**共享单车的调度研究**

**小组成员：赵瑞冬（组长）牛庆源 陈赓珊**

目 录

第一章 绪论

1.1 研究背景与目的

1.2校内、合肥市共享单车骑行规律

1.3 国内研究现状

1.4 本文工作

1.5 文章结构

第二章 相关知识

2.1 使用KMEANS方法构建投放模型

2.2使用时间序列预测法构建ARIMA模型

2.3 模型优化

第三章 具体研究算法

3.1 K-means聚类算法

3.2 遗传算法

第四章 结果分析

4.1 K-means算法

4.2 遗传算法

第五章 总结与展望

1. 绪论

1.1 研究背景与目的

根据《中国共享出行发展报告（2020-2021）》显示，截止 2020 年12 月底，共计有共享单车1950 万辆，注册用户超过3亿人次，日均订单数达到4700万单。随着单车数量和使用人数的激增，运营商能够收集到大量的用户骑行信息。这些信息 涵盖用户骑行时间、起始地点、单次骑行时长以及是否购买骑行卡等，形成了非 常有价值的轨迹信息，能够真实地反映了用户的骑行需求。此外，共享单车作为公共交通的一种重要补充形式，同时也作为校园交通、城市交通的“毛细血管”，能够很好地解决出行“最初一公里”、“最后一公里”的问题，同时满足城市居民多样化出行需求。

然而，尽管共享单车系统有显著的优势，但要有效预测各站点流量和平衡站

点车辆需求极具挑战性。用户的行为是动态变化的，这种变化在一个城市的每个地方不尽相同，这使得共享单车使用情况变得非常的不平衡，导致某些站点无车可用，而另一些站点出现车辆堆积的情况。

上下课区间、饭点时间的科大校园就存在严重的上述情况，很多学生抢不到共享单车，而又有很多单车在人流量少的地界停放。严重影响了科大学生的时间安排以及就学心情，造成了科大学子在特定时间段糟糕的用户体验。

同时，合肥市的共享单车营运公司尝试在城市划定区域，聘请管理人员对区域内各站点的车辆进行调度，在一定程度上缓解该问题。但是由于管理人员只能凭借经验判断出站点在不同时段可能需要调度的车辆数，效率较低，反而影响了共享单车系统的运行效率。比如合肥市的居民区地带，居民出行的时间相对集中，短时间内对车辆的需求量大，并且波动频繁，利用人工调度解决该地区的站点间车辆不平衡问题效率低，没有办法彻底解决“供不应求” 和“供过于求”的不平衡现象，这大大流失了客户资源。

对于共享单车运营管理商而言，合理且高效地在各个站点之间重新分配共享单车数目至关重要。唯此，才能让共享单车系统在减少资源和人力损耗的前提下有效运行，进而提高用户的使用体验，提高公司自身的经济效益。

1.2校内、合肥市共享单车骑行规律

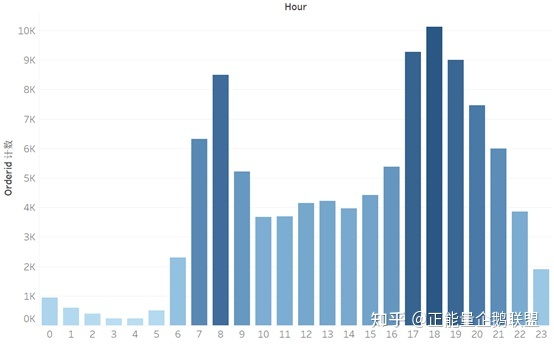
研究科大校园以及合肥市的共享单车调度问题基于区域内用户的骑行规律。影响单车流量预测以及用户骑行规律的因素复杂、多样。因此，本文着重于通过两个方法收集数据。

①查找共享单车公司公布的开源数据

如中国北京美团（摩拜）单车的数据等。但是由于城市地理因素差异的问题，该数据只能提供合肥市城市居民用户使用共享单车时间段的参考。

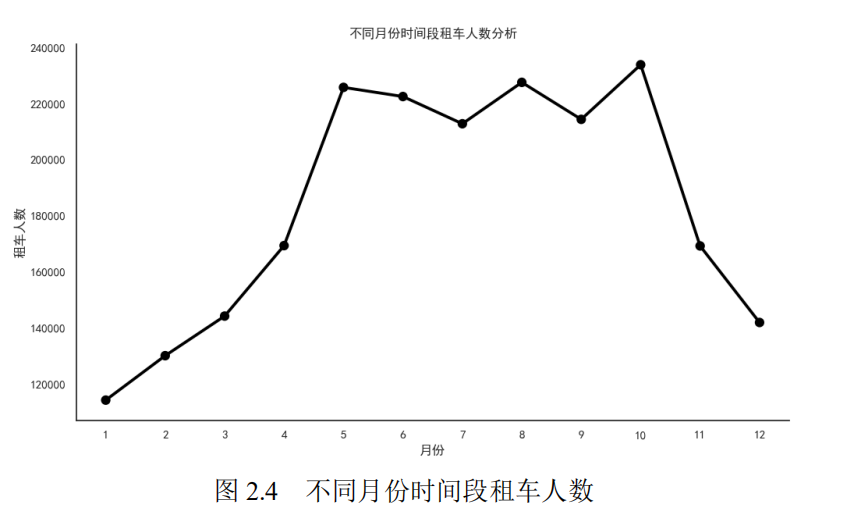
通过下图柱状图，我们可以直观地看到，每天的两个骑行高峰期是早8点和晚6点，刚好和上下班高峰期相吻合，是共享单车的使用峰值。而晚6点的单车使用量比早8点多，早8点的订单量少于晚6点的订单量可能因为下午下班用户目的地更多样化，也可能是早上上班族花时间找单车的时间成本过高。

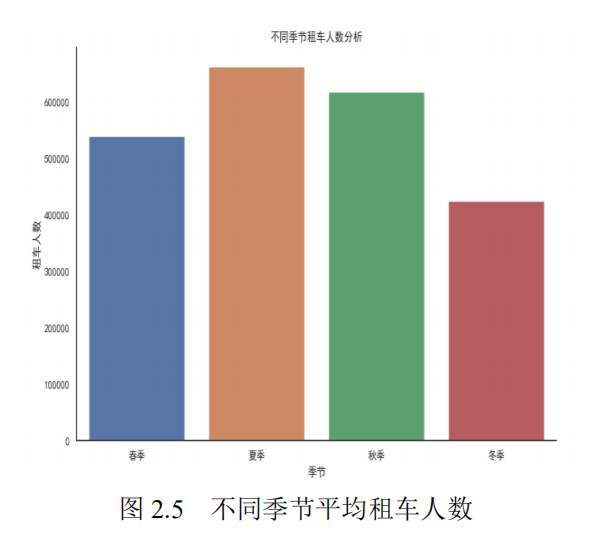
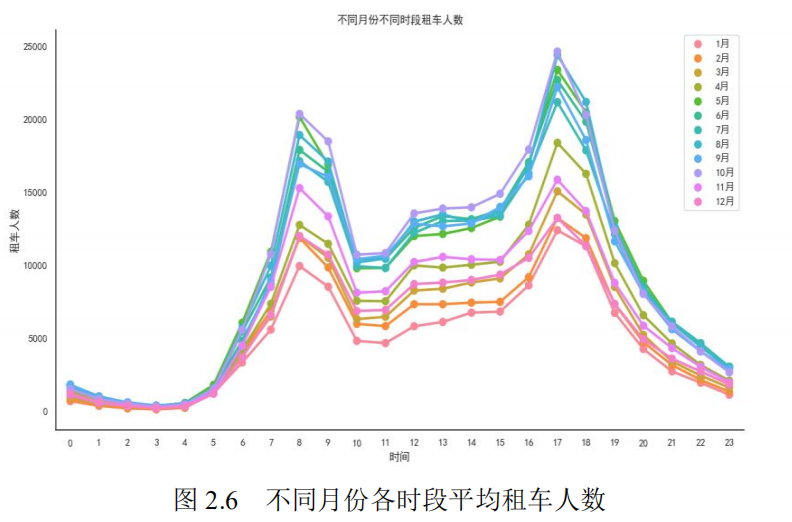
图1-不同时段共享单车使用情况

