

# 基于车流量下信号灯优化模型的交通流量管控

## 摘要

针对问题一，通过应用 K-means 聚类算法，将一天划分为四个时间段，分别为凌晨静默期 0-5 点、早高峰时期 6-11 点、午高峰时期 12-17 点和晚高峰时期 18-23 点。同时，将路口的相位划分为南向北、北向南、东向西、西向东及其包含的直行和转弯，共 12 种相位。以由东向西的 3 个相位，4 个小时段为例子，其车流量数据支撑表 3 在正文中展示，由于表格数据量过大，其余表格数据在支撑材料上。

针对问题二，参考基于车流量检测的智能交通信号配置优化研究的相关文，为了将问题简化分析，并且从实际出发，结合多种交通管理标准文件，设置了合理的信号灯优化配置方案。通过结合机器学习技术，对优化方案进行了多次模拟验证，结果表明该方案能够满足车辆正常通行不堵车的前提下，实现了车流量均速最大。具体红绿灯优化配置方案详情请见支撑材料中的《工作日的各路口红绿灯优化配置表》与《工作日的各路口红绿灯优化配置表》。

针对问题三，利用 Python 编程运行 K-Means 聚类分析以找出巡游车辆，对数据分析发现，可以判定 Cluster 2 可能是最符合巡游车辆特征的簇，因为它们的时间差较小、移动距离较短且频繁经过更交叉口。并针对景区出入口统计出车辆景区内停车时间为 2.87 小时。进一步分析后，做出了以 15 分钟为一个时间段的巡游车辆统计图。使用微积分法，对整个巡游车辆统计图以 2.87 小时为时间段的面积分析统计，找出图中在以 2.87 小时为横轴的情况下，面积最大的区域即为所求的一天中的停车需求量的峰值，由此提出在 5 月 1 日增加 311 个停车位，5 月 2 日增加 403 个停车位，5 月 3 日增加 359 个停车位，以及 5 月 4 日增加 218 个停车位的建议，以满足需求。

针对问题四，使用了熵权法和 TOPSIS 整合为模型，对五一黄金周期间的临时管控措施在两条主路上的效果与平常工作日、休息日在车流量、巡游车辆占比、车流平均移速三个数据进行了对比与评价(数据结果详情见表 8、数据的对比与评价结果详情见表 9)。结果表明，管制措施使得路口的交通通行能力强于休息日，而弱于工作日。这说明管制措施的效果明显，能够提高交通通行能力。

**关键词：**K-Means 聚类分析 交通流量管控 熵权法和 TOPSIS 整合

## 一、问题重述

自古以来，衣食住行便是人们生活中最重要的需求，而在近几年来，城市化高速发展，汽车的快速普及，使得几乎是家家户户都有一辆车，由此，又引发了相应的交通管控问题。

**问题一：**首先针对附件二进行数据分析和处理，利用处理所得到的数据，针对经中路-纬中路交叉口进行分析，求出该路段的车流量，并且对数据进行差异分析。然后，利用所得到的分析结果，进行一天时段的划分。最后，针对每一个划分的时段，结合处理附件二所得的数据进行各个相位的车流量进行估计。

**问题二：**利用已有的数据与模型，建立本题目的数学模型来对经中路与纬中路沿途所有交叉口的信号灯配时方案进行合理的优化。对于该方案确保能在交通安全且合理流畅的基础前提上，提升该道路车流的平均车流速度。

**问题三：**分析五一长假期的车辆流动数据，特别是那些在景区附近循环寻找停车位的车辆行为。基于此数据，计算出为了应对假期车流高峰，景区周边需要额外开放多少临时停车场空间。

**问题四：**根据附件3中提供的五一假期临时交通管理措施详情，评估这些措施在经中路与纬中路两条主要街道上实施的效果，包括对交通流量、速度以及停车情况的影响。

## 二、模型假设

**1、数据完整性假设：**假设附件2提供的监控数据是完整的，即所有通过监控点的车辆都被记录了下来。

**2、交通模式假设：**假设小镇的交通模式在一周内是稳定的，即工作日和周末的车流量模式相似，没有特殊事件或节假日影响。

**3、车辆行为假设：**假设所有车辆在通过交叉口时的行为是随机的，即左转、直行和右转的概率是均匀分布的。

**4、交通信号假设：**假设在数据收集期间，交通信号灯的设置是固定的，没有因为特殊情况而调整。

**5、车辆类型假设:**假设所有车辆的类型(如私家车、公交车、货车等)对车流量的影响是一致的,或者可以忽略不计。

**6、天气和季节影响假设:**假设在数据收集期间,天气和季节变化对交通流量没有显著影响。

**7、交通规则遵守假设:**假设所有司机都遵守交通规则,没有违章行为影响车流量的统计。

**8、数据准确性假设:**假设监控设备记录的数据是准确的,没有技术故障或人为错误导致的数据丢失或错误。

### 三、符号说明

符号	说明
$k$	聚类分量
$c_i$	第 $i$ 个聚类中心
$J$	问题一聚类分析目标函数
$K_{\text{平均}}$	每小时平均车流量
$C_{\text{总}}$	三个高峰期的总车
$X_{\text{主}}$	主方向车流量
$Y_{\text{副}}$	为副方向车流量
$Z_{\text{左}}$	左转车流量,
$G_{\text{左}}$	对应相位左转的概率
$L_{\text{车}}$	对应相位的车流量
$\mu$	簇质心 $\mu$
$\mu_j$	表示第 $j$ 个簇的质心
$A_i$	每个时间段 $i$ 的总和

---

$C_j$	第 $j$ 个时间间隔内的车辆数量
$J_{\text{间隔数}}$	$J_{\text{间隔数}}$ 为时间间隔的数量,
$G_{\text{停}}$	每一辆车的平均停车时间
$n$	每一段时间巡回车辆统计数

---

## 四、模型建立与求解

### 4.1、对问题一的模型建立与求解

#### 4.1.1、对问题一进行分析

问题一提出了对车流量进行分析的要求，其中，目标分析路段为经中路-纬中路交叉口。对于此，首先需要对附件二中该交叉口的车辆数据进行数据清理和与异常值处理的数据处理工作。然后，对处理所得到的数据，进行可视化操作等，取得车流量的变化趋势图等，以此了解车流量变化，而后根据车流量的变化将一天划分为不同的时间段。接下来，统计每个时间段内的车流量，并区分直行与转弯的车辆数量。之后，建立数学模型来描述不同时间段内车流量的变化规律。最后，根据模型结果估计不同时间段内各相位的车流量，并进行合理性验证。

#### 4.1.2、对问题一的模型建立

首先，问题的核心在于，将一天分成几个时间段，而后估算不同时间段的各个相位的车流量。由此，我们利用 python 编程，对附件二数据进行适当的处理，剔除无效的数据。对相关数据按照时间序列进行合理排布，构建时间序列。

以车流量的差异为基础，将一天分为若干个时间段，对此，利用 K-means 聚类算法<sup>[1]</sup>对时间段进行划分。考虑到本题目数据为二维数据，对于此使用欧几里得距离作为距离公式进行计算。公式为：

$$\|x_j - c\|^2 \quad (1)$$

可得公式：

$$J = \sum_{i=1}^k \sum_{x_j \in C_i} |x_j - c_i|^2 \quad (2)$$

其中， $J$  为目标函数， $k$  为聚类分量， $c_i$  为第  $i$  个聚类中心， $x \in C_i$  表示为数据点  $x_j$  属于第  $i$  个簇。

由于不同尺度的特征，会存在影响聚类效果的可能性，对此，我们要进行归一化处理，其公式为：

$$x_i = \frac{x_j - c_i}{\sigma_x} \quad (3)$$

经过这般的标准化后，我们可以使得每一个特征转化为标准差 1，均值为 0 的数值分布特征。

为了提升聚类算法中  $k$  值的选取，轮廓系数是一个有效的评估工具。它通过比较数据点在其所属簇内的紧密度与最近簇的距离来评价聚类效果。

$$S(i) = \frac{M(i) - N(i)}{\max(M(i), N(i))} \quad (4)$$

经过 K-means 聚类算法对时间段进行划分后，对附件二的数据进行汇总和处理，利用程序对现有数据进行随机取样操作，对经中路-纬一路、纬中路-景区入口、经三路-纬中路、环南路-经中路当中的经中路-纬中路方向来车的数量进行统计取样。取样后，获得相关数据如下表所示：

表 1 4 个路口的车流量表

路口	日车流量
经中路-纬一路	12747
纬中路-景区入口	8532
经三路-纬中路	8088
环南路-经中路	6503

由此可以得到通过经中路-纬中路路口的车辆向各个相位行动的概率，公式如下：

$$T_i = \frac{x_i}{x_{\text{直}} + x_{\text{左}} + x_{\text{右}}} \quad (5)$$

$i \in (\text{直}, \text{左}, \text{右})$

其中  $T_i$  为移动的概率， $x_i$  为  $i$  方向的路口。

#### 4.1.3、对问题一的模型求解

利用 python 程序，对数据进行处理。而后编辑 K-means 聚类算法程序，将相关数据导入，设定在  $k=4$  的基础前提下，对时间段进行划分。考虑到驾驶方向分别有南往北、东往西、西往东、北往南这四个方向。因此，需要对其进行综合考虑，并且，将相关图标与数据进行可视化操作，其结果如下图所示：

