公交车在高峰和平峰期间转换的调度

摘要

公交车是为市民出行提供服务的“准公共”产品。它服务的对象是公众而非特定的个人，也就是说，不考虑任何一个人在任何时候都能得到公交服务这样的要求。另一方面，公共交通不以盈利为目标但也不是免费的。因此，公交要在给定的财政拨款约束下，兼顾“尽可能减少私家车使用以缓解城市交通拥堵”和“尽量让公众满意”两大目标。公交车的站点布局、线路规划、车型配备、票价制定、发车频率和车辆调度等都要按照它的基本属性和目标来进行设计和调整的。

在问题一，我们根据网上寻找到的一个时间段对应的公交车上人数上下行的数据，获得对应时间段上下公交车的人数，以及每个站点之间的距离，并可知本公交线路一共 14 个站点，从初始站到终点站的距离为 14.58km。之后我们又搜集相关的论文，给出了公交线路高峰和平峰的定义。

在问题二，我们采用线性规划的方法，总体的目标函数以乘客的等待时间和公交车的满载率的加权和作为指标，旨在使得乘客的等待时间和拥挤程度达到一个平衡。最终我们求得高峰时段发车间隔为 2 分钟左右，平峰时段为 6-7 分钟，同时我们选取一步到位的方法对公交车数量进行调控，因为公交车发车间隔均较短，在各时间段保持相同的发车间隔更会提高效率。

在问题三，我们选用线性回归的方法进行建模，利用现有数据预测在新的环境，日期下可能出现的“高峰”，“平峰”，误差大概在 0.123 左右。我们先对数据进行预处理，对其归一化，然后在进行线性回归的建模，并利用梯度下降法进行优化，寻找最优解。

在问题四，我们利用实际的数据来验证结果，在预测得到较好的公交“高峰”和“平峰”情况下，获得了好的调度效果。

关键词　多目标规划　　线性回归

1 问题重述

1.1 问题背景

公交车是为市民出行服务的。市民出行并不是“均匀”的，有乘客多的时段（称为“高峰期”）和乘客不太多的时段（称为“平峰”期），起讫点也不尽相同。容易理解，高峰期公交车发车频率高，投入的运营车辆也多，以满足乘客的需要；而平峰期则要相应地减下来，以节约成本开支。我们想来关注一下这方面的“调度问题”。

1.2 需解决的问题

问题一：给出一条公交线路“高峰”和“平峰”的定义，并说明其合理性。

问题二：对高峰和平峰任意给定的一组数据，给出“转换期”的调度方案，并说明在什么指标。下，该方案是可行的、最优的。进一步，讨论调度方案对参数的稳健性和敏感性。

问题三：给出“高峰”和“平峰”的预测方法，并试通过实际运行数据验证你的结果。

1.3 问题的相关分析以及注记

“高峰期”和“平峰期”的划分并不是绝对的，完全可以由“决策者”的价值标准来定，但如何划分将直接关系到后面的工作。其次，我们注意到，公交车一旦从起点站出发，就必须驶完一个单程而不能在中途停运。因此从高峰期到平峰期把运营的公交车数量“减下来”是需要一个过程的，很难“立竿见影”。

同时，高峰到平峰的过程显然与驶完一个单程所需的时间，或等价地，与线路的长短有关，因此要思考“分步减下来”还是“一步到位”。特别，当乘客数“急剧变化”，从高峰到平峰又到高峰的转换来得非常快，以至于“减下来”还来不及见效就要马上“恢复”，就可能“得不偿失”。

2 模型假设与符号说明

2.1 模型的假设

2.1.1 题目假设

（1）所考虑的公交线路是确定的，这意味着它的长度、站点、单程耗时及运营成本都已经定了，一票制的票价也是定的且不考虑乘客的差异和优惠。都不需要去另做假设。

（2）为了不把问题弄得太复杂，就像我们无法考虑每一个乘客的行为和愿望一样，我们也只考虑确定的某一条公交线路。尽管公交线路是一个网，两条公交线路可能有部分重合，但我们也不考虑考虑它们之间的“替代”和“竞争”。

（3）调度的目标是使得在乘客数下降和恢复的过程中，通过相应地减少和恢复投入运营的车辆数量来保证这个“阈值”（也就是“下界”）不被突破，当然（“盈利”的）“上界”是不受限的。

2.1.2 新增假设

（1）在每个小时内所有乘客平均分布在路线上的所有站点上。

（2）对于某一站点的所有乘客，到达该站点的时刻服从均匀分布。即他们的平均等待时间为此站点前后两辆车之间的发车间隔的一半。

（3）当公交车到达某一站点时，我们通过发车间隔的调整，保证在此站点上等待的市民能全部上车。即不考虑公交车满载继续等待下一辆的情况。

（4）以公交车发车时所在的某个小时内乘客总数，作为该公交车单次路线上的乘客上下车情况，每个小时间情况独立。

2.2 符号说明

符号说明

符号	符号说明	单位
t_i	在第 i 小时内的发车间隔	min
$S_{i,j}$	第 i 小时内以 t_i 间隔发车，到达第 j 个车站前车上的人数	人
$N_{i,j}$	在第 i 小时内在第 j 个车站下车的总人数	人
$M_{i,j}$	在第 i 小时内到达第 j 个车站的市民总人数	人
q_m	公交车的最大载客量	人
W_m	每小时线路上的最小收益	元
$F(t_i)$	在第 i 小时内发车的公交车上乘客的等待时间	min
$f(t_i)$	在第 i 小时内发车的公交车上的拥挤程度	/
C	每辆公交车单次行程的运营费用	元
P	公交车的票价	元
n	公交线路上的站点总数	个