这启示运营公司可以利用低频的使用时间进行单车的回流调度。为了避免用户在急用时找不到单车的情况，应多在地铁，办公区域投放车辆。

选取数据时我们不止选择一组（一地）的数据，对于至少两组（两地）的数据进行对比研究。以发现共同特点以及不同特点，从而科学地探究数据不同的原因，补充分析影响共享单车投放以及消费者骑车的因素。

所以我们也参考了美国City Bike 在曼哈顿中城的 2018 年 1 月-12 月的骑行数据。





从以上数据和图像中发掘信息，曼哈顿的租车高频时间段在4-11月，每月中又属星期一-星期三的租车量居高。这给后面的调度模型的时间窗分类提供依据。

②寻找科大学生用户、合肥市居民用户做问卷调查

我们依据影响因素提出问题，发放在网上以供社交圈内的科大学子填写，整理得到的数据。

我们又在社交圈内、城市论坛等处收集了合肥市市民的意见。当然，不仅仅是消费者，也对投放者、运营者进行了相关调查。

整理为以下问题：1.缺少调度车，调度进程缓慢；存在大量闲置调度车辆，资源浪费。2.调度没有对需求点的闲置车辆及时取走或补充空缺车辆，用户满意度下降。3.某一区域的共享单车供不应求或供过于求。

1.3 国内研究现状

国内诸多学者针对共享单车调度问题展开了许多研究。陈菁等根据大学校园内共享单车使用的特点进行了需求区域的划分，建立了大学校园共享单车调度路径规划模型。关宏志等提出一种分层调度策略，将调度区域划分为道路层和小区层，并采用不同的调度车 辆。徐国 勋等提出了“红包车”机制，将一些闲置的共享单车设置为红包车从而鼓励用户参与调度，并以总成本最小建立了混合整数规划模型。吕畅等以供需关系为基础，提出一种站点分块策略，并设计了双层禁忌搜索算法进行求解。在调度算法方面，遗传算法具有很好的收敛性，相同精度下计算速度快且鲁棒性高，因而被广泛使用。万敏建立了以成本最小化为目标、带时间窗的区域共享单车调度模型，并通过遗传算法求解该调度模型。于德新等引入了精英策略和进化算子，并采用了 TOPSIS法来求解最优路径。蒋塬锐等在调度结构中引入了调度池，设计了单亲遗传算法求解。

1.4 本文工作

我们了解了相关算法，尝试理解一些算法的原理，通过搜集资料找到了美国曼哈顿市的共享单车数据，并借助python工具使用K-MEANS聚类算法对数据进行了处理并得到了可视化图表。并且提出一种考虑高校课程时刻表的校园共享单车调度方法，并通过遗传算法进行求解，以满足调度点出行高峰时的共享单车需求为主要目标，兼顾调度成本和学生满意度2个指标获得最优的调度方案。

1.5 文章结构

本文第二章介绍了有关算法的知识，第三章对我们使用的方法进行了介绍，第四章对运行过程和结果进行了分析，最后对所做工作进行了总结。

1. 相关知识

2.1 使用KMEANS方法构建投放模型

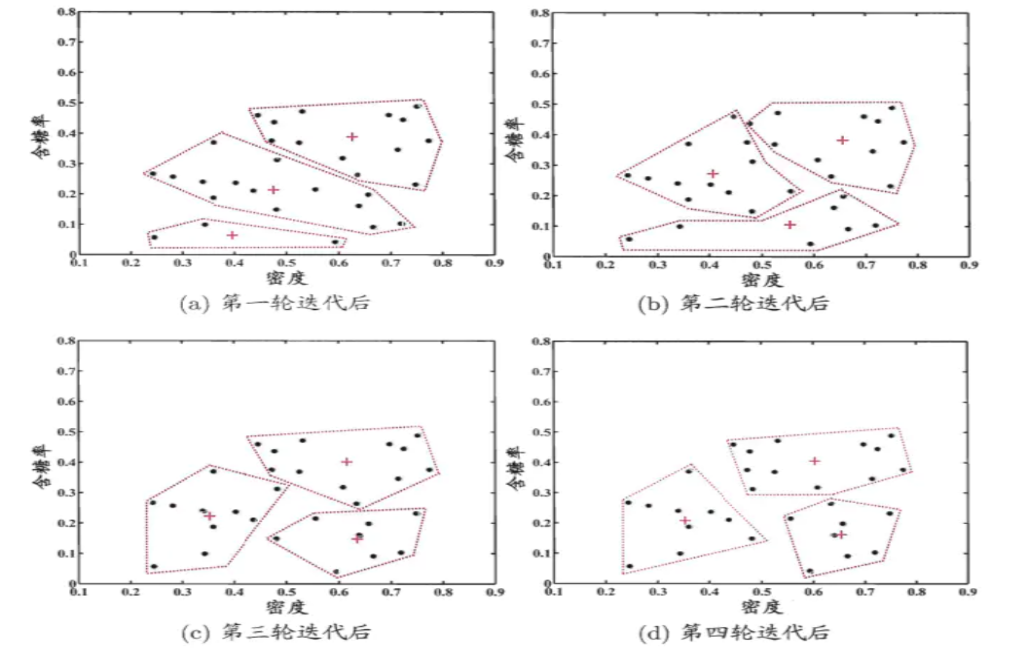
除了以上根据相关因素得到的数据之外，我们还需要一些其他的数据。用于构建投放模型，我们需要对不同地区和不同区域投放点的数量做一个统计。对于各个投放点的位置，可以选择去查找初始点位，运用KMEANS方法减少点位（分类），框选每一个小区域进行调度的研究；或者如果调查范围够大，直接随机选取初始点位用KMEANS法模拟投放点。

使用KMEANS方法的想法来源于

KMEANS聚类法：选k个类中心，将与类中心最近的点分配给类中心，重新计算类中心，迭代多次（如下图所示），直至类中心不变（或者类中心移动小于某一值）为止。

可以先利用合理的假设来减少次要因素的影响，构建完成之后与真实数据对比验证。

例如：假设聚类后的每一类为一个调度需求点，每一个需求点之间均独自调度，每一个调度需求点内所有的投放点服务完后才能完成下一个调度任务。

假设初始的投放数量均为最佳的投放数量。调度数量 = 以调度区域为终点数 - 以调度区域为起始点数。

收集好数据，接下来就是模型的构建。

可以想到的是我们可以将所有影响到调度的因素收集起来，作为自变量，调度方式作为因变量，构建函数模型，用残差或者其他方法检验拟合是否正确，从而得出函数模型，再与用于验证模型的数据作比较，可以得出模型正确与否，否则再调整。

这种方法体现不出探索性，把课题研究的重点大部分放在了找因素上，因素很多很杂，就算用主成分分析法挑出主要因素（比如骑行时间和骑行距离可以整合成一个变量，因为两者具有线性关系），函数的构建也大概率只是一个试错的过程，探究方法的重要性体现不出来。

2.2使用时间序列预测法构建ARIMA模型

经过查阅和参考相关论文，我们找到了另一种方法：时间序列预测法

即利用历史大数据，采用机器学习中的相关模型进行预测，机器学习目前发展十分成熟，预测模型有很多，例如 SVM、ARIMA、RF 等，很多学者都将其运用在各种预测问题上，取得了不错的成果。

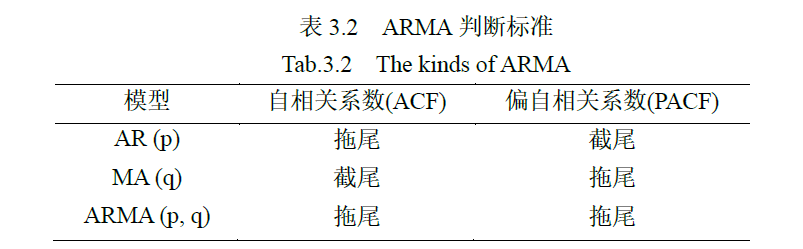
这里主要研究应用最广泛的ARIMA模型。

ARIMA模型实现过程叙述

① 序列平稳化----确定数据的平稳性

首先将分析的时间序列绘制成散点图，然后观察图的大致趋势，分析其自相关和偏相关函数等指标，判断此序列是否平稳。若该序列不平稳，则对其进行差分处理使其转 变为平稳序列，然后进行分析。