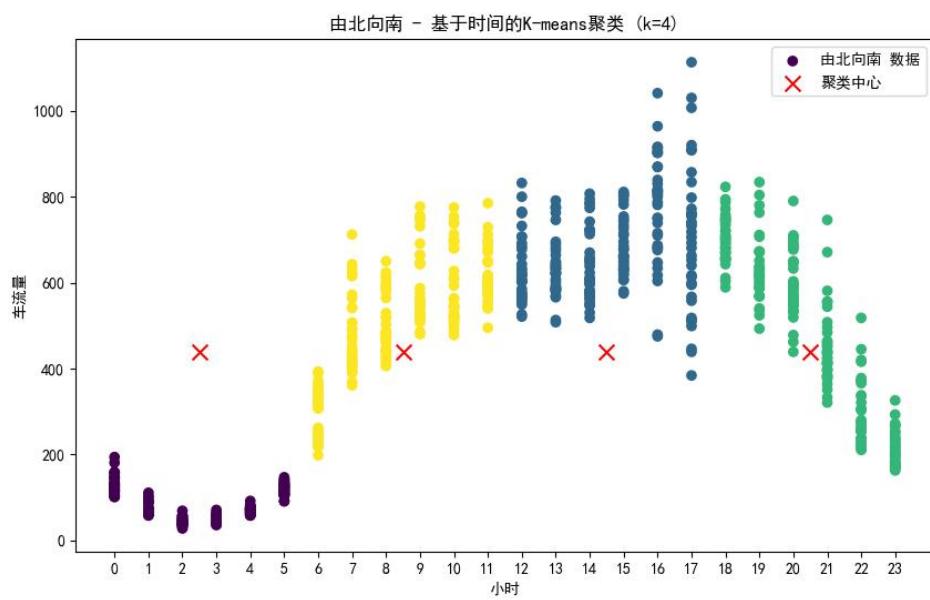


图 1 经中路-纬中路交叉口由北向南时段划分图

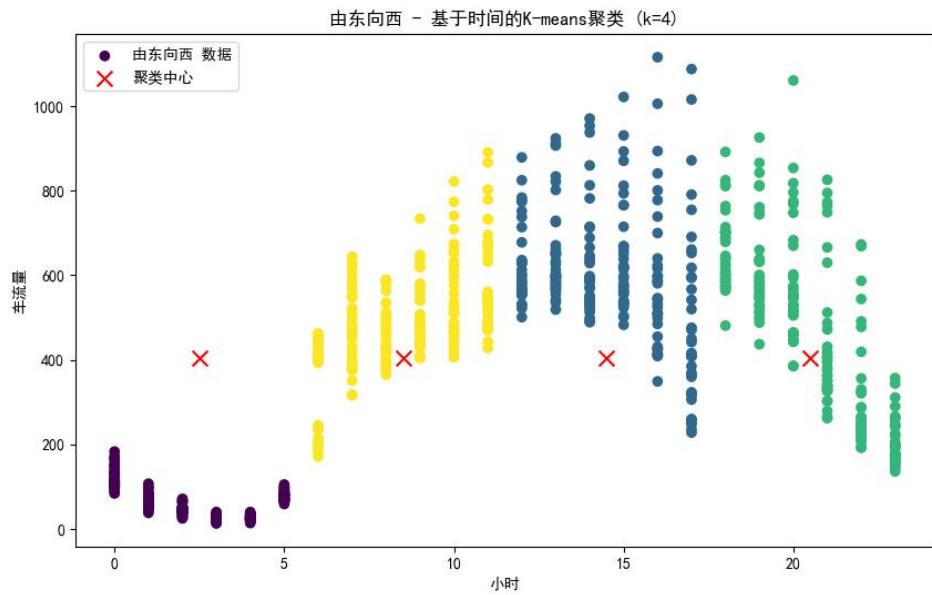


图 2 经中路-纬中路交叉口由东向西时段划分图

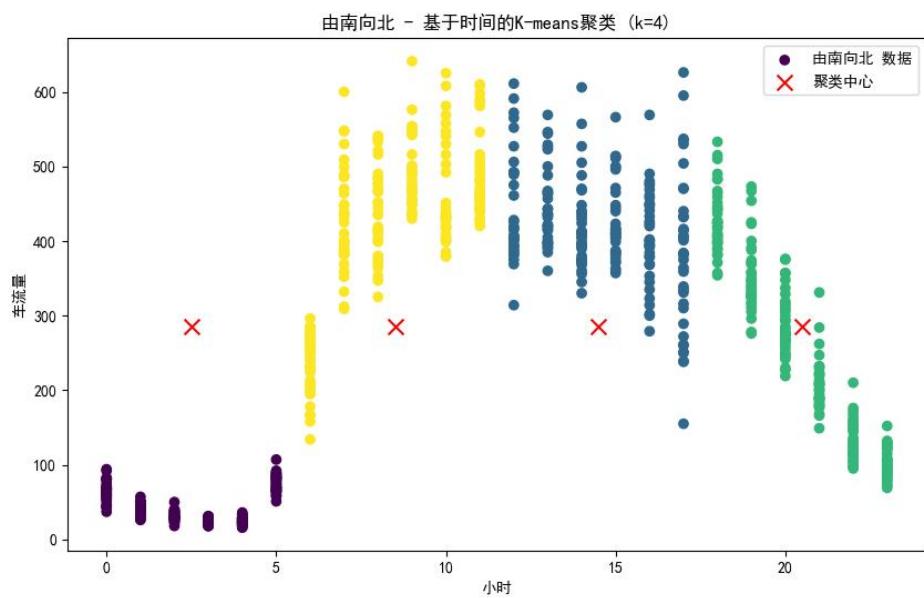


图 3 经中路-纬中路交叉口由南向北时段划分图

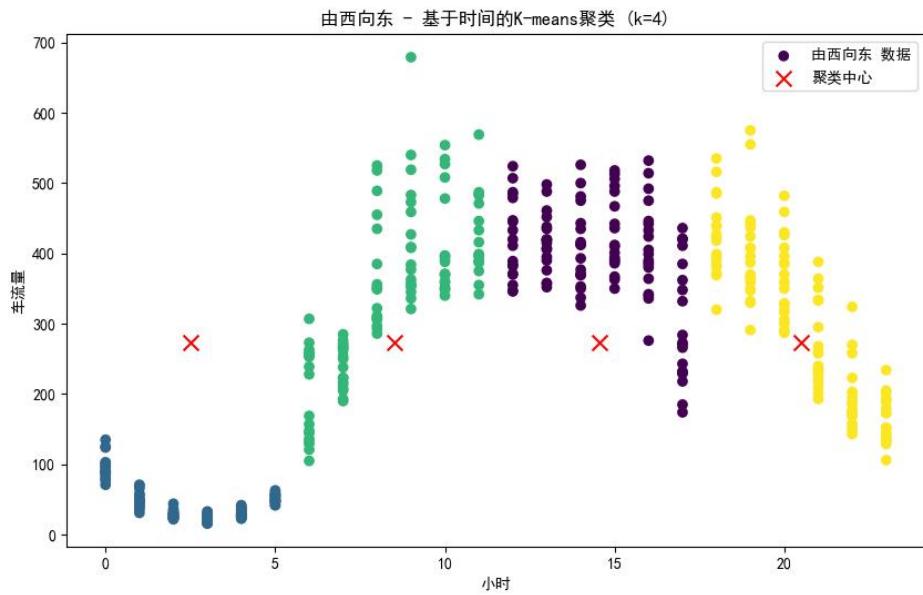


图 4 经中路-纬中路交叉口由西向东时段划分图

经由以上图标所示，可发现都满足相同的时间段划分，说明了改路口的车流量数据仅仅收到时间影响，与来车的方向没有必然的关系。并且说明了该路口的时间划分分别为凌晨静默期 0–5 点、早高峰期时期 6–11 点、午高峰期时期 12–17 点、晚高峰期时期 18–23 点。