3 问题一的建模求解

3.1 数据的描述与分析

我们从网上获得到深圳某路公交车各个站各个时间段上下车的人数，将人数大致进行处理，得到各个时间段上下车人数粗略的统计柱形图。

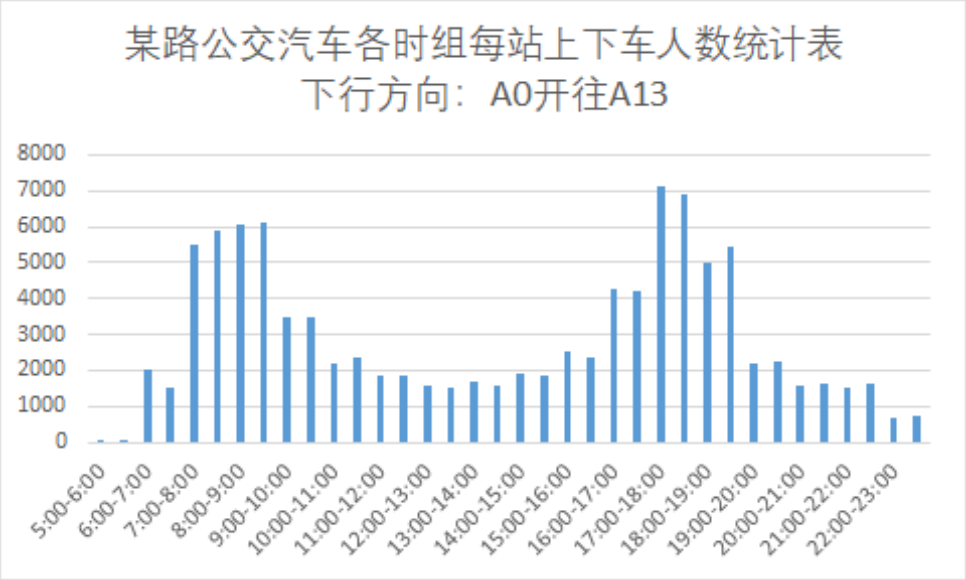

图 1: 一天中各时段人数

接着我们为了更直观的表示，将数据进一步处理后，并用折线图表示，

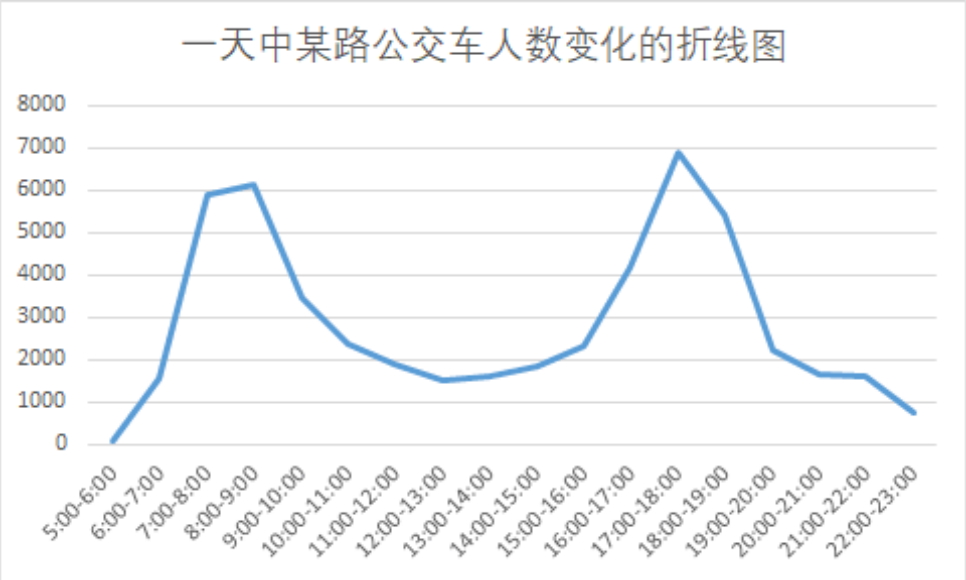

图 2: 一天中各时段人数 (折线图)

3.2 公交车高峰以及平峰的定义

3.2.1 高峰以及平峰的定义

根据相关论文的知识以及相关生活常识，我们可以大致认定在早上上班时间以及晚上下班时间，车内人数较多，为公交线路的“高峰”，其余时间为公交线路的“平峰”。考虑不同城市的对应生活习惯，以及公交线路所在地区的差异，我们对公交线路“高峰”具体时间的划分则根据数据来判断，我们暂定人数超过 3600 即为高峰时期。