② 模型识别----确定模型的形式

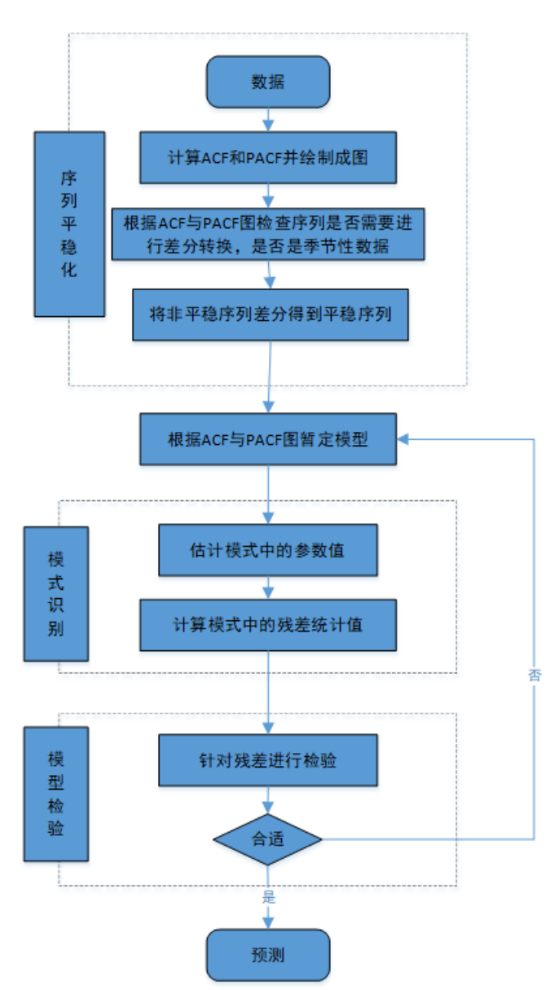
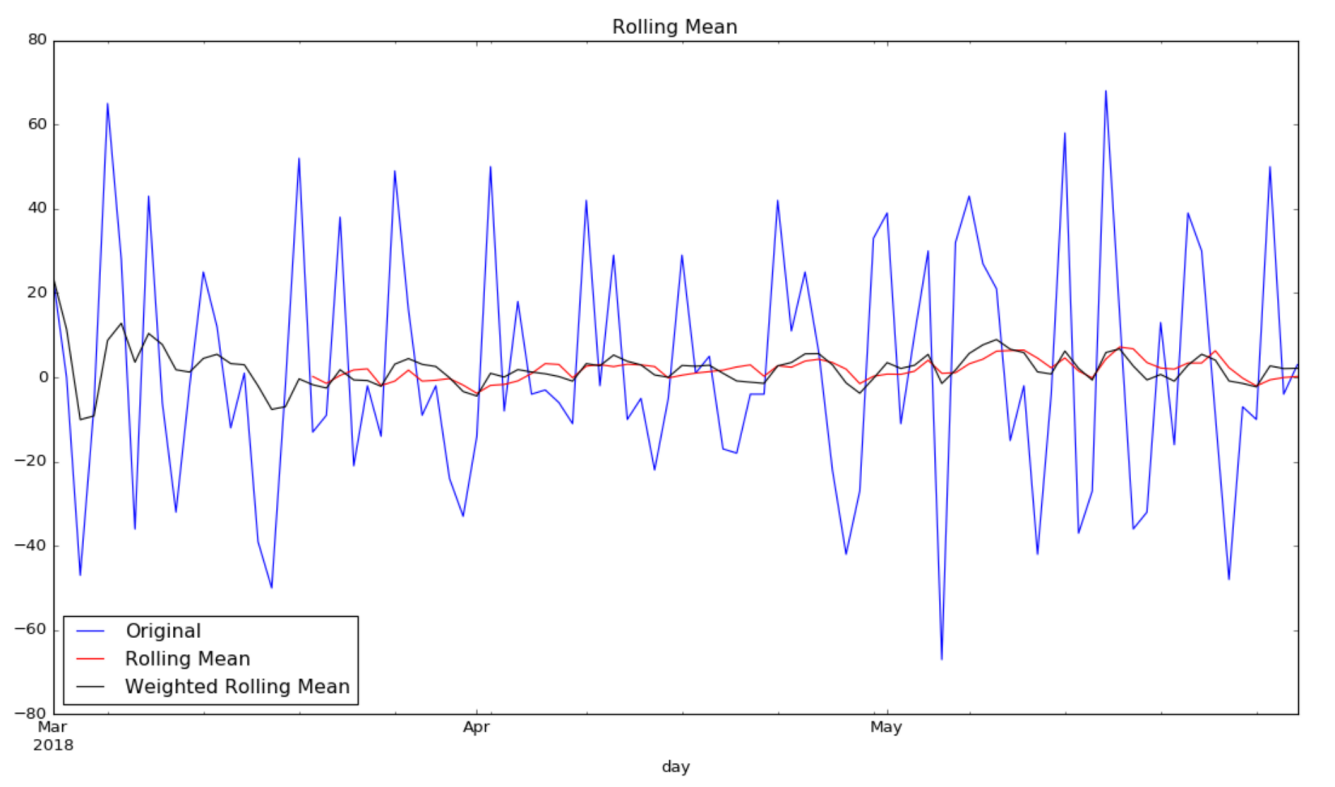
依据下表的判断标准判断模型是何种类型，若ACF拖尾，PACF截尾，则该模型数据AR模型。若ACF截尾，PACF拖尾，则该模型数据MA类型。若两者均为拖尾，则模型属于ARMA类型。