以此四个时段为基础，对这四个时段的四个相位车流量进行估计，在结合了附件二的数据之后，得到了各个方向在各时段的车流量数据折线图，如下图所示：

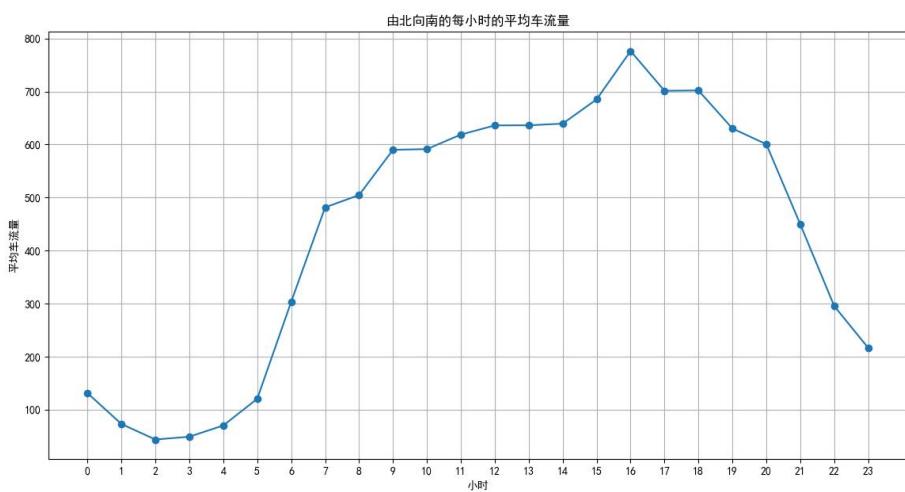


图 5 经中路-纬中路交叉口由北向南时段每小时平均车流量

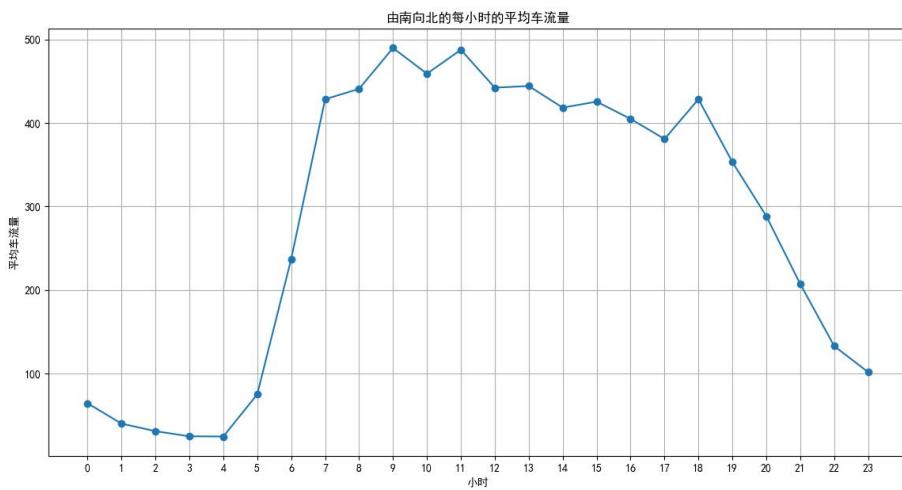


图 6 经中路-纬中路交叉口由南向北时段每小时平均车流量

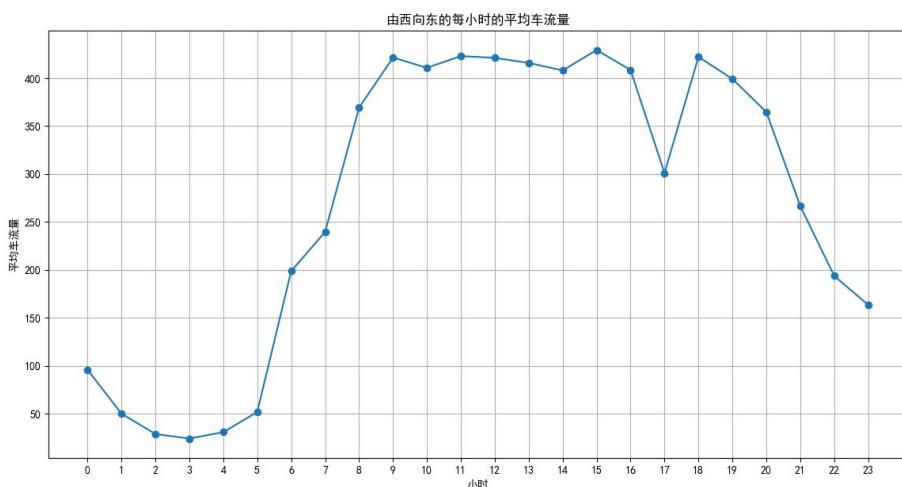


图 7 经中路-纬中路交叉口由西向东时段每小时平均车流量

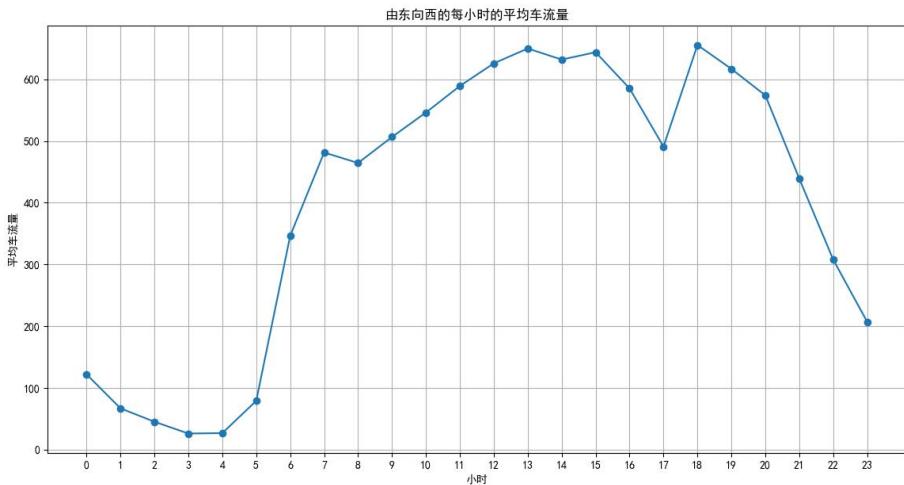


图 8 经中路-纬中路交叉口由东向西时段每小时平均车流量

综合其数据，可得其柱状图如下图所示：

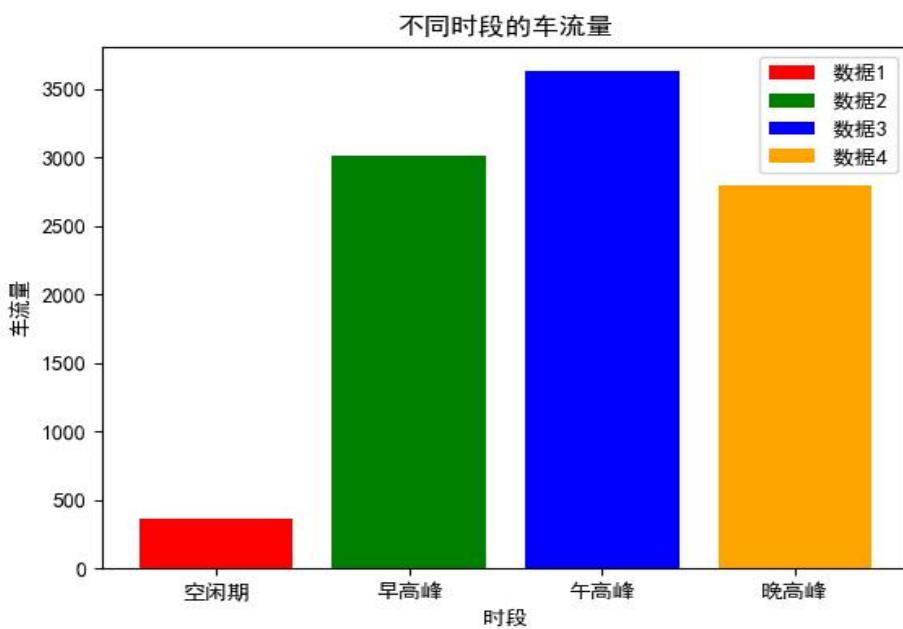


图 9 东向西相位不同时段车流量图

继续利用公式 (5) 对表 1 的数据进行计数，可以得到经中路-纬中路路口上来自各个方向直行与转弯的概率，如下表所示：

表 2 经中路-纬中路路口上来自各个方向直行与转弯的概率表

方向	概率
由南向北直行	0.45

---

由南向北左转	0.2
由南向北右转	0.35
由北向南直行	0.34
由北向南左转	0.36
由北向南右转	0.3
由东向西直行	0.24
由东向西左转	0.3
由东向西右转	0.46
由西向东直行	0.3
由西向东左转	0.42
由西向东右转	0.28

---

#### 4.1.4、对问题一模型的结果分析

根据以上模型的求解与分析，在结合所取得的数据以及可视化图标，利用 K-means 聚类算法进行分析和处理，可以将一天划分为四个时间段，分别为凌晨静默期 0-5 点、早高峰时期 6-11 点、午高峰时期 12-17 点、晚高峰时期 18-23 点。利用附件二数据，估计四个时段中各个相位的车流量数据。

其中以由东向西相位为例，具体数值如下表所示（其余参考附件）：

表 3 路口东向西相位的车流量数据

---

时间	相位	平均车流量
0-5 点	由东向西直行	87.84
0-5 点	由东向西左转	109.8
0-5 点	由东向西右转	168.36
6-11 点	由东向西直行	723.12
6-11 点	由东向西左转	903.6

---

6-11 点	由东向西右转	1,385.98
12-17 点	由东向西直行	870.48
12-17 点	由东向西左转	1,088.10
12-17 点	由东向西右转	1,668.42
18-23 点	由东向西直行	672
18-23 点	由东向西左转	840.00
18-23 点	由东向西右转	1,288.00

注：其他数据详细位于附件的其他相位车流量数据表中

## 4.2、对问题二的模型建立与求解

### 4.2.1、对问题二的分析

问题二要求对经中路和纬中路上所有交叉口的信号灯进行优化配置，以在确保车辆通行的同时，最大化两条主路上的车流平均速度。为此，我们需要对附件二的车流量数据进行相关的处理，根据不同时段调整信号灯策略，建立考虑直行和转弯影响的交通流模型，设计优化算法确定信号灯配时方案，并通过仿真测试评估效果。为了优化配给，考虑到五一假期等特殊时间所占时间不多，并且根据题目内容说明，在特殊时间内存在有相应的交通管制措施。因此，为避免数据影响，加强结论可靠性，本题不考虑五一假期等的特殊情况。

### 4.2.2、对问题二模型的建立

由于附件二数据量过于庞大，在本题目开始建模之前应该先对数据进行处理，在本题目中，考虑到了日常生活中，时间主要有工作日与休息日的区分，两个时间之间的车流量有本质上的区别，由此，我们对数据进行适当且合理的简化，只需要取每周工作日的一天与休息日的一天，便可以作为具有代表性的数据。同时，在问题一的解答中，可以发现，在每一天的凌晨 0 点-5 点，车流量极少，已经满足了通行需求，优化上限不高，效果不明显，因此可以将这一部分的数据进行排除，只取每一天的 6 点-23 点的数据即可。利用如此操作，成功实现了在不影响数据准确性与可靠性的情况下对数据做了优化。

完成数据处理后，我们利用基于车流量的信号灯配置方法进行对本题目的信号灯优化，首先，我们要对路口进行区分和辨别，主要区分为繁忙路口与非繁忙路口。对于此，我们基于《道路交通信号灯设置与安装规范》及《步行和自行车交通环境规划设计标准》等标准文件，对型号灯周期做出设定为 90 秒一个周期。且在非繁忙的正常路口上，设置时间为 30 秒绿灯，15 秒转向灯，45 秒红灯为最优配置。

而后，利用处理所得到的关于各个路口的每小时车流量数据，利用公式：

$$K_{\text{平均}} = \frac{C_{\text{总}}}{18} \quad (6)$$

式中， $K_{\text{平均}}$  为每小时平均车流量， $C_{\text{总}}$  为三个高峰期的总车流量。利用该公式结合处理所得的数据，我们可以得到各个路口分别在工作日与非工作日里的路口各个相位的每小时车流量及整个路口的每小时车流量。

在本题目情景结合《道路交通拥堵度评价方法》，推算出在本题目情景中，以每小时 400 的车流量作为界定。对与车流量小于 400 的情况判定为非繁忙的正常情况，属于非繁忙路口使用正常配置；对与车流量大于 400 的，判定为繁忙情况，属于繁忙路口，需要根据具体车流量做出相应的信号灯优化。

对繁忙路口的优化上，设定以 1、2 相位为主方向；3、4 相位为副方向。这实际中主、副方向的信号灯情况是相反的，即当主方向为绿灯时，副方向为红灯。基于此，我们设定  $X_{\text{主}}$  为主方向车流量， $Y_{\text{副}}$  为副方向车流量。利用公式：

$$t_{\text{主}} = T_{\text{灯}} \cdot \frac{X_{\text{主}}}{X_{\text{主}} + Y_{\text{副}}} \quad (7)$$

可以得到一个繁忙路口的红绿灯最优分配时间，式中  $t_{\text{主}}$  为主方向绿灯时间， $T_{\text{灯}}$  信号灯周期。

对与绿灯方向上，需要细分出左右转弯的车流，而在生活实际当中，右转车流可以直接转弯，不考虑信号灯问题。因此，我们单独对左转进行分析。在问题一中，我们得到了路口关于各个相位上左转的概率，因此利用公式：

$$Z_{\text{左}} = G_{\text{左}} \cdot L_{\text{车}} \quad (8)$$

可以得到对应相位当中转弯的车流量。式中  $Z_{\text{转}}$  为左转车流量,  $G_{\text{左}}$  为对应相位左转的概率 (见表 2),  $L_{\text{车}}$  为对应相位的车流量。

同时将公式 (8) 做简单变换可以得到对应相位直行的车流量  $X_{\text{直}}$  再利用公式:

$$t_{\text{转}} = \frac{Z_{\text{左}}}{Z_{\text{左}} + X_{\text{直}}} \cdot t_{\text{绿}} \quad (9)$$

#### 4.2.3、对问题二模型的求解与结果分析

利用 python 编程对数据处理后, 我们用上述建模, 制作了详尽的交通信号灯时间优化配置数据, 涵盖了多个交叉路口在一天中不同时间段的红绿灯设置。我们将每个交叉路口都分为四个主要时段: 凌晨 (0 点到 5 点)、早晨 (6 点到 11 点)、下午 (12 点到 17 点) 和晚上 (18 点到 23 点), 根据车流量, 并结合题目要求为每个时段分别记录了主方向和副方向的绿灯时间 (包括直行和转向灯时间) 以及相应的红灯时间。以环西路-纬中路为例子, 其在凌晨时段, 考虑到车流量少, 属于非繁忙的普通情况, 于是按照标准给出了: 主方向绿灯持续 30 秒, 副方向也是 30 秒; 而到了早晨时段, 车流量增加, 变为了繁忙路口, 属于繁忙情况, 于是按照公式 (7) (8) (9) 进行计算, 得出结果为: 主方向绿灯时间增加到 33 秒, 副方向减少到 28 秒。以确保交通的顺畅和安全。满足出行的需求, 如此便满足题目要求, 在保证了车辆通行的前提下, 使得两条主路上的车流平均速度最大。

下面以环西路-纬中路在工作日的安排为例, 制成具体的交通灯优化配置表如下所示 (其余表单详情在附录中, 请见《休息日各路口红绿灯优化配置表》与《工作日各路口红绿灯优化配置表》):