3.2.2 合理性解释

接着我们对公交车的“高峰”以及“平峰”阶段进行合理性的解释，公交车的“高峰”，也就是正常时段，是指运营总收入与运营总成本不小于某一个给的的“阈值”的时段，我们可知公交单程的行驶路程 $s = 14.61km$，根据网上资料所得，公交车百公里油耗为 30-35 升，也就是 1 公里 0.35 升左右，假设柴油价格为 6 元每升，单程行驶的成本

$$C_1 = s \times 0.35 \times 6 = 30$$

在深圳，我们设司机一个月的工资为 9000 元，平均每天 300 元，假设一天可以一辆车可以跑 20 个单趟，那么司机的成本为 15 元，我们暂时不考虑车的价格以及车辆维护的成本。

$$C = C_1 + C_2 = 30 + 9000 \div 30 \div 20 = 45$$

所以，最终一辆公交车行驶单趟的总成本 C 为 45 元。

我们假设，公交车所有乘客的票价均为 1 元，一辆公交车 50 人的时候满座，高峰时段为车内无空座的时期。考虑到单躺车次中乘客有上有下，我们设一趟车所载人数为 60 以上时，此公交车处在正常时段，即高峰时期。

$$w = 60 \times 1 - 45 = 15$$

本车的收益为 15 元，同时我们设阈值为 15。

每趟车人数大于 60 时，即 1 小时的时间段人数大于 3600，所以人数大于 3600 的时间段定为高峰时期，人数小于 3600 的时期为低峰时期。

4 问题二的建模

4.1 对调度任务的简要分析

本问题，我们考虑的是每个车的利润问题，当一个车内人数不足时，公交车所带来的利润则会小于阈值。而我们调度的分析即在高峰期增加车辆，在平峰期削减车辆，以达到车内人数满足的需求，从而使利润满足阈值。

同时，在确保利润的同时，我们也要尽可能保证乘客的满意度，如果乘客长时间等不到车，或者是一辆车中乘客过多，都会导致乘客的满意度下降。

通过问题一的定义，我们得出，对于深圳某路公交车数据，某一个典型工作日高峰期为早上 7：00-10：00（9：00-10：00 这一时间段的人数接近 3600，为使得实验结果明显，归为高峰期）和晚上 16：00-19：00，平峰期为 9：00-16：00。为此，我们通过这一具有典型意义的数据进行说明我们的调度模型。

为了符合题目中“尽量让公众满意”的目标，我们可以从常使中定义，对于乘公交车这一事件，等待的时间越短，公交车上的拥挤程度越低，乘客的满意度越高，即发车间隔越短，满意度会越高。但又要考虑到公交公司的财政约束，需要保证一定的收益，即要考虑发车间隔不能过短，导致公交系统运营成本过高。因此，需要综合考量这三个互相约束的目标。

4.2 对调度任务的建模

针对问题二我们选择用线性规划对上述问题进行建模以及求解

$$
\begin{aligned}
min \quad & \alpha F(t_i) + \beta f(t_i) && (1)\\
s.t. \quad & S_{i,j}(t_i) = S_{i,j-1}(t_i) + \frac{M_{i,j}}{60} \times t_i - \frac{N_{i,j}}{60} \times t_i && (2)\\
& S_{i,j}(t_i) \leq q_m && (3)\\
& F(t_i) = \sum_{j=1}^{n} \frac{M_{i,j}}{60} \times t_i \times \frac{t_i}{2} && (4)\\
& f(t_i) = \sum_{j=1}^{n} \frac{S_{i,j}}{q_m} \times \frac{S_{i,j}}{\sum_{j=1}^{n} S_{i,j}} && (5)\\
& W(t_i) = \sum_{j=1}^{n} M_{i,j} \times P - C && (6)\\
& W(t_i) \geq Wm && (7)\\
& && (8)
\end{aligned}
$$

其中 α 与 β 均为参数，代表着我们对乘客的拥挤程度以及等待时间的权重分配, 也是我们需要学习的参数。总体的目标函数以乘客的等待时间和公交车的满载率的加权和作为指标，旨在使得乘客的等待时间和拥挤程度达到一个平衡。

在约束条件中，(2)(3) 为车辆总载客量的限制。对于每一辆公交车，道道每一个站点前后的总人数都应该少于最大载客量，这是直观的。(4) 为每辆车总的等待时间，按照假设，乘客到达任一站点的时刻服从平均分布，可以证明，同一站点在前后两辆公交车间隔中到达的乘客的等待时间的期望为前后两辆公交车发车间隔的一半。因此，按照每个站的乘客总数对发车间隔的一半进行加权和是合理的。(5) 为每辆车在线路上总的拥挤程度，即每段路程车上的总人数对当时的拥挤程度进行加权和。(6)(7) 为公交车运行的成本限制。虽然公交公司不以盈利为目的，但需要一定的收益保证整个公交系统的运营，因此每辆公交车运营的收益需要达到一个阈值 W_m。

4.3 对调度问题模型的求解

针对上述某一线城市典型公交线路的具体数据分析，并在网上收集到符合实际的具体参数值。对于一辆城市公交，最大载客量一般为 100，即 $q_m = 100$。考虑到公交车的便利性，一般公交车票价为 2 元，即 $P = 2$。上述公交车共 14 个站，即 $n = 14$，而通过问题一中的计算，我们估计出一辆公交车运营的成本为 45 元，即 $C = 45$。

