| | |
|---|---|
| 队伍编号 | 366 |
| 赛道 | A |

# 基于流量预测的动态载频开关研究

## 摘 要

随着移动通信系统的发展，合理地设计基站节能开关策略越来越重要。考虑到基站具有潮汐效应，若一直以高容量时段的载频数量来运行基站配置将会造成极大的能源消耗，因此需要根据小区的实际需求，动态配置基站载频，以节约能耗，提高网络能效。

本文数据处理阶段分为数据清洗和数据转换两部分，其中数据清洗包括去除重复值、去除离群值、填补缺失值，数据转换用于统一数据格式，旨在改善数据质量，便于后续分析预测；数据分析阶段包括流量序列可视化处理、平稳性检验和三种不同序列的相关性检验，分析结果为流量预测和小区聚类的可行性提供一定的理论依据。

针对问题一，不同的小区流量序列之间具有较强的相关性，在时域上体现为流量变化特性的相似性。本文综合考虑流量变化的相对趋势和绝对幅度，对各个小区的流量序列分别提取 RANK 向量特征和差分特征，并将特征归一化后以一定的权重进行融合。聚类算法中利用 Gap 统计量获得最优类簇数，通过 TBD 距离度量各个小区流量特征之间的相似度，使用 SSE 评价指标确定最佳的聚类结果，并分析每一类小区的流量特点。

针对问题二，考虑基站运行的潮汐现象，旨在设计一种智能基站动态载频配置方案以灵活调整网络容量，提升网络能效。本文首先将流量负载划分为 4 个等级，并根据等级划分结果预设置相应的基站载频开关阈值。为了避免频繁地载频关断和网络初始化，本文限制操作执行的间隔时间。在设定好的载频开关操作时间点，基于下一时段的流量预测以及实时负载情况自动控制基站载频。通过仿真实验，从系统功耗和系统能效两方面验证动态载频配置方案的有效性。

关键词：相关性检验，RANK 向量，特征融合，聚类，潮汐效应，动态载频配置

# 目 录

# 一、问题重述

## 1.1 问题背景

近年来，移动通信技术不断发展，小区个数不断增加，数据业务需求日益增长，电信运营商已采取增加载频等扩容措施来提升网络容量，改善用户体验。无线通信网络的设计规划通常需要依据网络最大负载情况确定，考虑到基站运行过程中普遍存在的潮汐效应，在用户业务量需求较低的时段，若仍按高容量时段的载频数量来运行基站配置，将会带来不必要的能源消耗。若根据各个小区内不同时间点的流量变化设置自动开关限制部分载频，能够有效降低系统能耗，提升网络能效。

基于流量预测的动态开关载频需要对各个小区的流量数据进行实时有效地感知和处理，但小区数目众多，直接分析效率低下，考虑到具有相似流量变化趋势的小区可以采用统一的基站载频配置方案，因此基于提取到的流量序列特征将多个小区聚类是较为关键的一步，不仅有助于后续高效地分析，而且克服了流量精准预测中单个小区数据量不足的缺点。分析基站未来的流量对设置合适的开关载频阈值十分重要，阈值过低会造成用户的体验感差，过高则会造成资源的浪费，因此根据实际情况设置合适的动态载频开关比例具有重要的研究价值。

## 1.2 问题重述

已知小区在2018年3月1日至4月19日的小时级上行流量和下行流量数据，要求依据小区在50天内上行和下行流量的变化趋势，得到小区聚类结果和基站动态载频配置方案。具体地，本文考虑并解决以下两个问题：

问题一：在小区近50天内上行和下行流量数据已知的情况下，提取流量时间序列数据的特征，依据特征对小区进行聚类，并分析每一类小区的流量特点。

问题二：考虑到基站运行具有潮汐现象，需要根据流量变化设计自动开关限制部分载频的方法，有效优化基站的运行。设置合适的基站开关载频的流量阈值，在不降低用户体验质量的情况下，降低系统能耗。