表 4 工作日下环西路-纬中路交通灯优化配置表

交叉路口	时段	主方向绿灯时间		主方向红灯时间	副方向绿灯时间		副方向红灯时间
		直行灯时间	转向灯时间		直行灯时间	转向灯时间	
环西路-纬中路	0点到5点	30	15	45	30	15	45
	6点到11点	33	18	51	28	11	39
	12点到17点	36	20	56	24	10	34
	18点到23点	33	18	51	28	11	39

结合机器学习，对最终的信号灯优化方案做多次模拟，结果说明优化明显，车流通过率提高，在满足了车辆正常通行不堵车的前提下，实现了车流均速最大。

### 4.3、对问题三的模型建立与求解

#### 4.3.1、对问题三的分析

问题三要求分析附件二当中五一黄金周期间的数据，以确定寻找停车位的巡游车辆数量，并估算假期景区需要临时征用多少停车位才能满足需求。这需要对附件二中提供的车流数据进行详细分析，特别是关注那些在景区周边道路上来回低速绕行的车辆行为模式。通过识别这些车辆的特征（如频繁在同一区域内出现、长时间停留在特定路段等），可以估计出寻找停车位的车辆数量。然后，根据这一估计值和景区现有的停车设施容量，可以进一步计算出为了满足游客需求，景区需要额外征用的停车位数量。

### 4.3.2、对问题三模型的建立

考虑到假期出游的时间因数，由于最后一天为假期末尾，车辆离开多，到来少，景区车流量减少，对停车位的需求回归到正常情况，因此可以仅仅对五一前四天进行分析

对于巡游车辆的判定，我们首先需要确定巡游车辆会具有哪一些行为。结合到生活实际与题目信息及要求，我们可以判定：由于没有停车空间，且驾驶员注意力集中与车位的寻找，所以巡游车辆往往会在某一区域内频繁移动、速度较慢的或反复的经过同一条路线。因此，我们可以继续利用附件二，对其涉及时间差、移动距离、方向变化的数据内容运用 K-Means 聚类分析进行判别。

为了方便分析，我们首先需要对数据进行特征提取，并进行标记。内容如下表所示：

表 5 数据特征标记表

标记目标	标记内容
时间差	较短的时间差可能表明车辆在频繁巡回
移动距离	较短的移动距离可能表明车辆在小范围内巡游
方向变化	方向变化较频繁的车辆可能是巡游车辆，因为它们可能在寻找停车位时频繁转弯

与问题一建模相同，运用 K-Means 聚类分析我们需要计算每一个数据点与当前所有簇的质心的距离。对此，应用距离公式：

$$d(x_1, \mu) = \sqrt{\sum_{i=1}^n (x_{1i} - \mu_i)^2} \quad (10)$$

式中  $x_1$ =(时间差, 移动距离, 方向变化) 与各个簇质心  $\mu$  的距离。而关于质心  $\mu$  公式如下：

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x \quad (11)$$

式中  $\mu_j$  表示第  $j$  个簇的质心。

再设立目标函数  $J_{\text{损}}$  用于衡量聚类结果的质量。每一次的迭代都是通过最小化这个目标函数来优化质心的位置。

$$J_{\text{损}} = \sum_{j=1}^k \sum_{x \in C_j} d(x, \mu_j)^2 \quad (12)$$

在面对临时车位需求的计算上，我们可以利用微积分，取出每天某一个时段中，最大的停车需求，即为在一个停车位完成一辆车停靠的平均所需求的时间，以这个时间为时间段，寻找整一天内，在这个时间段有最多的巡游车辆的位置。只要满足这个时间段的停车需求就可以绝对满足一整天的停车需求。通过滑动窗口法依次计算每个时间段  $i$  的总和  $A_i$ ，找到最大的值，即为当天的最大面积。相关公式为：

$$A_i = \sum C_j (j \leq i \leq j+n-1) \quad (13)$$

其中  $A_i$  为当天最大的停车需求量， $C_j$  是第  $j$  个时间间隔内的车辆数量，而对于时间间隔数，再设  $J_{\text{间隔数}}$  为时间间隔的数量， $G_{\text{停}}$  为每一辆车的平均停车时间， $n$  为每一段时间巡回车辆统计数，其计算公式为：

$$J_{\text{间隔数}} = \frac{G_{\text{停}}}{n_{\text{间隔}}} \quad (14)$$

#### 4.3.3、对问题三模型的求解

利用 K-Means 聚类分析我们判定出了巡游车辆，具体分析出有三种数据特征，具体有下表

表 6 数据特征表

种类	特征点	内容
Cluster 0	时间差	8466 秒(约 2.35 小时)
	移动距离	2.39
	方向变化	1.62
Cluster 1	时间差	118821 秒(约 33 小时)
	移动距离	1.32
	方向变化	0.98
Cluster 2	时间差	3876 秒(约 1.08 小时)
	移动距离	0.45
	方向变化	0.26

对于 Cluster 0，其车辆在交叉口之间经过的时间较长，在交叉口之间移动距离相对较大，方向变化较频繁。说明这个簇的车辆特征可能是正常行驶的车辆，它们在较长的时间内经过多个交叉口，并且改变方向较多。

对于 Cluster 1，其车辆在交叉口之间经过的时间非常长。移动距离适中，方向变化相对较少，这个簇的车辆可能是长时间不再出现在交叉口的车辆，可能是停靠后再继续行驶的情况。

对于 Cluster 2，时间较短，表明车辆频繁经过交叉口，车辆移动的距离非常短。方向变化较少。这个簇的车辆可能是巡游车辆，因为它们在短时间内移动距离较小并且频繁经过交叉口。

由此可以判定 Cluster 2 可能是最符合巡游车辆特征的簇，因为它们的时间差较小、移动距离较短且频繁经过更交叉口。

而后利用 python 编写相关程序对景区出入口的数据进行分析，寻找出进入景区的车辆的平均的停车时间为 2.87 个小时。

在根据生活经验与数据分析，认为大约每一辆车的巡游时间是 15 分钟，因此我们以 15 分钟为时间段，统计每 15 分钟的巡游车辆的数量，将其中结果进行可视化处理，得到结果如下图所示：

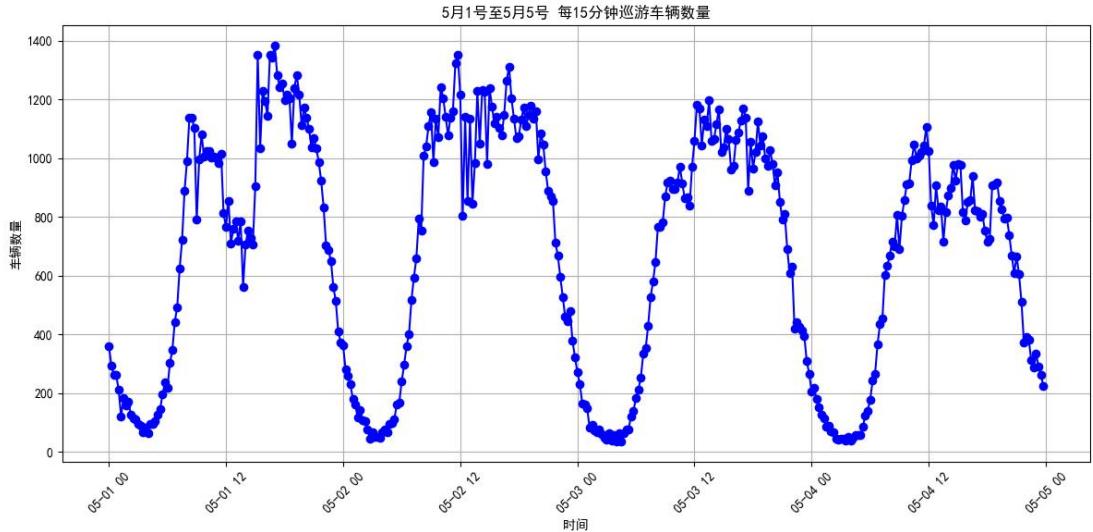


图 10 每 15 分钟巡游车辆数据统计

得到结果后，为设定景区借用临时停车位以天为单位，那么分析一天当中，最大的停车需求，即可做到在满足每日停车需求的前提下，实现资源利用效率最大化，节省了大量的资源。利用上述建模内容中的微积分内容，带入数值到公式（13）（14）得到结果如下图所示：

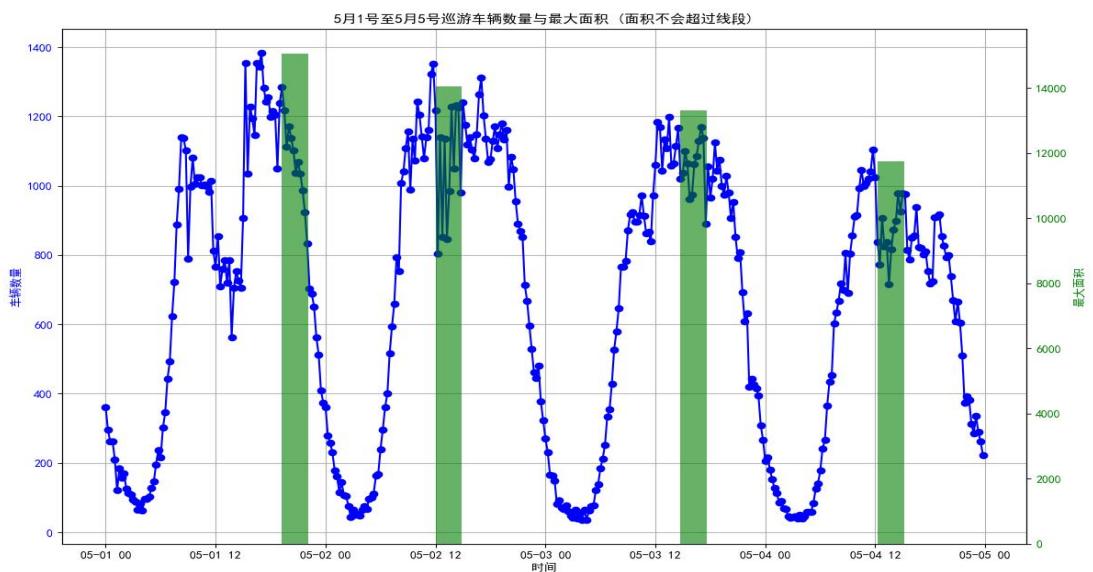


图 11 每天 2.87 小时内最多巡游车辆位置分布图

具体数据如下表所示：

表 7 每日临时停车位需求量

日期	临时车位需求量	峰值时间
2024/5/1	311	2024/5/1 20:52
2024/5/2	403	2024/5/2 16:07
2024/5/3	359	2024/5/3 17:22
2024/5/4	218	2024/5/4 18:52

由此得出结论，在五一黄金周期间，由于最后一天为假期末尾，车辆离开多，到来少，景区车流量减少，对停车位的需求回归到正常情况，因此对五一前四天进行分析，得出结论为在5月1日增加311个停车位，5月2日增加403个停车位，5月3日增加359个停车位，5月4日增加218个停车位。即可满足需求。