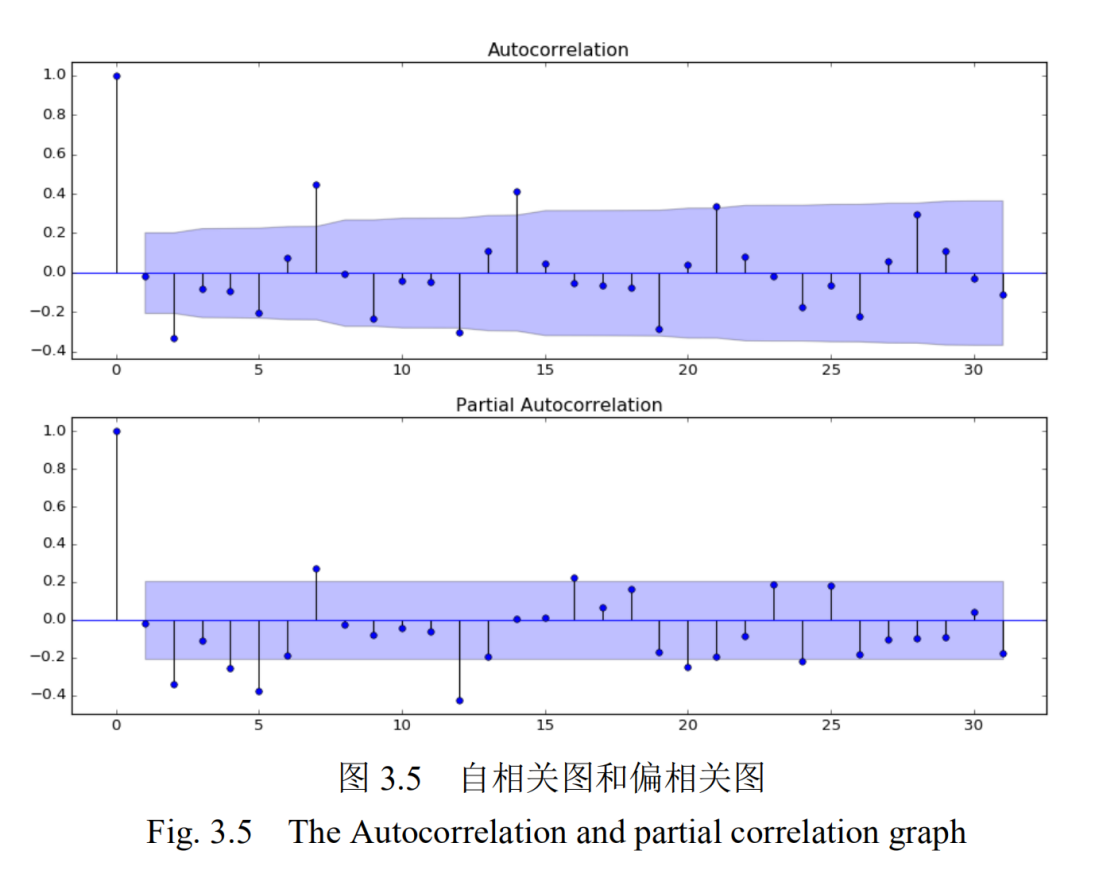
③ 模型检验----根据残差选择合适的模型

对确定好的模型，进行参数估计，并进行随机性检验，诊断残差序列是否为白噪声。然后对检验后的模型确定变量的滞后阶数，即确定p和q的值。

④ 模型预测----对选择的模型进行预测和评价

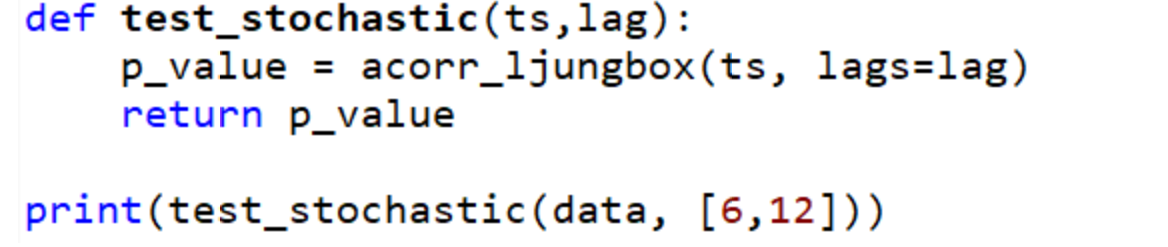
用确定好参数的模型，对目标数据进行预测和评价。

序列平稳化检测：下图的蓝色曲线为调度区域1的调度需求量原始图；红色曲线是滑动窗口为20的移动平均图，可以看出此图均值在 0 上下浮动；黑色曲线为指数平均图，用于剔除周期等因素对数据造成的影响。

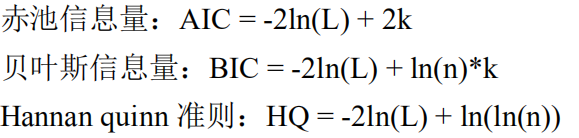
模型识别（确定类型为AR, MA,还是ARMA。）：依据自相关系数和偏相关系数识别。