我们通过 python 进行仿真模拟，以 1min 为最小单位，对每个小时进行调优，最终得出了每个小时以市民乘客等待时间和车辆拥挤程度指标下的最优发车间隔（调度方案）。

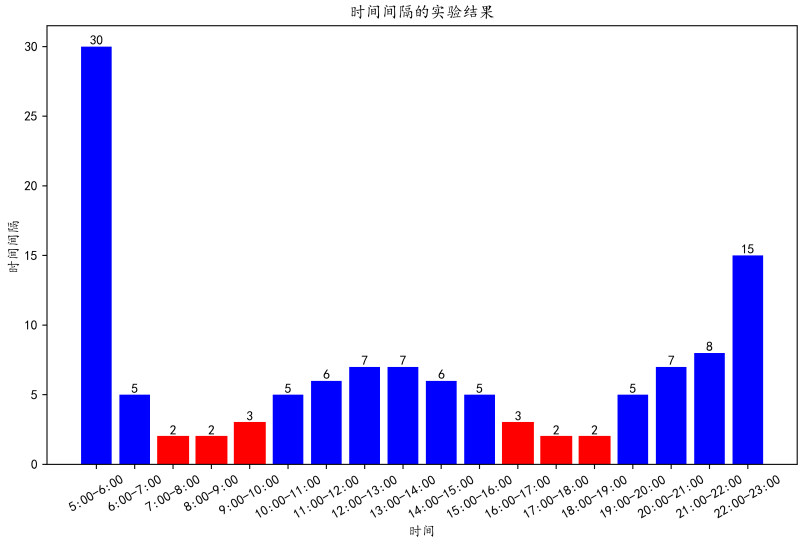

图 3: 时间间隔的实验结果

可以从每个小时内的最佳时间间隔看出，对于高峰期的发车间隔，保持在 2-3min 为最佳，这一结果也符合生活中的常识。

以下通过多个角度进行分析结果的合理性：

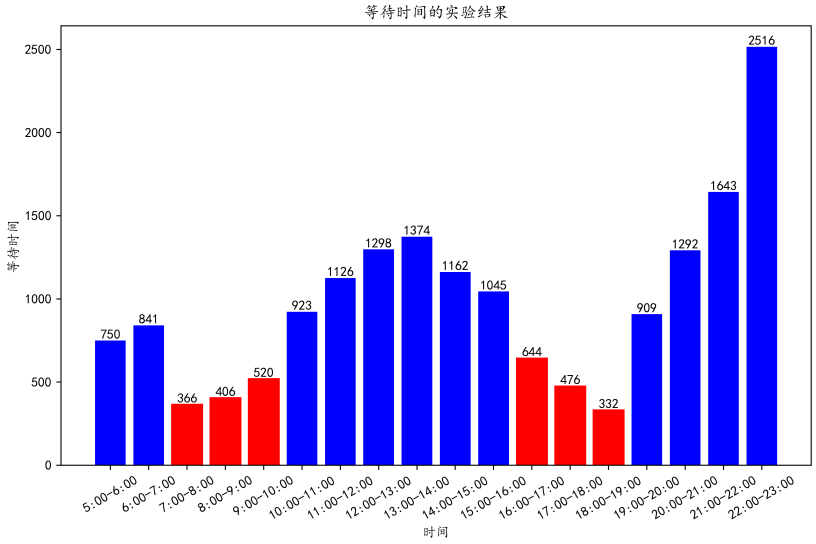

图 4: 等待时间的实验结果

对于高峰期，发车间隔时间短，因此乘客等待的时间相应较短。

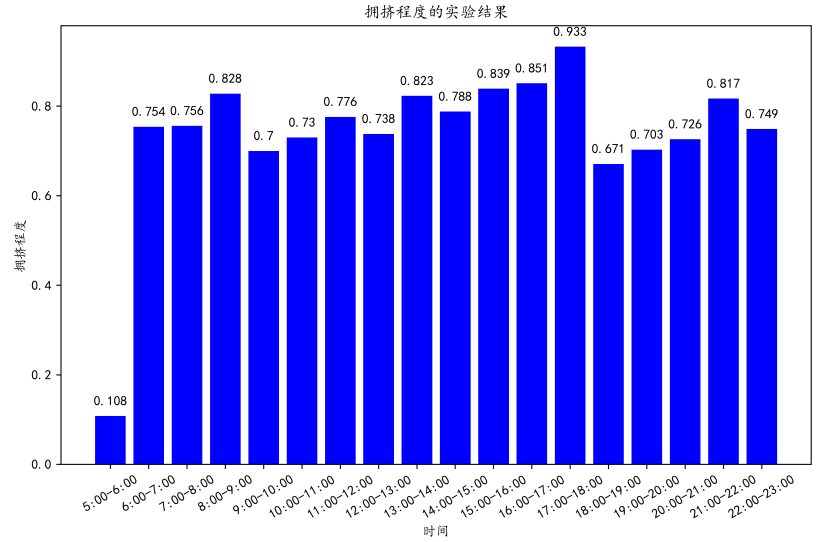

图 5: 拥挤程度的实验结果

通过调节发车间隔时间，可以保证公交车的拥挤程度（即满载率）在 0.67-0.94（5：00-6：00 因乘客数量过少，不具有代表性）之间，使得公交车空间得到充分利用。