## 4.4、对问题四的模型建立与求解

### 4.4.1、对问题四的分析

问题四的背景是针对五一黄金周期间，小镇对景区周边道路实行的临时性交通管理措施。这些管理措施可能包括限行、分流等策略，旨在缓解假期期间由于游客增多导致的交通压力。目标则是评估这些临时管控措施在两条主路上的效果，例如，与平常的工作日，休息日相比，五一的节假日在车流量，平均车流速度以及环游车辆比例的变化等等。

### 4.4.2、对问题四模型的建立

根据题目要求，我们需要对五一黄金周期间，小镇周边的临时交通管制措施在两条主路上的效果。由此，我们需要建立相关的评价模型。可以利用附件二中，关于小镇在非五月黄金周的车流数据与五月黄金周的数据进行比对。在这里我们选用了相对典型的数据，比如随机选用了具有代表性的小镇工作日的数据，休息日的数据已经假期数据。

明确了数据的要求，接下来我们要确定评价的指标与对象。在本题目附件中，可以用于评价的数据有车流量，平均车流速度以及环游车辆的占比作为评价的指标，而五一黄金周期间为本题目评价的对象。

对此我们使用熵权法对数据进行处理，使 TOPSIS 方法对结果进行整合。对于熵权法，我们通过各个数据的信息熵，来对每个指标的重要性进行判断，进而分配权重。具体公式如下，从正向指标上：

$$X_i' = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \quad (15)$$

对与反向指标上：

$$X_i' = \frac{X_{\max} - X_i}{X_{\max} - X_{\min}} \quad (16)$$

对与比例  $P_{ij}$  的计算公式：

$$P_{ij} = \frac{X_{ij}'}{\sum_{i=1}^n X_{ij}'} \quad (17)$$

对与信息熵  $E_j$  的计算公式：

$$E_j = -k \sum_{i=1}^n P_{ij} \ln P_{ij}, \quad k = \frac{1}{\ln n} \quad (18)$$

权重计算公式：

$$W_j = \frac{1 - E_j}{\sum_{j=1}^m (1 - E_j)} \quad (19)$$

对于熵权法计算，为了取得得出相对接近度  $C_i$ ，我们需要使用 TOPSIS 方法通过计算每个方案与理想解和负理想解的距离，公式如下：

加权值为：

$$X_i' \times W_j \quad (20)$$

对于理想解和负理想解的计算公式：

$$A^+ = (\max X_1, \min X_2), \quad A^- = (\min X_1, \max X_2) \quad (21)$$

对于距离上的计算：

$$D^+ = \sqrt{\sum_{i=1}^n (X_i' - A^+)^2}, \quad D^- = \sqrt{\sum_{i=1}^n (X_i' - A^-)^2} \quad (22)$$

最后是对相似接近度的计算公式：

$$C_i = \frac{D^-}{D^+ + D^-} \quad (23)$$

#### 4.4.3、对问题四模型的求解

根据以上模型方法，从附件中分别提取出工作日、休息日、节假日的相关数据，以作为评价模型层次分析法的数据，具体如下表所示：

表 8 工作日、休息日、节假日数据内容表

日期	一天总车流量	车速	巡游车辆占总车辆占比	一天通过总车辆	巡游车辆数	峰值车流量(单位: 小时)	平均车流量
工作日	114751	21.27	0.000185358	55020	15986	9554	4781
休息日	90790	20.83	0.000229431	47050	10502	7313	3783
节假日	126983	17.31	0.000136317	64884	13990	9952	5391

将这些数据，导入到制成的熵权法运算的 Python 代码与运算 TOPSIS 方法的 Python 代码中，进行数据的分析与评价，由计算得出结果，如下表所示：

表 9 工作日、休息日、节假日数据对比与评价结果表

---

7

2

0	0.000	0.888	0.412	0.538	0.412	0.433
.	2294	8888	2505	6885	2505	5193
8	31	89	34	74	34	81

3

1

7	0.000	0	1	0	0.536	0.463	0.536	0.536
.	1363	0	1	0	2181	7818	2181	2181
3	17				49	51	49	49

1

---

结果上，相对接近度 (TOPSIS)越接近 1，则效果越好，由此根据表格数据，得出结论，五一黄金周期间，该小镇对景区周边道路实行的临时性交通管理措施，使得在车流量最大（与对照组相比）的情况下，却让路口的交通通行能力强于正常情况下的休息日，而弱于正常情况下的工作日。由此达到了交通管制的目的，因此，可以说明临时管控措施在两条主路上的对到了通行优化效果明显。

## 五、模型评价与改进

本篇论文通过 K-means 聚类算法划分时间段，提升了动态时间段的划分效果，但可以尝试其他聚类方法提高稳健性；交通信号灯优化模型有效提升车流速度，可以引入自适应信号灯技术优化时长；停车位需求预测合理，但也可以加入更多影响因素或许以提高预测精度；临时交通管控评估通过熵权法和 TOPSIS 整合多项指标，结果有效，但可以通过增加实地数据补充评估。整体模型合理，具有应用价值，但可通过优化算法和引入更多影响因素提升效果。

## 六、参考文献

[1]李伟龙, 张晓晴, 胡雅洁, 等. 基于 K-means 聚类算法的负荷峰谷时段划分[J]. 电气开关, 2024, 62(04):29–31.

[2]李智猛. 基于车流量检测的智能交通信号配时优化研究[D]. 浙江科技学院, 2022. DOI:10.27840/d.cnki.gzjkj.2022.000300.

- [3]GB 14886-2016, 道路交通信号灯设置与安装规范[S].
- [4]GA/T 115-2020, 道路交通拥堵度评价方法[S].
- [5]赵新哲, 曲朝晖, 杨长青. 基于熵权-TOPSIS-秩和比法的国企医院运营评价研究[J]. 中国医院, 2024, 28(07) : 71-75. DOI:10.19660/j.issn.1671-0592.2024.7.17.

## 七、附录

### 附录一、支撑材料目录

文件	说明
工作日的各路口红绿灯优化配置表.xlsx	对问题二工作日期间信号灯优化的具体配置信息表
休息日的各路口红绿灯优化配置表.xlsx	对问题二休息日期间信号灯优化的具体配置信息表
由北向南相位车流量数据表.xlsx	对问题一经中路-纬中路交叉口上由北向南相位上直行与转弯在四个时段上的具体流量数据表
由东向西相位车流量数据表.xlsx	对问题一经中路-纬中路交叉口上由东向西相位上直行与转弯在四个时段上的具体流量数据表
由南向北相位车流量数据表.xlsx	对问题一经中路-纬中路交叉口上由南向北相位上直行与转弯在四个时段上的具体流量数据表
由西向东相位车流量数据表.xlsx	对问题一经中路-纬中路交叉口上由西向东相位上直行与转弯在四个时段上的具体流量数据表
代码.zip	包含了模型建立与公式方法的所有代码

### 附录二、Python(3.12.1)问题一数据提出代码

```
import pandas as pd

# 读取 CSV 文件
df = pd.read_csv('E 题\附件 2.csv', encoding='gbk')
# 数据清理
df = df[~df['车牌号'].isin(['无车牌', 'unnnowk'])]

# 筛选经中路-纬中路交叉路口数据
jzl_wzl_df = df[df['交叉口'].str.contains('经中路-纬中路', na=False)]
jzl_wzl_df_1 = jzl_wzl_df.iloc[:1000000, :]
jzl_wzl_df_2 = jzl_wzl_df.iloc[1000000:, :]

with pd.ExcelWriter('1、第一题\Data\经中路-纬中路(没有相位分组) .xlsx',
engine='openpyxl') as writer:
    jzl_wzl_df_1.to_excel(writer, sheet_name='第一个表格', index=False)
```

```

jzl_wzl_df_2.to_excel(writer, sheet_name='第二个表格', index=False)

# 对经中路-纬中路交叉路口数据按相位分组
df['时间'] = pd.to_datetime(df['时间'], format='%Y-%m-%dT%H:%M:%S.%f')
df['方向'] = df['方向'].astype(str)

east_weat_1_df = df[df['方向'].str.contains('1', na=False)]
west_east_2_df = df[df['方向'].str.contains('2', na=False)]
south_north_3_df = df[df['方向'].str.contains('3', na=False)]
north_south_4_df = df[df['方向'].str.contains('4', na=False)]
# 对分组后的数据按时间排序
east_weat_1_df_sorted = east_weat_1_df.sort_values(by='时间')
west_east_2_df_sorted = west_east_2_df.sort_values(by='时间')
south_north_3_df_sorted = south_north_3_df.sort_values(by='时间')
north_south_4_df_sorted = north_south_4_df.sort_values(by='时间')
# 保存数据到 Excel 文件
with pd.ExcelWriter('Data\经中路-纬中路(相位分组).xlsx', engine='openpyxl') as writer:
    east_weat_1_df.to_excel(writer, sheet_name='由东向西', index=False)
    west_east_2_df.to_excel(writer, sheet_name='由西向东', index=False)
    south_north_3_df.to_excel(writer, sheet_name='由南向北', index=False)
    north_south_4_df.to_excel(writer, sheet_name='由北向南', index=False)

# 筛选直行转弯分析数据
jin3_weizhong_1_df = df[(df['方向'] == 1) & (df['交叉口'] == '经三路-纬中路')]
weizhong_2_df = df[(df['方向'] == 2) & (df['交叉口'] == '纬中路-景区出入口')]
jingzhong_wei1_3_df = df[(df['方向'] == 3) & (df['交叉口'] == '经中路-纬一路')]
huannan_jingzhong_4_df = df[(df['方向'] == 4) & (df['交叉口'] == '经中路-环南路')]

with pd.ExcelWriter('Data\直行转弯分析数据.xlsx', engine='openpyxl') as writer:
    jin3_weizhong_1_df.to_excel(writer, sheet_name='经三路-纬中路', index=False)
    weizhong_2_df.to_excel(writer, sheet_name='纬中路-景区出入口', index=False)
    jingzhong_wei1_3_df.to_excel(writer, sheet_name='经中路-纬一路',
                                index=False)
    huannan_jingzhong_4_df.to_excel(writer, sheet_name='环南路-经中路',
                                    index=False)

```

### 附录三、Python(3.12.1)问题一每天各小时车流量的统计

```

import pandas as pd

# 每天各小时车流量的统计
def read_and_process_sheet(file_path, sheet_name):

```

```

df = pd.read_excel(file_path, sheet_name=sheet_name, engine='openpyxl')
df['时间'] = pd.to_datetime(df['时间'])
df['日期'] = df['时间'].dt.date
df['小时'] = df['时间'].dt.hour
return df.groupby(['日期', '小时'])['车牌号']
].count().reset_index().rename(columns={'车牌号': '车流量'})

# 数据
file_path = 'Data/经中路-纬中路(相位分组).xlsx'
sheets = ['由东向西', '由西向东', '由南向北', '由北向南']

# 处理每个工作表
traffic_data = {}
for sheet in sheets:
    traffic_data[sheet] = read_and_process_sheet(file_path, sheet)
# 写入到 Excel 文件
output_file = 'Data/经中-纬中每天各小时车流量统计.xlsx'
with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
    for sheet_name, data in traffic_data.items():
        data.to_excel(writer, sheet_name=sheet_name, index=False)