如果结果为左图所示，两种系数均为拖尾，则依据对照表，可以确定模型为ARMA模型，接下来需要确定q和p。

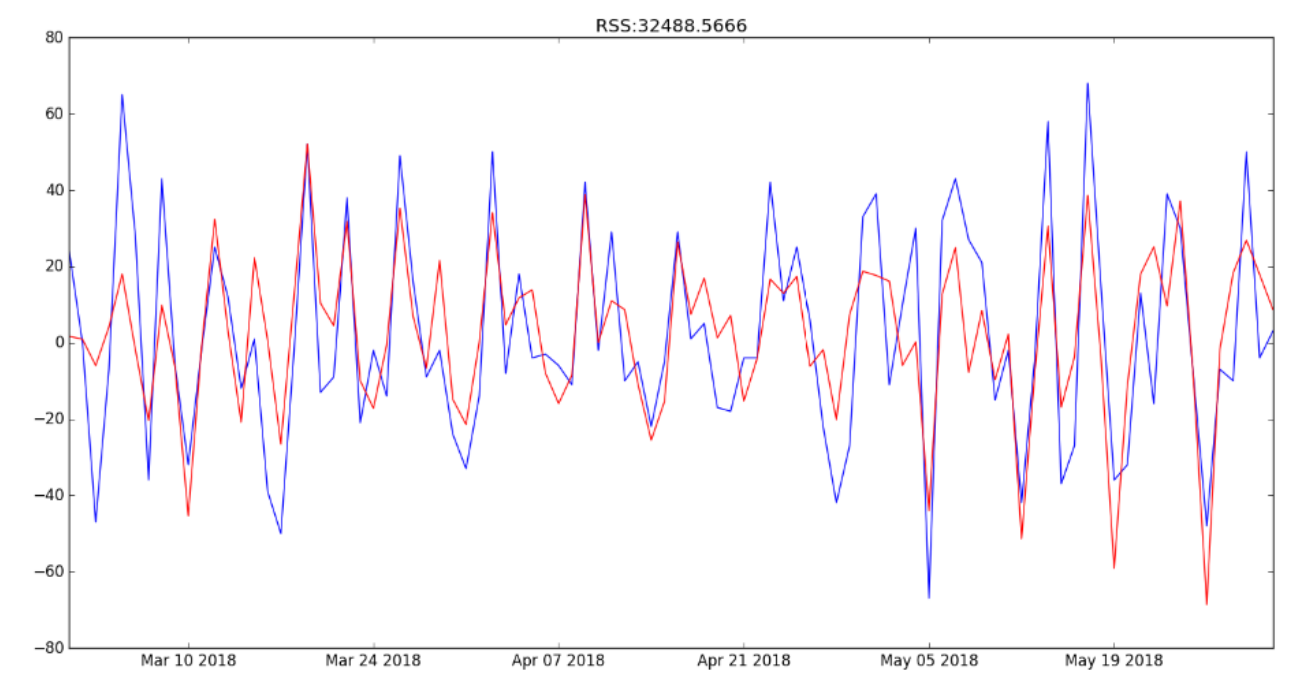
模型检验。

首先白噪声检验：编写程序检验

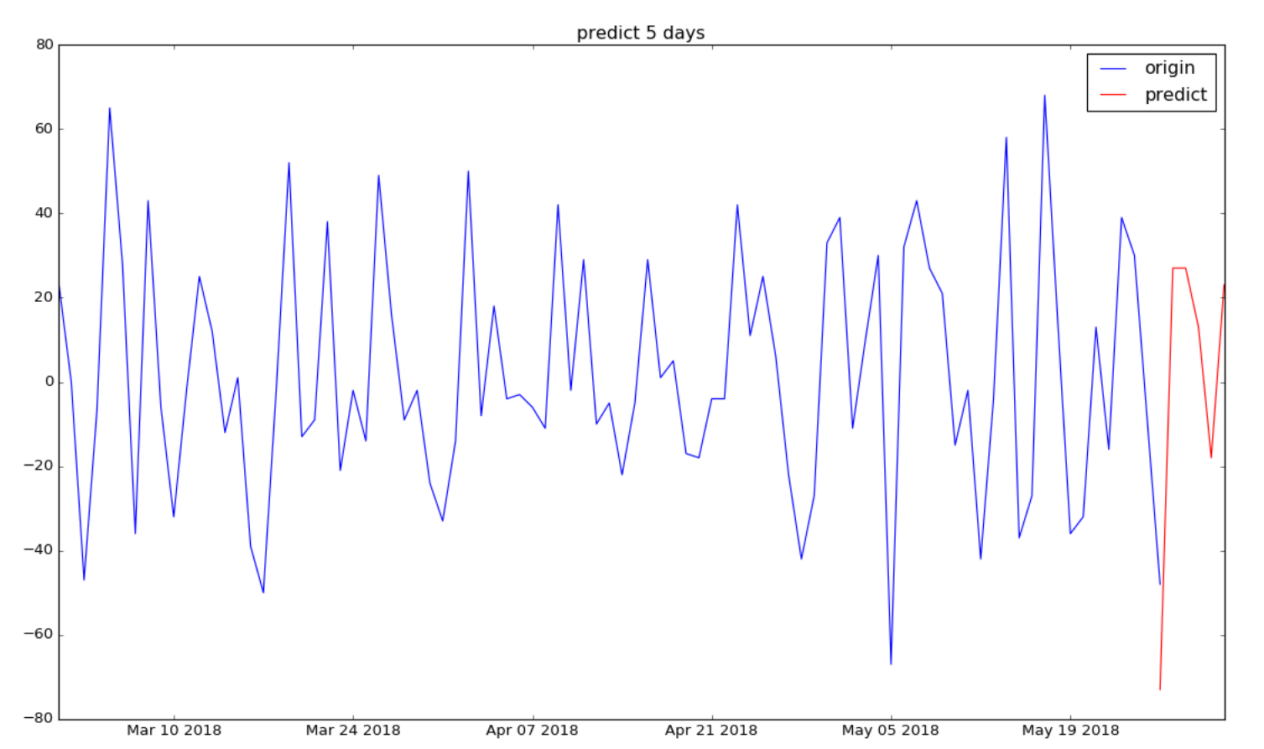
如果不是白噪声序列，则确定p和q。

 参考：

以上三个准则确定p和q的值。

模型预测。作图进行预测。将真实值与预测值放在同一张图中观察，再对残差等数据观察拟合度如何。

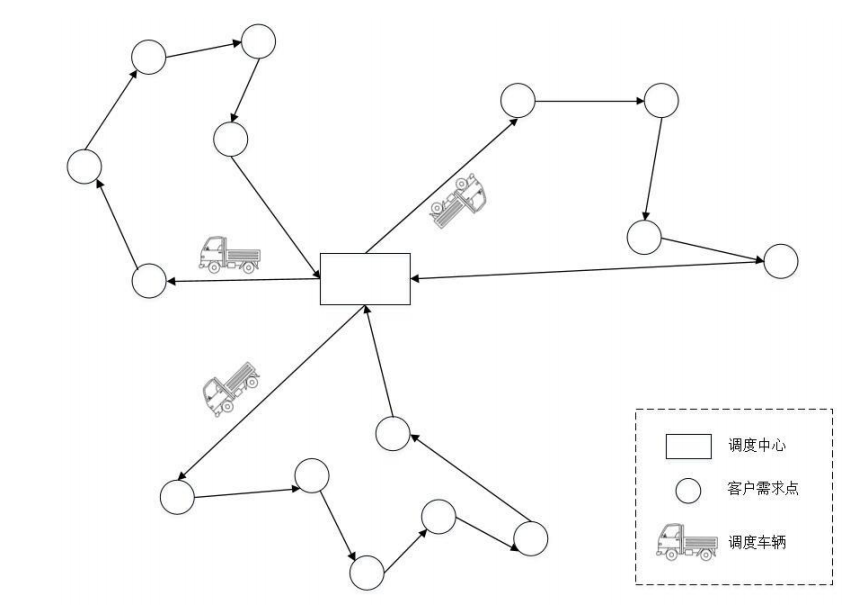
真实和预测的对比：

2.3模型的优化

构建并检验之后可以进行模型的优化。调度车辆的优化也就是对调度路径的优化，即VRP问题。

VRP问题：它指由配送中心向具有不同需求的不同客户提供货物，由一个车队或多个车队组织适当的路线，向不同客户需求点发送货物，并能够在满足一定约束条件下达到路程最短或企业成本最小或耗费时间最少等目的。

基本示意图如下：



VRPSDPTW问题属于VRP问题的一种，而VRP问题早被许多学者证明为NP-hard问题。即当问题的求解规模增加时，相应地求解难度也会增加。针对小规模的问题，一般采用精确算法；针对大中规模的问题，一般采用启发式算法求解。

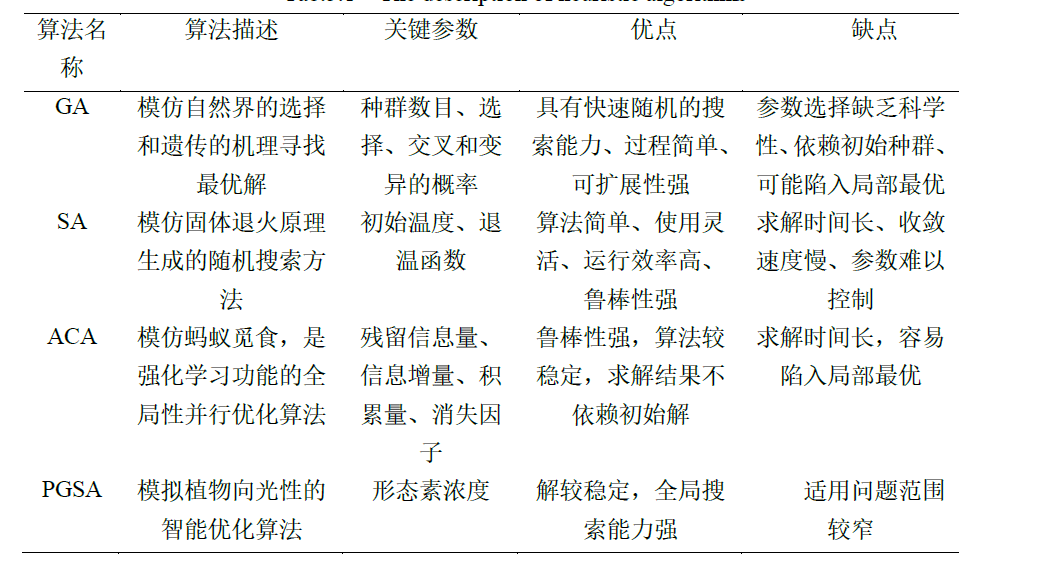
精确算法：解决此类VRP问题的精确算法包含分支定界法、动态规划法、枚举法等。精确算法的优点在于一定可以求得全局最优解，但是是需要付出时间代价的。一旦问题的求解规模的增加，求解时间则呈指数形式增长。故精确算法常被用于求解小规模的VRP问题，而大中规模的问题，则通常采用启发式算法。

启发式算法：启发式算法是相对于最优化算法提出的，为了解决最优化算法的求解速度慢等缺点。启发式算法首先给定每一个实例的可行解，然后利用一些规则对可行解进行调优，在短时间内得到问题的满意解。启发式算法一般分为传统启发式算法和现代启发式算法。

(1) 传统启发式算法

传统启发式算法一般需结合其余算法使用，因为采用此算法求解前需要构造初始解，然后对初始解不断改进，使其更优。虽然它能够在较短的时间内求解NP-hard问题，但是容易陷入局部最优解。常见的传统启发式算法包含插入算法、节约算法等。

(2) 现代启发式算法

现代启发式算法的设计灵感来源于自然界。上世纪50年代中期，仿生学被创立。许多科学家从生物钟寻求新的用于人造系统的灵感。一些科学家就从生物进化的机理中提出了模拟进化算法。例如遗传算法(Genetic Algorithm, GA)、模拟退火算法(Simulated Annealing, SA)、蚁群算法(Ant Colony Algorithm, ACA)、模拟植物生长算法(Plant Growth Simulation Algorithm, PGSA)等。不同的算法有各自的优缺点。

1. 具体研究方法

流程图：

3.1 K-means聚类算法

3.1.1 调查因素

1. 时间因素

（1） 季节以及每一个季节每天的骑行时段

（2） 工作日和非工作日

2. 空间因素

（1） 骑行距离

（2） 地区因素

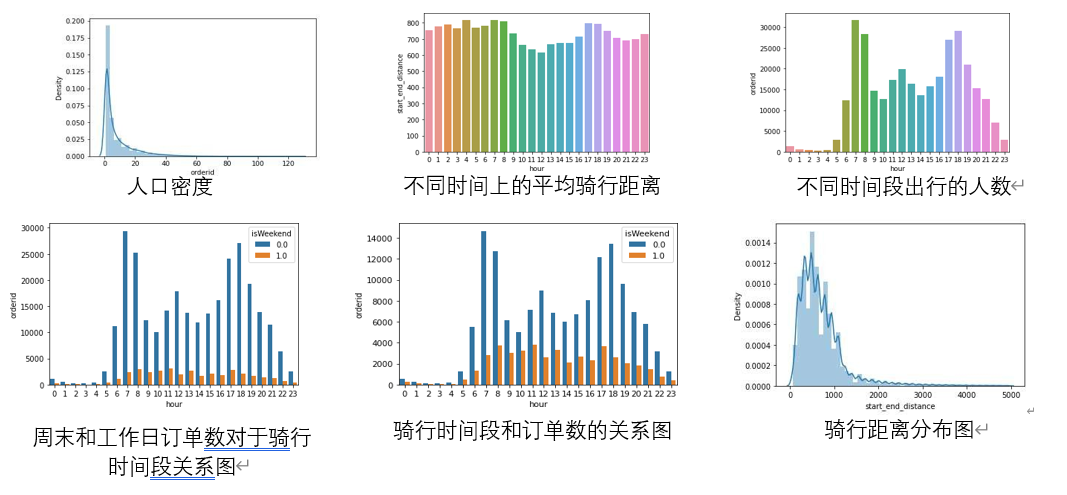
3. 用户因素。

（1） 职业

（2） 性别

（3） 年龄

4. 其他（如天气等）

3.1.2 网站参数图

3.1.3问题分析

我们最终的目的是将训练好的模型部署到真实的环境中，希望训练好的模型能够在真实的数据上得到好的预测效果，换句话说就是希望模型在真实数据上预测的结果误差越小越好。我们把模型在真实环境中的误差叫做泛化误差，最终的目的是希望训练好的模型泛化误差越低越好。用于构建最终模型的数据集通常有多个； 在构建模型的不同阶段，通常有训练集和测试集。