# 二、模型假设和符号说明

## 2.1 模型假设

假设一：假设在小区流量预测过程中，不发生因外界偶然因素导致的流量突变；

假设二：假设小区的上行流量和下行流量之间互不影响；

假设三：假设在小区短期流量预测的过程中，不发生基站扩容；

假设四：假设在小区流量预测过程中，各基站独立工作，不进行协同工作；

假设五：假设小区所处的蜂窝网络是全双工双层异构蜂窝网络，由宏基站和 K 个小型基站组成；

假设六：假设小区为每个小型基站的覆盖范围，即每个小区由一个小型基站和多个终端用户组成。

## 2.2 符号说明

| 符号 | 符号说明 | 符号 | 符号说明 |
|---|---|---|---|
| $\rho_l$ | 自相关系数 | $SINR_{u,c_i}^{UL}$ | 上行信道的信干噪比 |
| CC | 互相关系数 | $P_u$ | 用户终端 u 的传输功率 |
| $x_p$ | 第 $p$ 号小区流量时间序列 | $P_{c_i}^{\max}$ | 载频 $c_i$ 的最大传输功率 |
| $\hat{x}_p(h)$ | 第 $p$ 号小区差分特征序列 | $h_{f,u}$ | 用户终端 u 受到来自基站 f 的干扰增益 |
| $r_p(h)$ | 第 $p$ 号小区 RANK 向量 | $R_{u,c_i}^{-\alpha}$ | 用户终端 u 通过分载频 $c_i$ 到基站 f 之间的路径损耗 |
| $y_p(h)$ | 加权融合后的新序列 | $N_0$ | 高斯白噪声信道的功率谱密度 |
| $\alpha$ | $\hat{x}_p(h)$ 的权值 | $W$ | 载频带宽 |
| $\beta$ | $r_p(h)$ 的权值 | $R_{u,c_i}^{UL}$ | 载频 $c_i$ 服务的用户终端 u 的上行信道容量 |
| $t(\vec{y}_p,\vec{y}_q)$ | $\vec{y}_p$ 和 $\vec{y}_q$ 序列中对应点的距离 | $P_{CU}$ | 用户固定消耗功率 |

| | | | |
|---|---|---|---|
| $TBD(\vec{y}_p,\vec{y}_q)$ | $\vec{y}_p$ 和 $\vec{y}_q$ 序列间的距离 | $\eta_u$ | 用户 u 的能效 |
| $\vec{\mu}_k^*$ | 聚类中心 | $SINR_{f,c_i}^{DL}$ | 基站所分配的载频 $c_i$ 到用户的下行信道的信干噪比 |
| $C_p$ | 第 $p$ 类样本的集合 | $\zeta$ | 功率消耗系数 |
| $W_k$ | 簇内离散度 | $\tau_i$ | 表示载频 $c_i$ 是否开启 |
| $s_k$ | 修正因子 | k | 基站的载频配置数目 |
| $sd_k$ | 参考零分布簇内离散度 $\log(W_k^*)$ 的标准差 | $R_{c_i,u}^{DL}$ | 载频 $c_i$ 服务的用户终端的 u 的下行信道容量 |
| $L$ | 归一化网络流量 | $P_{BS}$ | 小区载频总功耗 |
| $\beta_i$ | 第 i 个归一化流量负载门限 | $P_{const}$ | 载频 $c_i$ 工作状态固定功耗 |
| $z_i$ | 第 $i$ 个阈值 | $\rho_{c_i}$ | 当前载频 $c_i$ 的负载因子 |
| $L_p$ | 下一时段的流量预测水平 | $n_i$ | 正被载频 $c_i$ 服务的用户数 |
| $z_p$ | 下一时段载频关闭的比例参数 | $U_{c_i}^{\max}$ | 载频 $c_i$ 的最大用户容量 |
| $z_n$ | 当前时段载频关闭的比例参数 | $\varphi_{c_i}$ | 载频 $c_i$ 直流到射频的转换因子 |
| $C$ | 基站 f 的载频集合 | $A_{c_i}^{\max}$ | 载频 $c_i$ 可负载最大网络流量 |
| $U_f$ | 所服务的用户集合 | $\omega$ | 效率权重因子 |