```

#### 附录四、Python(3.12.1)问题一 K-means 聚类分时段

```

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
# 配置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei'] # 正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题

# 数据
file_path = '1、第一题\Data\4、经中-纬中每天各小时车流量统计.xlsx'
sheet_names = ['由北向南', '由东向西', '由南向北', '由西向东']

# 遍历每个方向的表
for sheet_name in sheet_names:
    # 读取数据
    data = pd.read_excel(file_path, sheet_name=sheet_name)
    # 只使用时间（小时）进行 K-means 聚类
    hours = data[['小时']].values
    # 使用 KMeans 聚类模型进行分时段，k=4
    kmeans = KMeans(n_clusters=4, random_state=0)
    kmeans.fit(hours)

```

```

# 获取聚类标签和聚类中心
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# 可视化结果
plt.figure(figsize=(10, 6))
plt.scatter(data['小时'], data['车流量'], c=labels, cmap='viridis',
label=f'{sheet_name} 数据')
plt.scatter(centers[:, 0], [data['车流量'].mean() * len(centers)], c='red', marker='x',
s=100, label="聚类中心")
plt.xlabel('小时')
plt.xticks(range(0, 24))
plt.ylabel('车流量')
plt.title(f'{sheet_name} - 基于时间的 K-means 聚类 (k=4)')
plt.legend()
plt.savefig(f'24E 代码/Png/{sheet_name} - 基于时间的 K-means 聚类
(k=4).png')
plt.show()

```

## 附录五、Python(3.12.1)问题一车流量的分析

```

import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

xianwei = ['无','由东向西','由西向东','由南向北','由北向南']
def process_traffic_data(file_path,sheet_name,xw):
    df = pd.read_excel(file_path,sheet_name=sheet_name, engine='openpyxl')
    df['时间'] = pd.to_datetime(df['时间'])
    df['日期'] = df['时间'].dt.date
    df['小时'] = df['时间'].dt.hour
    traffic_per_hour = df.groupby(['日期', '小时'])['车牌号'].count().reset_index()
    traffic_per_hour.columns = ['日期', '小时', '车流量']

    # 计算每小时的平均车流量
    average_traffic_per_hour = traffic_per_hour.groupby('小时')['车流量
    ].mean().reset_index()
    average_traffic_per_hour.columns = ['小时', '平均车流量']
    # 计算每小时的车流量总数量
    total_traffic_per_hour = traffic_per_hour.groupby('小时')['车流量
    ].sum().reset_index()
    total_traffic_per_hour.columns = ['小时', '车流量总数']

```

```

merged_df = pd.merge(total_traffic_per_hour, average_traffic_per_hour, on='小时')

# 结果可视化
plt.figure(figsize=(14, 7))
plt.plot(average_traffic_per_hour['小时'], average_traffic_per_hour['平均车流量'],
         marker='o')
plt.title(xianwei[xw] + '的每小时的平均车流量')
plt.xlabel('小时')
plt.ylabel('平均车流量')
plt.xticks(range(0, 24))
plt.grid(True)
plt.savefig('Png\\' + xianwei[xw] + '的每小时的平均车流量' + '.png')
plt.show()

return merged_df

# 调用函数求解平均车流量
east_weat_1_df = process_traffic_data('Data\经中路-纬中路(相位分组).xlsx', '由东向西', 1)
west_east_2_df = process_traffic_data('Data\经中路-纬中路(相位分组).xlsx', '由西向东', 2)
south_north_3_df = process_traffic_data('Data\经中路-纬中路(相位分组).xlsx', '由南向北', 3)
north_south_4_df = process_traffic_data('Data\经中路-纬中路(相位分组).xlsx', '由北向南', 4)

with pd.ExcelWriter('Data\经中路-纬中路每天数据汇总.xlsx', engine='openpyxl') as writer:
    east_weat_1_df.to_excel(writer, sheet_name='由东向西', index=False)
    west_east_2_df.to_excel(writer, sheet_name='由西向东', index=False)
    south_north_3_df.to_excel(writer, sheet_name='由南向北', index=False)
    north_south_4_df.to_excel(writer, sheet_name='由北向南', index=False)

# 结果可视化
data = [366, 3013, 3628, 2800]
labels = ['空闲期', '早高峰', '午高峰', '晚高峰']
colors = ['red', 'green', 'blue', 'orange']
for i in range(4):
    plt.bar(labels[i], data[i], color=colors[i], label=f'数据 {i+1}')
plt.xlabel('时段')
plt.ylabel('车流量')
plt.title('不同时段的车流量')

```

```
plt.legend()
plt.savefig('Png\\不同时段的车流量.png')
plt.show()
```

## 附录六、Python(3.12.1)问题二整合代码

```
import pandas as pd
import openpyxl

# 筛选日期的数据
def extract_data_by_date(df, target_date):
    df['时间'] = pd.to_datetime(df['时间'])
    return df[df['时间'].dt.date == pd.to_datetime(target_date).date()]

# 写入 Excel
def write_to_excel(df_dict, output_path):
    with pd.ExcelWriter(output_path, engine='openpyxl') as writer:
        for sheet_name, data in df_dict.items():
            data.to_excel(writer, sheet_name=sheet_name, index=False)

# 统计交叉口和方向的车流量按小时分布
def traffic_summary_by_crossroad(df):
    df['小时'] = df['时间'].dt.hour
    summary = {}

    for crossroad in df['交叉口'].unique():
        crossroad_df = df[df['交叉口'] == crossroad]

        for direction in crossroad_df['方向'].unique():
            direction_df = crossroad_df[crossroad_df['方向'] == direction]
            hourly_traffic = direction_df.groupby('小时').size().reset_index(name='车流量')
            sheet_name = f'{crossroad}_{direction}'
            summary[sheet_name] = hourly_traffic

    return summary

# 进行时间段分类
def categorize_and_summarize_traffic(df):
    df['小时'] = df['时间'].dt.hour

    def categorize_time(hour):
        if 0 <= hour < 6:
            return '0-5'
```

```

        elif 6 <= hour < 12:
            return '6-11'
        elif 12 <= hour < 18:
            return '12-17'
        else:
            return '18-23'

    df['时间段'] = df['小时'].apply(categorize_time)
    return df.groupby('时间段')[‘车流量’].sum().reset_index()

# 处理 Excel 所有表单
def process_excel_file(file_path):
    xls = pd.ExcelFile(file_path, engine='openpyxl')
    all_data = {}

    for sheet_name in xls.sheet_names:
        try:
            df = pd.read_excel(xls, sheet_name=sheet_name)
            summary = categorize_and_summarize_traffic(df)
            all_data[sheet_name] = summary
        except Exception as e:
            print(f'Error processing sheet {sheet_name}: {e}')

    return all_data

# 主流程
df = pd.read_csv('E 题/附件 2.csv', encoding='gbk')

# 提取 4 月 19 日和 20 日的数据
df_4_19_work = extract_data_by_date(df, '2024-04-19')
df_4_20_week = extract_data_by_date(df, '2024-04-20')

# 写入初步筛选的数据
write_to_excel({'4 月 19 日工作日': df_4_19_work, '4 月 20 日周末': df_4_20_week}, 'Data/量化数据.xlsx')

# 处理交叉路口车流量数据
crossroad_4_19_data = traffic_summary_by_crossroad(df_4_19_work)
crossroad_4_20_data = traffic_summary_by_crossroad(df_4_20_week)
write_to_excel(crossroad_4_19_data, '2、第二题\Data\4 月 19 日各交叉路口数据.xlsx')
write_to_excel(crossroad_4_20_data, '2、第二题\Data\4 月 20 日各交叉路口数据.xlsx')

```

```

# 处理并总结时间段的车流量数据
file_path = '第二题/4月19日各交叉路口数据.xlsx'
summary_data = process_excel_file(file_path)
write_to_excel(summary_data, '第二题/4月19日各交叉路口数据汇总.xlsx')

file_path = '第二题/4月20日各交叉路口数据.xlsx'
summary_data = process_excel_file(file_path)
write_to_excel(summary_data, '第二题/4月20日各交叉路口数据汇总.xlsx')

```

## 附录七、Python(3.12.1)问题三 K-means 算法对巡游车辆进行聚类

```

# 导入必要的库
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt

# 加载数据
file_path = '3、第三题\五个交叉口.xlsx'
data = pd.read_excel(file_path)

# 数据处理与特征提取
# 计算移动距离（基于坐标）
data['坐标'] = data['坐标'].apply(lambda x: x.strip(')').split(', '))
data[['坐标_x', '坐标_y']] = pd.DataFrame(data['坐标'].tolist(), index=data.index)
data['坐标_x'] = pd.to_numeric(data['坐标_x'])
data['坐标_y'] = pd.to_numeric(data['坐标_y'])
data['移动距离'] = np.sqrt((data['坐标_x'] - data.groupby('车牌号')['坐标_x'].shift(1))**2 +
                           (data['坐标_y'] - data.groupby('车牌号')['坐标_y'].shift(1))**2)

# 时间
data['时间'] = pd.to_datetime(data['时间'])
data = data.sort_values(by=['车牌号', '时间'])
data['时间差'] = data.groupby('车牌号')['时间'].diff().dt.total_seconds()

# 方向变化
data['方向变化'] = data.groupby('车牌号')['方向'].diff().fillna(0).abs()
# 将 NaN 值填充为 0
data['移动距离'] = data['移动距离'].fillna(0)
data['时间差'] = data['时间差'].fillna(0)