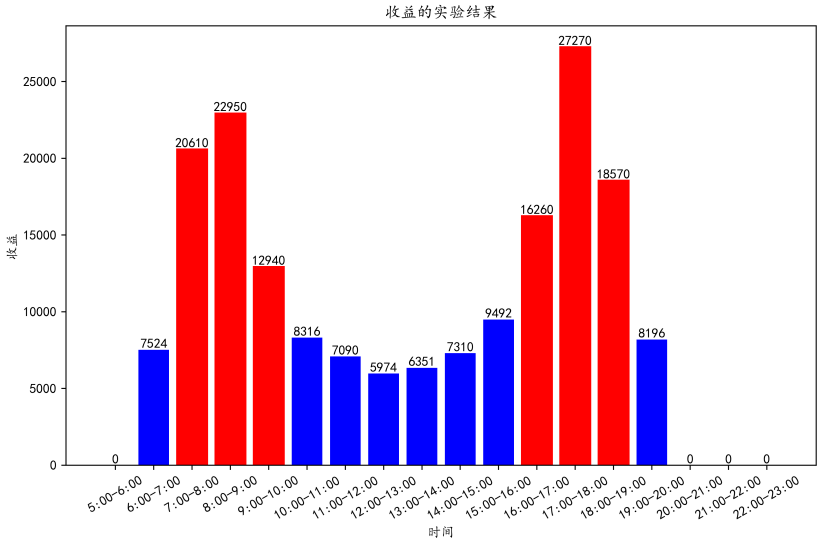

图 6: 收益的实验结果

对于高峰期的时间段，虽然发车数量较多，但因乘客较多，使得收益比平峰期更高，这也符合题目中以收益的阈值作为高峰期和平峰期定义的要求。

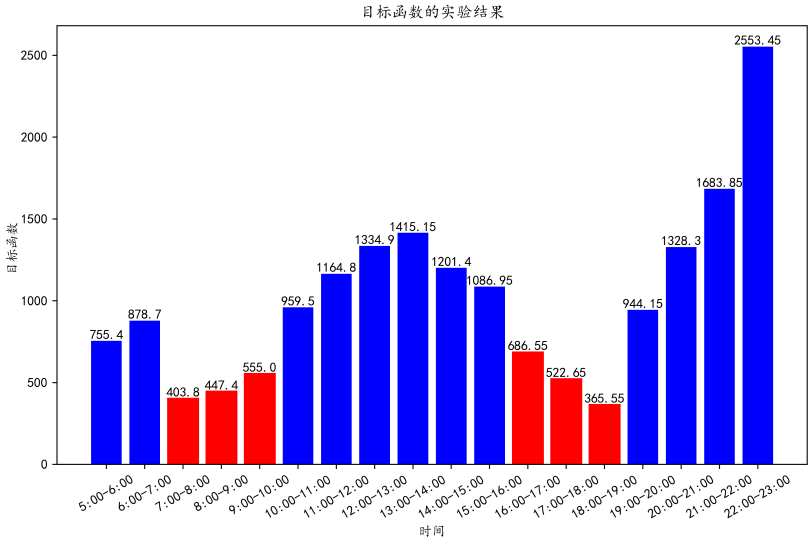

图 7: 目标函数的实验结果

对于目标函数中的参数, 分别取值 $\alpha = 1$ 和 $\beta = 50$，求得目标函数的值，为所有时间段内最低的部分，证明了我们的模型在很大程度上是符合实际情况的。

4.4 灵敏度分析

之后我们对参数进行了灵敏度分析，进一步讨论调度方案对参数的稳健性和灵敏性。

调度方案受人数的影响，同时也受高峰和平峰的影响，而人数以及车次同样也影响着，乘客满意度以及等待时间等因素。在人数小幅度变化，即高峰期 1 小时内人数上下浮动 495 之内时，平峰期一小时人数上下浮动 214 之内时，我们认为调度方案仍然保持稳定。

同时可以分析得出，公交车的拥挤程度保持一个平均水平，且都离上限有一定的距离，对调度方案影响较小。平峰时期，车次间隔时间过长，如果因为举办活动或者天气因素等不可预知因素影响，而导致人数剧增，调度方案更容易受到影响。

5 问题三的建模

5.1 问题的简单分析

预测公交车的“高峰”和“低峰”的方法有很多：一天 24 小时，工作日，非工作日，节假日，人们的出行方式，天气等因素。同时我们也要考虑到深圳市本身的经济，人民的生活方式，并结合我们已有的数据。

根据深圳 2018 年运行交通报告，深圳公共交通服务水平持续提升，早晚高峰公共交通出行分担率已超 60%，常规公交在公共交通客运总量的比重同比下降 3.2 个百分点。但常规公交站点 500 米覆盖率已增长至 95.8%，常规公交线路共 981 条，运营长度 2.1 万公里，市民乘搭公交变得更为方便。