# 三、数据预处理和初步分析

## 3.1 原始数据集介绍

题中所给的原始数据集来源于十万多个小区真实的网络流量采集数据，该流量数据集以小时为时间颗粒度，采集了十万多个小区的流量记录数据。在原始流量数据集中，采集的流量数据时间跨度为 2018 年 3 月 1 日至 2018 年 4 月 19 日，

涉及到 50 天，每天的流量数据按 00:00 到 23:00 共 24 个小时记录在同一个文件里，文件数据有三条：日期、时间、小区编号、上行业务量、下行业务量，一共有一亿三千多万条的流量记录。原始数据集中具体字段名和含义说明如下表 3.1 所示。

表 3.1 原始数据及字段名及含义说明

| 字段名 | 字段含义 |
| --- | --- |
| 日期 | 表明该数据是哪一天的流量 |
| 时间 | 表明该数据是那个小时的流量 |
| 小区编号 | 表明该数据是哪个小区的流量 |
| 上行业务量 | 表明该数据是用户到基站的流量 |
| 下行业务量 | 表明该数据是基站到用户的流量 |

根据表 3.1，可发现数据集中没有小区的经纬度或任何地理信息，无法对基站的确切位置进行确认，只能通过小区流量的变化趋势和其他相关特征对小区进行大概分类，无法有效地对小区或小区所属基站之间的空间联系进行挖掘，因此本文从时间序列方法的研究上着手对小区进行分类。

## 3.2 数据预处理

在原始数据集中，由于小区数量过多，分布广，加之各种不可控因素，导致所采集的数据会存在重复值、缺失值和异常值，无法直接进行数据挖掘。为了改善数据质量、提高后续模型预测结果的可靠性，本文首先对原始小区流量数据进行预处理操作。鉴于文献[1][2]中的处理思路，本文将数据预处理的步骤分为数据清洗（包括去除重复值、去除离群值、填补缺失值）和数据转换两部分。

### 3.2.1 数据清洗

步骤一：去除重复值

去除重复值是为了防止在给定数据中，某小区某个时间点的流量数据出现多次，或者数据不一致的情况，这种不一致在后续处理数据时会发生错误，所以本

文首先去除重复值。将小区编号、日期、时间作为对比指标，观测是否有两条或两条以上的数据指标相同的情况，表 3.1 给出了重复数据的一部分。

表 3.1 去重前的重复数据

| | DATE | HOUR | NAME | LABEL1 | LABEL2 |
|---|---|---|---|---|---|
| **19048** | 2018/3/1 | 0:00:00 | 19042 | 0.004881 | 0.006845 |
| **19049** | 2018/3/1 | 0:00:00 | 19042 | 0.004881 | 0.006845 |
| **134669** | 2018/3/1 | 1:00:00 | 19042 | 0.000333 | 0.000429 |
| **134670** | 2018/3/1 | 1:00:00 | 19042 | 0.000333 | 0.000429 |
| **248133** | 2018/3/1 | 2:00:00 | 19042 | 0.001927 | 0.001366 |
| **248134** | 2018/3/1 | 2:00:00 | 19042 | 0.001927 | 0.001366 |
| **361534** | 2018/3/1 | 3:00:00 | 19042 | 0.000657 | 0.000869 |
| **361535** | 2018/3/1 | 3:00:00 | 19042 | 0.000657 | 0.000869 |

针对重复数据，本文只保留其中数据索引值最小的数据，比如表 1 中索引 19048 与索引 19049 对应同一小区，因此需要删除后一条数据。对其它重复数据都进行该操作，结果如表 3.2：

表 3.2 去重后的重复数据

| | DATE | HOUR | NAME | LABEL1 | LABEL2 |
|---|---|---|---|---|---|
| **19048** | 2018/3/1 | 0 | 19042 | 0.004881 | 0.006845 |
| **134669** | 2018/3/1 | 1 | 19042 | 0.000333 | 0.000429 |
| **248133** | 2018/3/1 | 2 | 19042 | 0.001927 | 0.001366 |
| **361534** | 2018/3/1 | 3 | 19042 | 0.000657 | 0.000869 |