```

```
# 特征标准化
features = data[['时间差', '移动距离', '方向变化']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# K-Means 模型训练
kmeans = KMeans(n_clusters=3, random_state=0)
data['cluster'] = kmeans.fit_predict(scaled_features)

# 结果分析
# 计算每个簇的统计数据
cluster_analysis = data.groupby('cluster').mean()[['时间差', '移动距离', '方向变化']]
print(cluster_analysis)
# 结果可视化
plt.figure(figsize=(10, 6))
for cluster in range(3):
    cluster_data = data[data['cluster'] == cluster]
    plt.scatter(cluster_data['移动距离'], cluster_data['时间差'], label=f'Cluster {cluster}', alpha=0.6)
    plt.yscale('log')
    plt.xscale('log')
    plt.title('Cluster Visualization with Logarithmic Scaling')
    plt.xlabel('移动距离 (log scale)')
    plt.ylabel('时间差 (log scale)')
    plt.legend()
    plt.show()

# 每个簇的详细分析（关注 Cluster 2）
cluster_2 = data[data['cluster'] == 2]

# 时间差分析
time_diff_mean = cluster_2['时间差'].mean()
time_diff_median = cluster_2['时间差'].median()
time_diff_std = cluster_2['时间差'].std()

# 打印时间差统计结果
print(f"Cluster 2 时间差的均值: {time_diff_mean}")
print(f"Cluster 2 时间差的中位数: {time_diff_median}")
print(f"Cluster 2 时间差的标准差: {time_diff_std}")

# 可视化时间差分布
plt.figure(figsize=(10, 6))
sns.histplot(cluster_2['时间差'], kde=True)
plt.title('Cluster 2 时间差分布')
```

```
plt.xlabel('时间差 (秒)')
plt.ylabel('频率')
plt.show()

# 移动距离分析
move_dist_mean = cluster_2['移动距离'].mean()
move_dist_median = cluster_2['移动距离'].median()
move_dist_std = cluster_2['移动距离'].std()

# 打印移动距离统计结果
print(f"Cluster 2 移动距离的均值: {move_dist_mean}")
print(f"Cluster 2 移动距离的中位数: {move_dist_median}")
print(f"Cluster 2 移动距离的标准差: {move_dist_std}")

# 可视化移动距离分布
plt.figure(figsize=(10, 6))
sns.histplot(cluster_2['移动距离'], kde=True)
plt.title('Cluster 2 移动距离分布')
plt.xlabel('移动距离')
plt.ylabel('频率')
plt.show()

# 方向变化分析
direction_change_mean = cluster_2['方向变化'].mean()
direction_change_median = cluster_2['方向变化'].median()
direction_change_std = cluster_2['方向变化'].std()

# 打印方向变化统计结果
print(f"Cluster 2 方向变化的均值: {direction_change_mean}")
print(f"Cluster 2 方向变化的中位数: {direction_change_median}")
print(f"Cluster 2 方向变化的标准差: {direction_change_std}")

# 可视化方向变化分布
plt.figure(figsize=(10, 6))
sns.histplot(cluster_2['方向变化'], kde=True)
plt.title('Cluster 2 方向变化分布')
plt.xlabel('方向变化次数')
plt.ylabel('频率')
plt.show()

# 特征之间的相关性分析
correlation_matrix = cluster_2[['时间差', '移动距离', '方向变化']].corr()
print("Cluster 2 特征之间的相关性: ")
print(correlation_matrix)
```

```
# 可视化相关性矩阵
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Cluster 2 时间差、移动距离和方向变化的相关性')
plt.show()
```

## 附录八、Python(3.12.1)问题三 5月1号至5月4号每15分钟巡游车辆数量

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager

plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题

# 读取 Excel 文件
file_path = '4、第三题\巡游车辆.xlsx' # 请替换为您的文件路径
data = pd.read_excel(file_path)

# 将时间列转换为日期时间格式
data['时间'] = pd.to_datetime(data['时间'])

# 过滤 5月1号到5月5号的数据
filtered_data = data[(data['时间'] >= '2024-05-01') & (data['时间'] < '2024-05-06')]

# 设置时间列为索引
filtered_data.set_index('时间', inplace=True)

# 按 15 分钟时间段汇总车辆数量
summary_df = filtered_data.resample('15T').size()

# 绘制图表
plt.figure(figsize=(12, 6))
plt.plot(summary_df.index, summary_df, marker='o', linestyle='-', color='b')
plt.title('5月1号至5月5号每15分钟巡游车辆数量')
plt.xlabel('时间')
plt.ylabel('车辆数量')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

plt.savefig('5月1号至5月5号每15分钟巡游车辆数量.png')
```

```
# 显示图表
```

```
plt.show()
```

## 附录八、Python(3.12.1)问题三 5月1号至5月4号巡游车辆数量与最大面积(面积不会超过线段)

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager

# 设置字体，解决中文显示问题
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

# 读取 Excel 文件
file_path = '4、第三题\巡游车辆.xlsx'
data = pd.read_excel(file_path)

# 将时间列转换为日期时间格式
data['时间'] = pd.to_datetime(data['时间'])

# 过滤 5月1号到5月5号的数据
filtered_data = data[(data['时间'] >= '2024-05-01') & (data['时间'] < '2024-05-06')]

# 设置时间列为索引
filtered_data.set_index('时间', inplace=True)

# 按 15 分钟时间段汇总车辆数量
summary_df = filtered_data.resample('15T').size().to_frame(name='Vehicle Count')

# 定义时间窗口宽度 (2.87 小时 = 172 分钟)
T_hours = 2.87 # 时间窗口宽度
T_minutes = T_hours * 60 # 转换为分钟

# 计算每天的最大面积
def calculate_max_area_per_day(df, window_size_minutes):
    max_areas = []
    rightmost_times = []
    for day, group in df.groupby(df.index.date):
        rolling_sum = group['Vehicle
Count'].rolling(f{int(window_size_minutes)}T).sum()
        max_area = rolling_sum.max()
        max_areas.append(max_area)
        rightmost_times.append(day + pd.Timedelta(minutes=window_size_minutes))

    return max_areas, rightmost_times
```

```

max_time = rolling_sum.idxmax()
rightmost_time = max_time + pd.Timedelta(minutes=window_size_minutes)

max_areas.append(max_area)
rightmost_times.append(rightmost_time)

return pd.DataFrame({
    'Date': list(df.groupby(df.index.date).groups.keys()),
    'Max Area': max_areas,
    'Rightmost Time': rightmost_times
})

```

# 计算每一天的最大面积和最右边的时间点

```

max_areas_with_time = calculate_max_area_per_day(summary_df, T_minutes)

```

# 绘制结果：巡游车辆数量与最大面积

```

fig, ax1 = plt.subplots(figsize=(12, 8))

# 绘制 5 月 1 号至 5 月 5 号 每 15 分钟巡游车辆数量
ax2 = ax1.twinx()
ax1.plot(summary_df.index, summary_df['Vehicle Count'], marker='o', linestyle='-', color='b', label='巡游车辆数量')
ax1.set_xlabel('时间')
ax1.set_ylabel('车辆数量', color='b')
ax1.tick_params(axis='y', labelcolor='b')
ax1.grid(True)

# 绘制最大面积，确保面积不会超过线段
for date, max_area, rightmost_time in zip(max_areas_with_time['Date'],
                                            max_areas_with_time['Max Area'],
                                            max_areas_with_time['Rightmost Time']):
    leftmost_time = rightmost_time - pd.Timedelta(minutes=T_minutes)
    ax2.bar(leftmost_time + (rightmost_time - leftmost_time) / 2, max_area,
            width=T_hours / 24, color='green',
            alpha=0.6, align='center')

# 添加次坐标轴标签
ax2.set_ylabel('最大面积', color='green')
ax2.tick_params(axis='y', labelcolor='green')

# 添加标题
plt.title('5 月 1 号至 5 月 5 号巡游车辆数量与最大面积 (面积不会超过线段)')

```

```
# 调整布局并展示图表
plt.tight_layout()
plt.savefig('5月1号至5月5号巡游车辆数量与最大面积(面积不会超过线段).png')
plt.show()
```

## 附录九、Python(3.12.1)问题三平均停车时长

```
import pandas as pd
import openpyxl

# 读取文件
df_51_huanjingweek = pd.read_excel('4、第三题\巡游车辆.xlsx')

# 删除车牌号为空或为'unnnnowk'的行
df_51_huanjingweek = df_51_huanjingweek[~df_51_huanjingweek['车牌号'].isin(['无车牌', 'unnnnowk'])]
df_51_huanjingweek = df_51_huanjingweek[df_51_huanjingweek['交叉口'] == '纬中路-景区出入口']

# 按交叉口列进行分组
grouped = df_51_huanjingweek.groupby('车牌号')
print(f"分组的数量: {len(grouped)}")
# 存储最终结果
final_result = []

# 遍历每一组数据（每个交叉口）
for name, group in grouped:

    specific_rows = group.nlargest(2, '时间') # 这里以时间列排序，取前2行为例
    specific_rows['时间差'] = specific_rows['时间'].diff().dt.total_seconds() # 时间差以秒计算

    result = specific_rows[['车牌号', '时间差']].dropna()
    final_result.append(result)

# 合并所有组的结果
final_df = pd.concat(final_result)

# 将结果保存为新的Excel文件
final_df.to_excel('时间差结果.xlsx', index=False)
print("结果已保存到 '时间差结果.xlsx'")
```

## 附录十、Python(3.12.1)问题三求解最大平均停车时长

```
import pandas as pd

# 加载数据
excel_path = '4、第三题/巡游车辆.xlsx'
data = pd.read_excel(excel_path)

# 过滤出纬中路-景区出入
wzh_jq_entry_data = data[data['交叉口'] == '纬中路-景区出入口'].copy()

# 将时间列转格式提取日期信息
wzh_jq_entry_data['时间'] = pd.to_datetime(wzh_jq_entry_data['时间'])
wzh_jq_entry_data['日期'] = wzh_jq_entry_data['时间'].dt.date

# 根据车牌号与时间排序数据
wzh_jq_entry_data.sort_values(['车牌号', '时间'], ascending=[True, True],
inplace=True)

# 计算每辆车连续记录之间的时间差
wzh_jq_entry_data['时间差'] = wzh_jq_entry_data.groupby('车牌号')['时间'].diff().dt.total_seconds() / 3600 # 秒转小时

# 筛选同一天的时间差记录
wzh_jq_entry_data['上次日期'] = wzh_jq_entry_data.groupby('车牌号')['日期'].shift() # 获取上一次的日期
same_day_records = wzh_jq_entry_data[wzh_jq_entry_data['日期'] == wzh_jq_entry_data['上次日期']] # 保留同日记录

# 对每个车牌号选出停车时长
longest_parking_time_per_vehicle = same_day_records.groupby('车牌号')['时间差'].max()

# 计算所有车辆的平均最长时长
average_parking_duration = longest_parking_time_per_vehicle.mean()
print(f'所有车辆的平均最长停车时长为: {average_parking_duration:.2f} 小时')
```