我们选用曾经天池比赛的公交数据，本数据有广州市内及广佛同城公交线路的历史公交刷卡数据，我们挖掘固定人群在公共交通中的行为模式。建立公交线路乘车人次预测模型，从而在一定情况下预测特定时期的人数，“平峰”以及“高峰”。

首先我们选取其中的一条线路，对该条线路的一周之内的客流人数进行可视化，

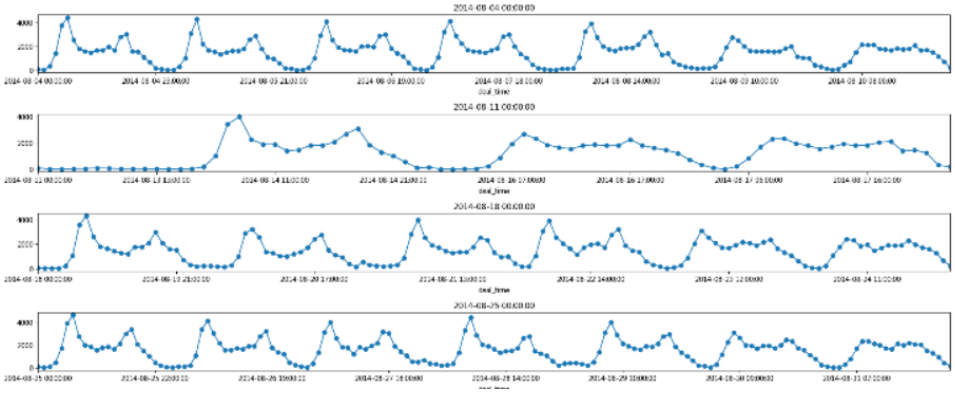

图 8: 一天中各时段人数 (折线图)

之后我们先对天气假期等因素进行定性分析，这里选取天气中气温的因素进行展示，

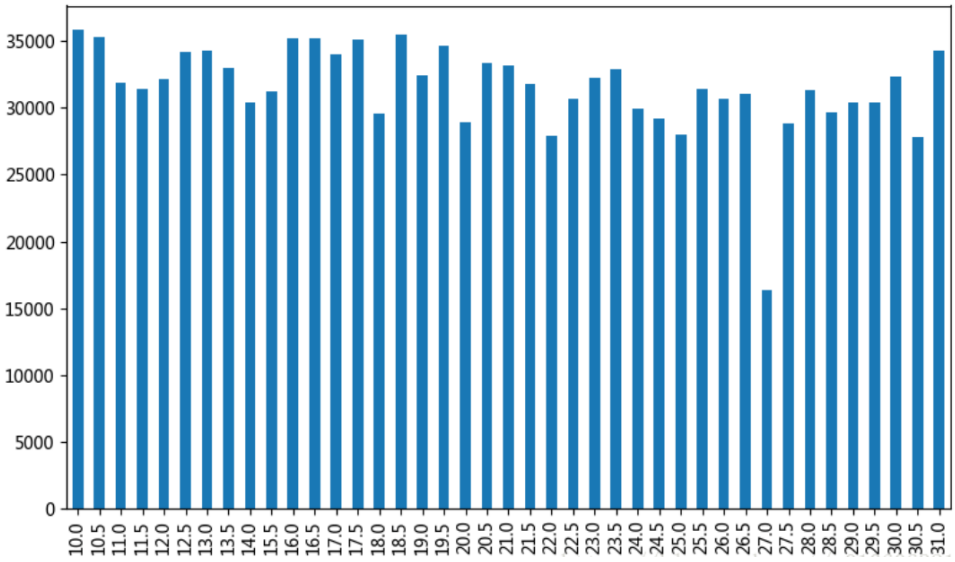

图 9: 不同气温下的人数

之后计算气温因素与人数的 Pearson 相关系数

$$\rho_{xy} = \frac{cov(XY)}{\sigma_x \sigma_y} \tag{9}$$

最终求得 Pearson 的相关系数为-0.456，两者呈负相关，人们在气温比较低的时候喜欢坐公交，从而可能推测气温低时高峰时间可能会更长，而且高峰时人数会更多。

5.2 问题的建模与求解

本题我们选用线性回归的方法对问题进行建模，我们把工作日，非工作日，节假日，人们的出行方式，天气因素等当作参数变量，并将其在已有的数据中进行训练，通过训练获得相关的参数信息，考虑不同的情况对高峰和平峰时段的影响。

选出特定的线路，我们采用交叉验证的方法进行数据集的划分，将其中 90% 的天数的数据设为训练集，其余部分设为验证集。

同时我们对各维度的数据进行 feature scaling，减小优化过程的波动。归一化的公式如下：

$$x = \frac{x - \mu}{s} \tag{10}$$

其中 x 为原数据，μ 为 x 的均值，s 为数据的最大值减去最小值。

在对数据进行预处理，我们开始正式建模，设高峰时间段为 y，列出公式为

$$y = \theta^T x \tag{11}$$

$$\vec{\theta} = (\theta_1, \theta_2, ..., \theta_n) \tag{12}$$

列出公式用已有数据进行拟合，求得 $\vec{\theta}$ 的值，在优化的过程中我们选择用随机梯度下降的算法。

线性回归中的损失函数：

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y) \tag{13}$$

其梯度为

$$\nabla_\theta J(\theta) = X^T X\theta - X^T y \tag{14}$$

因此使用梯度下降的迭代方程为

$$\begin{aligned} \theta^{t+1} &= \theta^t - \alpha\nabla_\theta J(\theta^t) \\ &= \theta^t + \alpha X^T(y - X\theta^T) \end{aligned} \tag{15}$$