我们先用训练集参与了模型调参的过程使其进行拟合以估计模型，实践中，训练集通常是由输入向量（标量）和输出向量（标量）组成的数据对（这里对应出发区块和终点区块）。其中输出向量（标量）被称为目标或标签。在训练过程中，当前模型会对训练集中的每个样本进行预测，并将预测结果与目标进行比较。根据比较的结果，学习算法会更新模型的参数。模型拟合的过程可能同时包括特征选择和参数估计。

由于训练集不能反映模型真实的能力， 因此选择测试集作为提供对最终模型的无偏评估。若测试集在训练过程中从未用到，则它也被称之为预留集。

3.1.4 用python实现录入和计算（代码部分）

#常规操作，首先导入各种需要的包

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from math import radians,cos,sin,asin,sqrt

import geohash

#读取训练集和测试集

train = pd.read\_csv("train.csv", sep = ',', parse\_dates = ['starttime'])

test = pd.read\_csv("test.csv", sep = ',', parse\_dates = ['starttime'])

#取样

train = train.sample(frac=0.1)

def \_processData(df):

#将starttime 分成weekday,day,hour三类，方便后续不同时间段数据展示

df['weekday'] = df['starttime'].apply(lambda s : s.weekday())

df['day'] = df['starttime'].apply(lambda s : str(s)[:10])

df['hour'] = df['starttime'].apply(lambda s : s.hour)

print('时间处理完成')

#将geohash字符串反编码，方便后续计算骑行距离

df['start\_lat\_lng'] = df['geohashed\_start\_loc'].apply(lambda s : geohash.decode(s))

df['end\_lat\_lng'] = df['geohashed\_end\_loc'].apply(lambda s : geohash.decode(s))

df['start\_neighbors'] = df["geohashed\_start\_loc"].apply(lambda s : geohash.neighbors(s))

#原数据中的geohash字符串是g7，现在转成g6

df['geohashed\_start\_loc\_6'] = df['geohashed\_start\_loc'].apply(lambda s : s[:6])

df['geohashed\_end\_loc\_6'] = df['geohashed\_end\_loc'].apply(lambda s : s[:6])

df['start\_neighbors\_6'] = df["geohashed\_start\_loc\_6"].apply(lambda s : geohash.neighbors(s))

print('Geohash处理完成')

#判断目的地是否在neighbors

def inGeohash(start\_geohash, end\_geohash, names):

names.append(start\_geohash)

if end\_geohash in names:

return 1

else:

return 0

df['inside'] = df.apply(lambda s : inGeohash(s['geohashed\_start\_loc'],s['geohashed\_end\_loc'],s['start\_neighbors']), axis = 1)

df['inside\_6'] = df.apply(lambda s : inGeohash(s['geohashed\_start\_loc\_6'],s['geohashed\_end\_loc\_6'],s['start\_neighbors\_6']), axis = 1)

print("Geohash近邻判断处理完成")

#计算起点与终点距离

def haversine(lon1, lat1, lon2, lat2):

"""

Calculate the great circle distance between two points

on the earth (specified in decimal degrees)

"""

lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

# haversine公式

dlon = lon2 - lon1

dlat = lat2 - lat1

a = sin(dlat/2)\*\*2 + cos(lat1) \* cos(lat2) \* sin(dlon/2)\*\*2

c = 2 \* asin(sqrt(a))

r = 6371 # 地球平均半径，单位为公里

return c \* r \* 1000

df['start\_end\_distance'] = df.apply(lambda s : haversine(s['start\_lat\_lng'][0],s['start\_lat\_lng'][1],s['end\_lat\_lng'][0],

s['end\_lat\_lng'][1]),axis = 1)

print("距离计算完成！！！")

return df

def \_timeAnalysis(df):

#数据包含的天数

print('数据集包含的天数如下：')

print(df['day'].unique())

print('\*'\*60)

#用户出行小时高峰期

g1 = df.groupby('hour')

print(g1['orderid'].count().sort\_values(ascending = False))

print('\*'\*60)

#周一到周日用车分析

g1 = df.groupby('weekday')

print(pd.DataFrame(g1['weekday'].count()))

print('\*'\*60)

#周一到周日不同时间的用车分析

df.loc[(df['weekday'] == 5) | (df['weekday'] == 6), 'isWeekend'] = 1

df.loc[~(df['weekday'] == 5) | (df['weekday'] == 6), 'isWeekend'] = 0

g1 = df.groupby(['isWeekend', 'hour'])

#计算工作日以及周末的天数

g2 = df.groupby(['day', 'weekday'])

w = 0 #周末天数

c = 0 #工作日天数

for i,j in list(g2.groups.keys()):

if j >= 5:

w += 1

else:

c += 1

temp\_df = pd.DataFrame(g1['orderid'].count()).reset\_index()

temp\_df.loc[temp\_df['isWeekend'] == 0 , 'orderid'] = temp\_df['orderid']/c

temp\_df.loc[temp\_df['isWeekend'] == 1 , 'orderid'] = temp\_df['orderid']/w

print(temp\_df.sort\_values(['isWeekend', 'orderid'], ascending = False))

sns.barplot(x = 'hour', y = 'orderid', hue = 'isWeekend', data = temp\_df)

def analysis\_1(data, target):

g1 = data.groupby(['day',target])

group\_data = g1.agg({"orderid":"count","userid":"nunique","bikeid":"nunique"}).reset\_index()

for each in ["orderid","userid","bikeid"]:

sns.distplot(group\_data[each])

plt.show()

return group\_data

# 两点距离

def distance(e1, e2):

return np.sqrt((e1[0]-e2[0])\*\*2+(e1[1]-e2[1])\*\*2)

# 集合中心

def means(arr):

return np.array([np.mean([e[0] for e in arr]), np.mean([e[1] for e in arr])])

# arr中距离a最远的元素，用于初始化聚类中心

def farthest(k\_arr, arr):

f = [0, 0]

max\_d = 0

for e in arr:

d = 0

for i in range(k\_arr.\_\_len\_\_()):

d = d + np.sqrt(distance(k\_arr[i], e))

if d > max\_d:

max\_d = d

f = e

return f

# arr中距离a最近的元素，用于聚类

def closest(a, arr):

c = arr[1]

min\_d = distance(a, arr[1])

arr = arr[1:]

for e in arr:

d = distance(a, e)

if d < min\_d:

min\_d = d

c = e

return c

if \_\_name\_\_=="\_\_main\_\_":

## 生成二维随机坐标

## arr形如：[ (x1, y1), (x2, y2), (x3, y3) ... ]

arr = np.random.randint(100, size=(100, 1, 2))[:, 0, :]

print(arr)

## 初始化聚类中心和聚类容器

m = 5

r = np.random.randint(arr.\_\_len\_\_() - 1)

k\_arr = np.array([arr[r]])

cla\_arr = [[]]

for i in range(m-1):

k = farthest(k\_arr, arr)

k\_arr = np.concatenate([k\_arr, np.array([k])])

cla\_arr.append([])

## 迭代聚类

n = 100

cla\_temp = cla\_arr

for i in range(n): # 迭代n次

for e in arr: # 把集合里每一个元素聚到最近的类

ki = 0 # 假定距离第一个中心最近

min\_d = distance(e, k\_arr[ki])

for j in range(1, k\_arr.\_\_len\_\_()):

if distance(e, k\_arr[j]) < min\_d: # 找到更近的聚类中心

min\_d = distance(e, k\_arr[j])

ki = j

cla\_temp[ki].append(e)