## 附录十一、Python(3.12.1)问题四求解量化数据的平均速度

```
import pandas as pd
import numpy as np

## 读取数据
# all_data = pd.read_csv('E 题\附件 2.csv',encoding='gbk')
```

```

# all_data['时间'] = pd.to_datetime(all_data['时间'])
# quant_data_4_19 = all_data[all_data['时间'].dt.date == pd.to_datetime('2024-04-19').date()]
# quant_data_4_21 = all_data[all_data['时间'].dt.date == pd.to_datetime('2024-04-21').date()]
# quant_data_5_3 = all_data[all_data['时间'].dt.date == pd.to_datetime('2024-05-03').date()]

## 删除无用数据
# quant_data_4_19 = quant_data_4_19[~quant_data_4_19['车牌号'].isin(['无车牌', 'unnnnowk'])]
# quant_data_4_21 = quant_data_4_21[~quant_data_4_21['车牌号'].isin(['无车牌', 'unnnnowk'])]
# quant_data_5_3 = quant_data_5_3[~quant_data_5_3['车牌号'].isin(['无车牌', 'unnnnowk'])]

## 存放在新的 excel, 方便后面测试代码
# with pd.ExcelWriter('4、第四题\评价_量化数据.xlsx') as writer:
#     quant_data_4_19.to_excel(writer, sheet_name='4.19_工作日')
#     quant_data_4_21.to_excel(writer, sheet_name='4.21_周末')
#     quant_data_5_3.to_excel(writer, sheet_name='5.3_五一')

# 读取数据
quant_data_4_19 = pd.read_excel('4、第四题\评价_量化数据.xlsx',
sheet_name='4.19_工作日')
quant_data_4_21 = pd.read_excel('4、第四题\评价_量化数据.xlsx',
sheet_name='4.21_周末')
quant_data_5_3 = pd.read_excel('4、第四题\评价_量化数据.xlsx',
sheet_name='5.3_五一')

road_distance = {
    '环西路-纬中路-经一路-纬中路':0.46,'经一路-纬中路-环西路-纬中路':0.46,
    '经一路-纬中路-经二路-纬中路':0.34,'经二路-纬中路-经一路-纬中路':0.34,
    '经二路-纬中路-经三路-纬中路':0.44,'经三路-纬中路-经二路-纬中路':0.44,
    '经三路-纬中路-经中路-纬中路':0.42,'经中路-纬中路-经三路-纬中路':0.42,
    '经中路-纬中路-纬中路-景区出入口':0.53,'纬中路-景区出入口-经中路-纬中路':0.53,
    '纬中路-景区出入口-经四路-纬中路':0.56,'经四路-纬中路-纬中路-景区出入口':0.56,
    '经四路-纬中路-经五路-纬中路':0.43,'经五路-纬中路-经四路-纬中路':0.43,
    '经五路-纬中路-环东路-纬中路':0.27,'环东路-纬中路-经五路-纬中路':0.27,
    '经中路-环北路-经中路-纬一路':0.52,'经中路-纬一路-经中路-环北路':0.52,
    '经中路-纬一路-经中路-纬中路':0.51,'经中路-纬中路-经中路-纬一路':0.51,
    '经中路-纬中路-经中路-环南路':0.71,'经中路-环南路-经中路-纬中路':0.71
}

```

```
}
```

```
def get_all_lu_avg_speed(data):
    lose = 0
    grouped_data = data.groupby('车牌号')
    num_groups = grouped_data.ngroups
    print(f"总共有 {num_groups} 个分组")
    all_speeds = []
    for car_id, group in data.groupby('车牌号'):
        # 以时间顺序进行升序排列
        group = group.sort_values(by='时间')

        # 遍历该分组内的所有行，计算时间差和速度
        for i in range(1, len(group)):
            # Step 2: 计算时间差，单位为小时
            time_diff = (group.iloc[i]['时间'] - group.iloc[i-1]['时间']).total_seconds() /
3600

            # Step 3: 拼接交叉路口内容
            road_segment = group.iloc[i-1]['交叉口'] + "-" + group.iloc[i]['交叉口']

            # Step 4: 在路段-距离字典中找到对应的距离
            if road_segment in road_distance:
                distance = road_distance[road_segment]

            # Step 5: 计算速度
            speed = round(distance / time_diff, 0)

            # Step 6: 将速度添加到列表中
            all_speeds.append(speed)
        else:
            lose += 1
    return all_speeds, lose

speeds_data_4_19 = get_all_lu_avg_speed(quant_data_4_19)
speeds_data_4_21 = get_all_lu_avg_speed(quant_data_4_21)
speeds_data_5_3 = get_all_lu_avg_speed(quant_data_5_3)

print('4.19 工作日平均速度: ', np.mean(speeds_data_4_19[0]))
print('4.21 周末平均速度: ', np.mean(speeds_data_4_21[0]))
print('5.3 五一平均速度: ', np.mean(speeds_data_5_3[0]))

print('4.19 工作日丢包数: ', speeds_data_4_19[1])
```

```
print('4.21 周末丢包数: ', speeds_data_4_21[1])
print('5.3 五一丢包数: ', speeds_data_5_3[1])
```

## 附录十二、Python(3.12.1)问题四求解量化数据的巡游车辆

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

# 通用数据加载和预处理函数
def load_and_preprocess_data(file_path, Sheet1, distance=1, speed_range=(10, 120)):
    data = pd.read_excel(file_path, sheet_name=Sheet1)
    data.columns = ['ID', '方向', '时间', '车牌号', '交叉口']
    data['timestamp'] = pd.to_datetime(data['时间'])
    data = data.sort_values(by=['车牌号', '时间'])
    data['时间差'] = data.groupby('车牌号')['时间'].diff().dt.total_seconds()
    data['速度'] = distance / (data['时间差'] / 3600)
    data_filtered = data[(data['速度'] < speed_range[1]) & (data['速度'] > speed_range[0])]
    return data_filtered

# 聚类并可视化函数
def perform_clustering(data_filtered, n_clusters=2):
    vehicle_features = data_filtered.groupby('plate_number').agg({
        '速度': 'mean',
        '交叉口': 'nunique'
    }).reset_index()

    kmeans = KMeans(n_clusters=n_clusters)
    vehicle_features['cluster'] = kmeans.fit_predict(vehicle_features[['速度', '交叉口']])

    plt.scatter(vehicle_features['速度'], vehicle_features['交叉口'],
    c=vehicle_features['cluster'])
    plt.xlabel('平均速度 (km/h)')
    plt.ylabel('出现的不同路段数')
    plt.title('巡游车辆的 K-means 聚类结果')
    plt.show()

    return vehicle_features['cluster'].value_counts()
```

```

# 巡游车辆判定函数
def identify_cruising_vehicles(data, threshold=3, batch_size=50000):
    location_counts = data.groupby(['车牌号', '交叉口']).size().reset_index(name='count')
    cruising_vehicles = location_counts[location_counts['count'] >= threshold]
    return len(cruising_vehicles['车牌号'].unique())

# 主流程：处理 4 月 19 日和 4 月 21 日的数据
file_paths = {
    '4 月 19 日': '4、第四题\评价_量化数据.xlsx',
    '4 月 21 日': '4、第四题\评价_量化数据.xlsx'
}

for date, file_path in file_paths.items():
    data_filtered = load_and_preprocess_data(file_path)
    cluster_counts = perform_clustering(data_filtered)
    print(f'{date} 车辆聚类结果:\n', cluster_counts)

# 处理 5 月 3 日数据
filtered_data = pd.read_excel('4、第四题\评价_量化数据.xlsx', sheet_name='5.3_五一')
may_3_data = filtered_data[filtered_data['方向\t时间\t车牌号\t交叉口'].str.contains('2024-05-03')]

may_3_data = may_3_data['方向\t时间\t车牌号\t交叉口'].str.split('\t', expand=True)
may_3_data.columns = ['方向', '时间', '车牌号', '交叉口']

total_cruising_vehicle_count_may_3 = identify_cruising_vehicles(may_3_data)
print("5 月 3 日巡游车辆总数:", total_cruising_vehicle_count_may_3)

```

### 附录十三、Python(3.12.1)问题四求解量化数据的车流量

```

import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import font_manager

# 设置字体
# 设置字体，解决中文显示问题
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

# 读取数据
quant_data_4_19 = pd.read_excel('4、第四题\评价_量化数据.xlsx',

```

```

sheet_name='4.19_工作日')
quant_data_4_21 = pd.read_excel('4、第四题\评价_量化数据.xlsx',
sheet_name='4.21_周末')
quant_data_5_3 = pd.read_excel('4、第四题\评价_量化数据.xlsx',
sheet_name='5.3_五一')

def jisuan_meixiaoshi_pingjvnceliuliang(data):

    data['小时'] = data['时间'].dt.hour

    hourly_unique_plates = data.groupby('小时')['车牌号'].nunique().reset_index()

    hours = pd.DataFrame({'小时': range(24)})
    hourly_unique_plates = pd.merge(hours, hourly_unique_plates, on='小时',
how='left').fillna(0)
    return hourly_unique_plates

def huatu(savefig_path,data):
    plt.figure(figsize=(12, 6))
    plt.bar(data['小时'], data['车牌号'], color='skyblue', edgecolor='black')

    plt.xlabel('小时')
    plt.ylabel('车牌号数量')
    plt.title('每小时内车流量')
    plt.xticks(range(0, 24))
    plt.savefig(savefig_path, dpi=300)
    plt.show()

pj_clj_4_19 = jisuan_meixiaoshi_pingjvnceliuliang(quant_data_4_19)
pj_clj_4_21 = jisuan_meixiaoshi_pingjvnceliuliang(quant_data_4_21)
pj_clj_5_3 = jisuan_meixiaoshi_pingjvnceliuliang(quant_data_5_3)

# 写入 excel
with pd.ExcelWriter('平均车流量.xlsx') as writer:
    pj_clj_4_19.to_excel(writer, sheet_name='4月19日平均车流量', index=False)
    pj_clj_4_21.to_excel(writer, sheet_name='4月21日平均车流量', index=False)
    pj_clj_5_3.to_excel(writer, sheet_name='5月3日平均车流量', index=False)

huatu('4月19日平均车流量图.png',pj_clj_4_19)
huatu('4月21日平均车流量图.png',pj_clj_4_21)
huatu('5月3日平均车流量图.png',pj_clj_5_3)

```

## 附录十四、Python(3.12.1) 熵权法+层次分析法求解评分

```
import numpy as np
import pandas as pd
import openpyxl

def mylog(p):
    n = len(p)
    lnp = np.zeros(n)
    for i in range(n):
        if p[i] == 0:
            lnp[i] = 0
        else:
            lnp[i] = np.log(p[i])
    return lnp

data = pd.read_excel('4、第四题\评价_量化数据.xlsx', sheet_name='熵权法', engine='openpyxl')
matrix = data.to_numpy()
print(matrix)
X = matrix[:, 1:4]
Z = X / np.sqrt(np.sum(X*X, axis=0))
print("标准化矩阵 Z = ")
print(Z)
n, m = Z.shape
D = np.zeros(m)
for i in range(m):
    x = Z[:, i]
    p = x / np.sum(x)
    e = -np.sum(p * mylog(p)) / np.log(n)
    D[i] = 1 - e
W = D / np.sum(D)
print("权重 W = ")
print(W)

# 由此做判断判断矩阵
A = np.array([[1,1/2,3/2],
              [2,1,3],
              [2/3,1/3,1]])

n = A.shape[0] # 获取 A 的行
print(n)

eig_val, eig_vec = np.linalg.eig(A)
Max_eig = max(eig_val)
```

```

CI = (Max_eig - n) / (n-1)
RI = [0, 0.0001, 0.52, 0.89, 1.12, 1.26, 1.36, 1.41, 1.46, 1.49, 1.52, 1.54, 1.56, 1.58,
1.59]

CR = CI / RI[n]

print('一致性指标 CI=', CI)
print('一致性比例 CR=', CR)

if CR < 0.10:
    print('因为 CR<0.10, 所以该判断矩阵 A 的一致性可以接受!')

    # 计算每列的和
    ASum = np.sum(A, axis=0)
    # 获取 A 的行和列
    n, _ = A.shape
    # 归一化
    Stand_A = A / ASum
    # 各列相加到同一行
    ASumr = np.sum(Stand_A, axis=1)
    # 计算权重向量
    weights = ASumr / n

    weights=[round(weight, 3) for weight in weights]
    print(weights)
else:
    print('注意: CR >= 0.10, 因此该判断矩阵 A 需要进行修改!')

# 进行计算并将结果写入第五列

```