我们认为人数在 500 人以内即为无误差，最终将优化后的模型对测试集进行预测，得到误差为 0.123938。

6 问题四的建模

本题要求在实际数据上运行所得结果，因为我们在第二问已经通过实际数据进行建模，现在再次展示各时段公交车发车间隔，并表示一天之内公交车运行的具体规划。

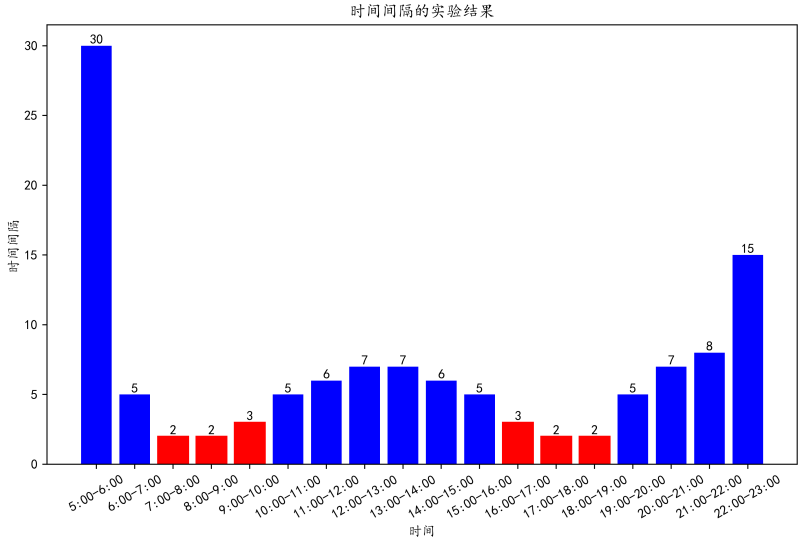

图 10: 调度方案（时间间隔）

附录

6.1 问题二：

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

s_data=pd.read_excel("深圳通刷卡数据.xlsx",sheet_name=1,index_col=False,index_row=False)
x_data=pd.read_excel("深圳通刷卡数据.xlsx",sheet_name=2)
ss_data=list(s_data['总数'][1:37:2])
sx_data=list(s_data['总数'][2:37:2])
xs_data=list(x_data['总数'][1:37:2])
xx_data=list(x_data['总数'][2:37:2])

#解决中文显示问题
plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False # 解决保存图像是负号'-'显示为方块的问题
plt.figure(figsize=(15,6))
time=["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","13:00-14:00","14:00-15:00",
"15:00-16:00","16:00-17:00","17:00-18:00","18:00-19:00","19:00-20:00","20:00-21:00","21:00-22:00
                                                    ","22:00-23:00"]
#plt.plot(time,ss_data,color='red',label="上行上客")
#plt.plot(time,sx_data,color='green',label="上行下客")
plt.plot(time,xs_data,color='red',label="下行上客",linestyle='-.')
plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
plt.axhline(y=3600,ls="-",c="b")#添加水平直线
plt.legend()
plt.xticks(rotation=30)
plt.savefig('./公交线路上下客人数.jpg',dpi=500)
plt.show()

ti_list=[0]*18
fi_list=[0]*18
Fi_list=[0]*18
Wi_list=[0]*18
score_list=[0]*18

k=1
ti=30

ti_list[k-1]=ti

qm=100
Wm=0
n=14
C=45
P=2

si=[]
si.append(0)
for j in range(1,15):
temp=int(round(si[j-1]+x_data.iloc[2*k-1,j]/60*ti-x_data.iloc[2*k,j]/60*ti))
if temp>qm:
print("!第",j,"站超出总人数! ")
si.append(temp)

fi=0
```

```
for j in range(1,15):
fi+=si[j]*si[j]/sum(si)/qm
fi=round(fi,3)

fi_list[k-1]=fi

Fi=[]
Fi.append(0)
for j in range(1,15):
Fi.append(x_data.iloc[2*k-1,j]/60*ti*ti/2)
Fi=int(round(sum(Fi)))

Fi_list[k-1]=Fi

count=0
for j in range(1,15):
count+=int(round(x_data.iloc[2*k-1,j]/60*ti))