# 迭代更新聚类中心

for k in range(k\_arr.\_\_len\_\_()):

if n - 1 == i:

break

k\_arr[k] = means(cla\_temp[k])

cla\_temp[k] = []

## 可视化展示

col = ['HotPink', 'Aqua', 'Chartreuse', 'yellow', 'LightSalmon']

for i in range(m):

plt.scatter(k\_arr[i][0], k\_arr[i][1], linewidth=10, color=col[i])

plt.scatter([e[0] for e in cla\_temp[i]], [e[1] for e in cla\_temp[i]], color=col[i])

plt.show()

print(k\_arr)

3.2 遗传算法

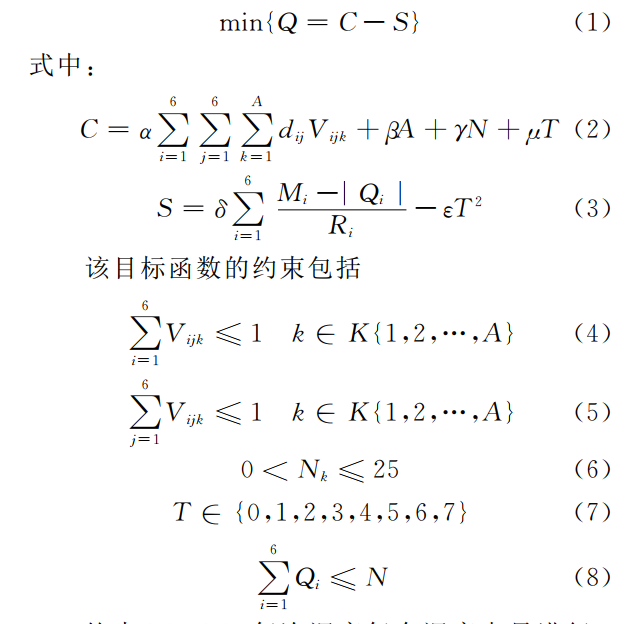
3.2.1 调度点的选取

通过实地考察，发现科大校园（仅考虑中、西区）共享单车停放最密集的地区为：第三教学楼A，第三教学楼C，西区芳华餐厅，中区学生宿舍，西区学生宿舍，西区活动中心，分别标号为调度点1—调度点6。

确定07:30-08:00､10:10-10:40､12:00-12:30､13:30-14:00､15:20-15:50､18:15-18:45､21:25-21:55 这7个时间段为1d的调研时间段,分别编号为时间段1~时间段7｡

3.2.2模型构建

本文提出的最优化目标函数O由2部分组成,其中C为总成本;S为总体满意度,优化目标为实现总目标值O最小｡在总成本C中,i､j为调度点编号;k为调度货车编号;Dij为调度点i与调度点j之间的距离;Vijk为判断变量,即第k辆货车经过调度点i到调度点j之间的路径时,Vijk取1,否则取0;A为当日使用的货车总数;N为当日调出的所有共享单车数量;T为当日安排的调度次数;*α*为调度货车运行1km所需成本;*β*为调度货车使用1次所需固定成本;*γ*为调出1辆共享单车所需的维护成本;*μ*为单次调度所需的固有成本｡在总体满意度S中,Mi为第i个调度点当前的共享单车数;Ri为第i个调度点的预期需求单车数;Qi为第i个调度点需要调度的车辆数,正为需要调入,负为需要调出;*δ*和*ε*为满意度系数。

优化的目标函数如(1)所示。

约束(4)､(5):每次调度每个调度点只进行1次调度｡

约束(6):科大的调度用车为小型三轮车,单次运力不超过25辆共享单车｡

约束(7):通过课程时刻表确定了7个出行高峰时间段,因此每天的调度次数为0~7。

约束(8):从调度总站调出的共享单车数量必须多于各调度点所需要的共享单车数量之和｡

约束(9):假设调度时所有货车同时到达和离开各站点,且路线相同,均统一先对某一辆货车进行操作,当该货车上的共享单车全部卸载完成时, 该车立刻从当前站点返回调度总站｡

另外,对于目标函数中的各成本系数及满意度系数则根据实际情况来决定｡

3.2.3 算法求解：

1.种群及个体生成

由于本次问题中需要考虑6个调度点,则以一个n×6的矩阵代表规模为n 的种群,矩阵中每个n\*6的矩阵代表种群规模为n的种群,矩阵中每一个1\*6的行向量就代表种群中的1个个体,例如,行向量【1 3 5 6 4 2】,就代表该个体所对应的解为:调度点1 > 调度点3 > 调度点5 > 调度点6 > 调度点4 > 调度点2｡

2.适应度函数设计

在上节的问题建模中，最后考虑的目标函数是成本值减去满意度值，目标函数的值越小则对应的解越好。由于适应度一般代表着１个个体的生存概率，其适应度值越大则生存下来的概率越大，因此本文的适应度函数设置为上节中目标函数的倒数。

3.操作设计

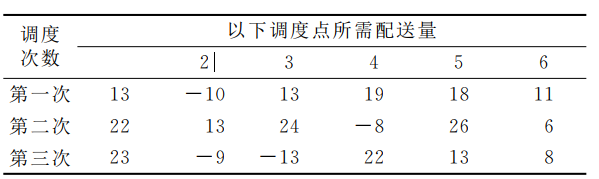
由于考虑的问题规模较小，因此只设置对优质个体的选择保留及对劣质个体的随机变异2种遗传操作。通过计算当前种群中每个个体的适应度函数值，对个体根据适应度函数值的大小进行降序排列，对排在前n%的个体直接选择保留，令排在后面的个体进行随机变异操作，这样经过多代繁殖后，优质的个体逐渐留存，劣质个体则不断变异产生新的优质个体。

4.实例分析

函数内的各个系数分别取值如下:种群规模size=20,最大繁殖代数maxgen=120,变异概率 pm=0.03,α=2.5,β=0.7,γ=1.5,μ=1.6,δ=1.2,ε=0.3｡

根据系数μ､ε 的取值可以计算出,若要使总目标值最小,应当在T=3 时,因此对每日的调度做如下安排:每日共进行3次调度,早晨8点后进行第一次调度,用以满足时间段2 和3 的用车需求;中午12 点半后进行第二次调度,用以满足时间段4､5､6的用车需求;傍晚7 点后进行第三次调度,用以满足时间段7 和第二天的时间段1 的用车需求｡

对各调度点进行实地调研后,将所得数据进行整合并求取平均值,各调度点每天3 次调度需要的配送量见下表。



以每天第一次调度为例进行求解,各个站点的现有量 M 和需求量R 如下｡

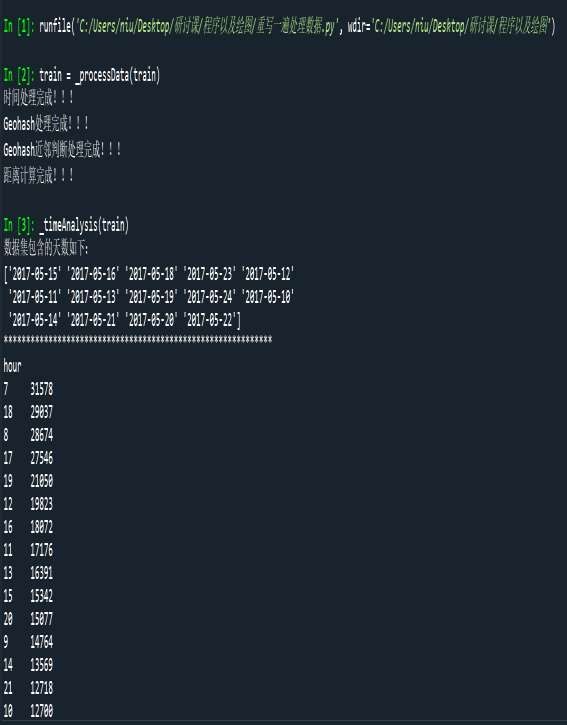
M = [97 86 45 63 22 28]

R = [110 76 58 82 40 39]

