 150 128
627 738 450 361
806 816 566 475
520 736 495 488
243 263 208 159
589 722 601 495
734 736 640 434
352 423 358 221
776 787 645 481
383 440 329 188
1067       955 875 550
];
```

```
f=sum((sum(D)-sum(s([1:end-1],:)))/s(end,:));

function    [g,h]=fun2(s)
D=[641    866 838 573
555 828 736 488
116 157 150 128
627 738 450 361
806 816 566 475
520 736 495 488
243 263 208 159
589 722 601 495
734 736 640 434
352 423 358 221
776 787 645 481
383 440 329 188
1067        955 875 550
];
[m,n]=size(D);
T=3;
H=30;
r=0.16072;
n_1=H/T;
%%
Q=zeros(m,n);
Q(:,1)=1428;
Q(:,[2:end])=1428-r*sum(sum(s(:,[2:end])));
% for j=2:n
% Q(:,j)=1428-r*sum(sum(s(:,2:j)));
% end
%%
lamda=zeros(m,n);
lamda(1,:)=D(1,:)/H;
lamda([2:end],:)=(D([2:end],:)+(D([1:end-1],:)-s([1:end-1],:)*n_1))/H;
%%
P=zeros(m,n);
P(1,:)=lamda(1,:)*T/n;
P([2:end],:)=(P([1:end-1],:)+lamda([2:end],:)*T-s([2:end],:))/n;
%    for i=2:m
%          P(i,:)=(P(i-1,:)+lamda(i,:)*T-s(i,:))/n;
%    end
 %%
%    ub=zeros(m,n);
%    for i=1:m
%          for j=1:n
%                ub(i,j)=min(Q(i,j),P(i,j));
%          end
```

```
%    end
g=s-Q;
h=s-P;
%    f=sum((sum(D)-sum(s([1:end-1],:)))/s(end,:))
```

附录 3.2：改进的模拟退火算法 matlab 程序

```
clc,clear
load sj.txt %加载敌方 100 个目标的数据，数据按照表格中的位置保存在纯文本文件 sj.txt 中
x=sj(:,1:2:8);x=x(:);
y=sj(:,2:2:8);y=y(:);
sj=[x y]; d1=[70,40];
sj=[d1;sj;d1]; sj=sj*pi/180;
d=zeros(102); %距离矩阵 d
for i=1:101
 for j=i+1:102
 temp=acos(sj(i,1)-sj(j,1))*cos(sj(i,2))*cos(sj(j,2))+sin(sj(i,2))*sin(sj(j,2));
 d(i,j)=6370*acos(temp);
 end
end
d=d+d';
S0=[];Sum=inf;
rand('state',sum(clock));
for j=1:1000
 S=[1 1+randperm(100),102];
 temp=0;
 for i=1:101
 temp=temp+d(S(i),S(i+1));
 end
 if temp<Sum
 S0=S;
 Sum=temp;
 end
end
e=0.1^30;L=20000;at=0.999;T=1;
%退火过程
for k=1:L
 %产生新解
c=2+floor(100*rand(1,2));
c=sort(c);
c1=c(1);c2=c(2);
 %计算代价函数值
 df=d(S0(c1-1),S0(c2))+d(S0(c1),S0(c2+1))-d(S0(c1-1),S0(c1))-d(S0(c2),S0(c2+1));
 %接受准则
 if df<0
 S0=[S0(1:c1-1),S0(c2:-1:c1),S0(c2+1:102)];
```

```
  Sum=Sum+df;
  elseif exp(-df/T)>rand(1)
  S0=[S0(1:c1-1),S0(c2:-1:c1),S0(c2+1:102)];
  Sum=Sum+df;
  end
  T=T*at;
  if T<e
  break;
  end
end
  % 输出路径及路径长度
S0,Sum
```