Wi=count*P -C
if Wi < Wm:
print("!收益低于最小收益! ")
Wi_list[k-1]=Wi

print(" ")

score=50*fi+Fi

score_list[k-1]=score

print("一辆车总载客量: ",count)
print("以ti间隔发车的每站前车上的总人数: ",si)
print("以ti间隔发车的拥挤程度: ",fi)
print("以ti间隔发车的等车时间: ",Fi,"min")
print("以ti间隔发车的利润: ",Wi,"元")
print("以ti间隔发车的目标函数: ",score)
print(fi_list)
print(Fi_list)
print(Wi_list)
print(score_list)
print(ti_list)

plt.figure(figsize=(10,6))
bar=plt.bar(time,score_list,color='b',label="时间间隔")
for x,y in zip(time,score_list):
plt.text(x,y+0.02,y, ha='center',va='bottom')

bar[2].set_color('r')
bar[3].set_color('r')
bar[4].set_color('r')
bar[11].set_color('r')
bar[12].set_color('r')
bar[13].set_color('r')

#plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
#plt.legend()
plt.xticks(rotation=30)
plt.xlabel("时间")
plt.ylabel("目标函数")
plt.title("目标函数的实验结果")
```

```
plt.savefig('./目标函数.png',dpi=500)
plt.show()

plt.figure(figsize=(10,6))
bar=plt.bar(time,ti_list,color='b',label="时间间隔")
for x,y in zip(time,ti_list):
plt.text(x,y+0.02,y, ha='center',va='bottom')

bar[2].set_color('r')
bar[3].set_color('r')
bar[4].set_color('r')
bar[11].set_color('r')
bar[12].set_color('r')
bar[13].set_color('r')

#plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
#plt.legend()
plt.xticks(rotation=30)
plt.xlabel("时间")
plt.ylabel("时间间隔")
plt.title("时间间隔的实验结果")
plt.savefig('./时间间隔.png',dpi=500)
plt.show()

plt.figure(figsize=(10,6))
bar=plt.bar(time,fi_list,color='b',label="时间间隔")
for x,y in zip(time,fi_list):
plt.text(x,y+0.02,y, ha='center',va='bottom')
'''
bar[2].set_color('r')
bar[3].set_color('r')
bar[4].set_color('r')
bar[11].set_color('r')
bar[12].set_color('r')
bar[13].set_color('r')
'''
#plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
#plt.legend()
plt.xticks(rotation=30)
plt.xlabel("时间")
plt.ylabel("拥挤程度")
plt.title("拥挤程度的实验结果")
plt.savefig('./拥挤程度.png',dpi=500)
plt.show()

plt.figure(figsize=(10,6))
bar=plt.bar(time,Fi_list,color='b',label="时间间隔")
for x,y in zip(time,Fi_list):
plt.text(x,y+0.02,y, ha='center',va='bottom')

bar[2].set_color('r')
bar[3].set_color('r')
bar[4].set_color('r')
bar[11].set_color('r')
bar[12].set_color('r')
bar[13].set_color('r')

#plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
#plt.legend()
```

```
plt.xticks(rotation=30)
plt.xlabel("时间")
plt.ylabel("等待时间")
plt.title("等待时间的实验结果")
plt.savefig('./等待时间.png',dpi=500)
plt.show()

all_Wi=[0]*18
print(all_Wi)
for j in range(1,15):
all_Wi[j]=int(60/ti_list[j]*Wi_list[j])
plt.figure(figsize=(10,6))
bar=plt.bar(time,all_Wi,color='b',label="时间间隔")
for x,y in zip(time,all_Wi):
plt.text(x,y+0.02,y, ha='center',va='bottom')

bar[2].set_color('r')
bar[3].set_color('r')
bar[4].set_color('r')
bar[11].set_color('r')
bar[12].set_color('r')
bar[13].set_color('r')

#plt.plot(time,xx_data,color='green',label="下行下客",linestyle='-.')
#plt.legend()
plt.xticks(rotation=30)
plt.xlabel("时间")
plt.ylabel("收益")
plt.title("收益的实验结果")
plt.savefig('./收益.png',dpi=500)
plt.show()
```

6.2 问题三：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

train_data_X_path = '../data/line6_train_data_dum_scale.csv'

train_data_X = pd.read_csv(train_data_X_path)

lables = train_data_X['card_id']

features = train_data_X.drop(['date', 'card_id'], axis=1)

linreg = LinearRegression()

linreg.fit(features, lables)

model_path = "../model/linreg_6.model"

joblib.dump(linreg, model_path)
import sklearn.preprocessing as preprocessing
#进行标准化
```

```
scaler = preprocessing.StandardScaler()

temperature_h_scale = scaler.fit(train_data_dum[['temperature_h','temperature_l','
                                                 temperature_average','temperature_abs','
                                                 weather_average','weather_abs']].values)

scaleDf=DataFrame(temperature_h_scale.transform(train_data_dum[['temperature_h','temperature_l',
                                                 'temperature_average','temperature_abs','
                                                 weather_average','weather_abs']].values),columns=[
                                                 'temperature_h_scale','temperature_l_scale','
                                                 temperature_average_scale','temperature_abs_scale'
                                                 ,'weather_average_scale','weather_abs_scale'])

train_data_dum_scale=pd.concat([train_data_dum,scaleDf],axis=1)

train_data_dum_scale.drop(['temperature_h', 'temperature_l', 'temperature_average', '
                                                 temperature_abs', 'weather_average', 'weather_abs'
                                                 ], axis=1, inplace=True)
```