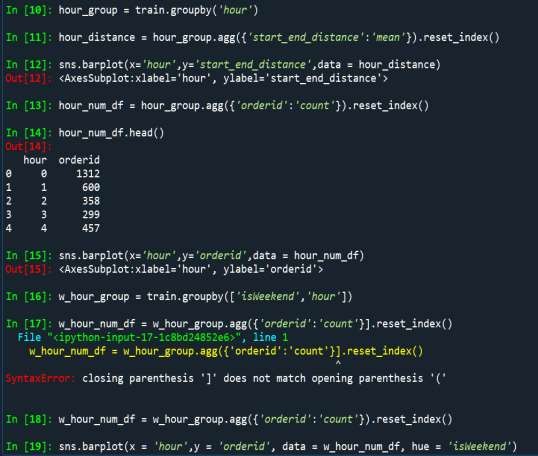
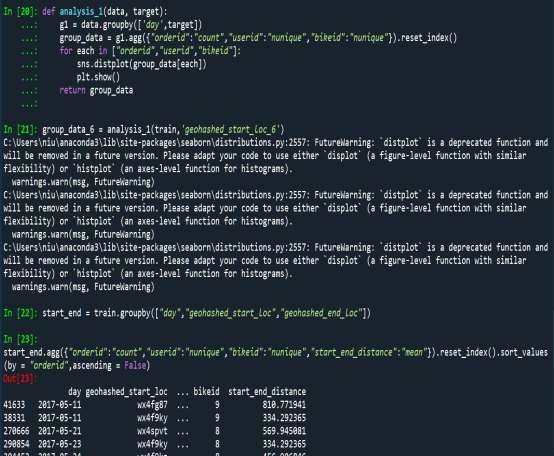
从而计算得到Q｡

Q = [13 -10 13 19 18 11]

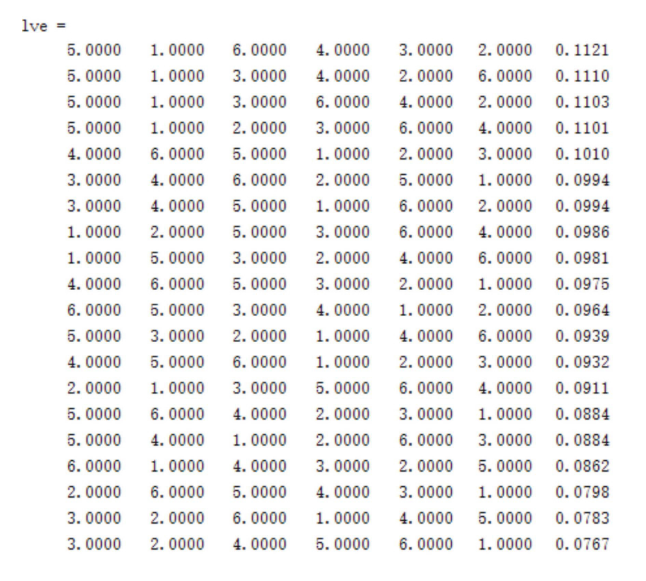
1. 运行结果与分析

4.1 K-means运行结果

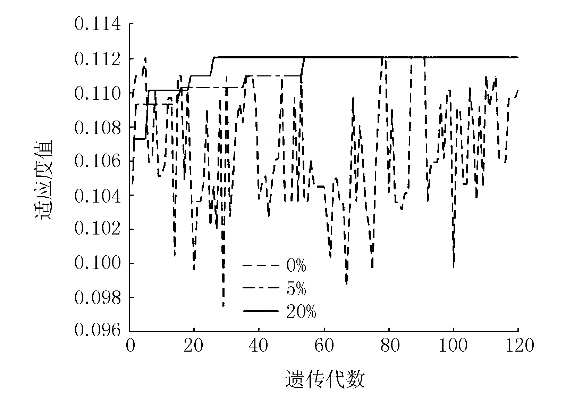




4.2 遗传算法

使用MATLAB编写程序进行求解，将参数代入，得到如图所示的结果：

每个个体的前6 个元素代表问题的解,第7 个元素代表对应的适应度值,由此可以得到在该初始条件下最佳的行动轨迹为:调度点5→ 调度点1→调度点6→调度点4→调度点3→调度点2｡

另外,选择保留的比例对于遗传算法的求解效率有着较大的影响,对上述条件下分别保留0%,5%,20%的优质种群的求解过程进行记录,得到结果见遗传代数图｡

本算法的优点如下:

1)对调度点的需求满足程度更高｡

2)每天调度次数固定,减少了工作量并降低了调度成本｡

3)提高了共享单车的利用率｡

第五章 总结与展望

本文提出的校园共享单车调度方法，综合了课程时刻表因素，利用遗传算法提升调度方案求解的准确性。总体来说，考虑课程时刻表因素的共享单车调度算法在高校内适应性强、实用性好，充分结合了校内的出行特征，对满足共享单车需求、降低调度成本、提升共享单车利用率均有明显帮助。此外，遗传算法在解决校园小规模车辆路径问题上有明显优势，计算速度快且不易陷入局部最优，保留一定比例的优质子代对算法求解有明显帮助。

本文考虑的只是科大校园一个理想的缩略情况，未来将考虑校园内更多的调度点，将算法的覆盖面扩展至整个校园，并尝试获取运营商掌握的车辆实时分布情况的部分信息，提升算法实时优化的能力；同时深入细化课程时刻表对高峰时间的划分，考虑在不同地点不同时间上课的人数对于共享单车调度的影响。

**参考文献：**

1. 赵明明. 数据驱动下的共享单车调度优化研究[D].大连理工大学,2019.DOI:10.26991/d.cnki.gdllu.2019.001780.
2. 杨证轲,董恺凌,张学梅.国内外共享单车研究综述[J].成都大学学报(社会科学版),2018(02):27-33.
3. 参考自相关系数 ACF与偏自相关系数PACF，拖尾和截尾 - ylxn - 博客园 (cnblogs.com)
4. 罗春芳,张国华,刘德华,朱定欢.基于Kmeans聚类的XGBoost集成算法研究[J].计算机时代,2020(10):12-14.DOI:10.16644/j.cnki.cn33-1094/tp.2020.10.004.
5. 基于K-means聚类算法改进算法的研究[J]. 魏杰. 信息通信. 2018(05)
6. 关宏志,卢笙,宋茂灿.共享单车分层调度策略研究[J].重庆交通大学学报(自然科学版),2020,39(02):1-7.
7. 徐国勋,李妍峰,金大祥,李军.“红包车”机制下的共享单车调度问题[J].系统工程理论与实践,2020,40(02):426-436.
8. 石兵,黄茜子,宋兆翔,徐建桥.基于用户激励的共享单车调度策略[J/OL].计算机应用:1-11[2022-06-11]
9. 陈植元,林泽慧,金嘉栋,李建斌.基于时空聚类预测的共享单车调度优化研究[J].管理工程学报,2022,36(01):146-158.DOI:10.13587
10. 杨俊闯,赵超.K-Means聚类算法研究综述[J].计算机工程与应用,2019,55(23):7-14+63.
11. 卢琰. 共享单车车辆调度问题研究[D].西南交通大学,2018.
12. 周奕杉. 共享单车需求预测及优化调度算法的研究与应用[D].西安电子科技大学,2021.DOI:10.27389
13. 张辉,郑彭军.基于蚁群算法的城市公共自行车调度研究[J].科技与管理,2015,17(06):32-36.DOI:10.16315
14. 谢艺伟. 共享单车空间分布演化规律及其调度策略研究[D].南京师范大学,2021.DOI:10.27245
15. 李艳,徐伦.基于BP神经网络实现共享单车调度优化研究[J].信息记录材料,2020,21(10):171-172.DOI:10.16009
16. 吴会丛,王敬.基于改进蚁群算法的共享单车配送调度研究[J].计算机应用与软件,2020,37(09):35-41+55.
17. 徐国勋. 城市共享单车调度优化问题研究[D].西南交通大学,2019.DOI:10.27414
18. 岳晓鹏,全启圳,何月华.基于模拟退火算法的公共自行车调度优化问题[J].科学技术创新,2021(36):19-21.
19. 蒋塬锐,贾顺平,李军.基于调度池的共享单车调度研究[J].交通信息与安全,2019,37(05):124-132.
20. 钟礼志. 基于K-means聚类算法的共享单车电子围栏规划[D].西南交通大学,2021.DOI:10.27414