步骤二：去除离群值

为了确保后续预测结果的准确性，训练数据必须符合实际情况，本文假设在短期预测中，不发生因外界偶然因素导致的流量突变。故针对每个小区流量，若某个小时的流量数值远大于其相邻时刻流量，则可以认定该流量数据是异常数据，本文首先针对异常数据进行去除。统计发现，各小区的流量数据基本都在正常的范围内，并没有出现某个数据远大于其他数据的情况，所以本文认为所给数据中各小区流量不存在异常数据。

步骤三：填补缺失值

对小区流量序列时间点统计可知，流量数据非连续数据集，而且某些小区缺失的流量数据不是连续时间点数据，而是穿插在整个流量记录的时间段内，如果

将含有缺失值的基站数据全部去除的话，将造成严重的数据浪费，研究结果会仅局限于少量的小区和时间点。为了最大限度地利用原始数据集进行研究分析，需要采取缺失值填补的方法，将含有较少缺失值的小区进行有效地拟合填充，从而为下一步研究提供足够大的数据量和增强后续算法模型的泛化性能和准确率[2]。

数据集中的数据应该是每个小区 2018 年 3 月 1 日至 4 月 19 日的小时级流量数据，理论上每小区应有 1200 条数据。但根据统计结果可知，大部分小区有 1176 条数据，其余小区数据均出现不同程度的数据缺失。本文针对少于 1176 条数据的小区进行缺失流量值填补。考虑到本数据集缺失值比重低，且构成的时间序列的周期长，本文采用下述填充方法：对单个数据点的缺失采用前后两个序列的平均值作为缺失数据填补；对多个数据点的缺失采用多重随机插补法。

### 3.2.2 数据转换

由于原始数据集中存在错误的时间信息，比如 2018 年误写为 018 年，而且所给日期都是 2018/x/xx，不利于处理，所以需要进行整合和纠错，同时把提取出的时间信息都变为 2018-x-xx 形式，便于后续算法识别和做时间序列分析。

## 3.3 数据分析

### 3.3.1 可视化处理

以 186 号小区为例，本文将小区流量数据进行可视化处理，观察小区五十天的历史流量变化规律。图 3.1 为该小区五十天和第一周的流量时序图。

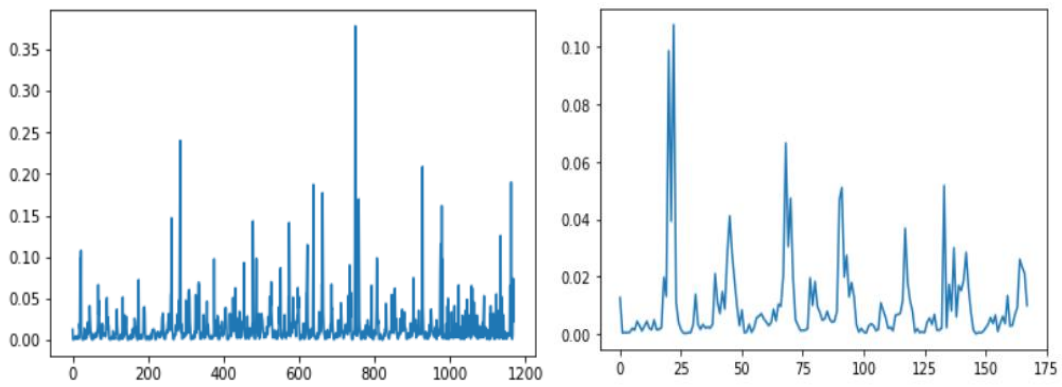



图 3.1 第 186 号小区的流量时序图

从可视化结果可以看出，小区流量出现近似周期性的变化，流量基本都在20:00 至 24:00 之间达到峰值，在 0:00 至 9:00 之间处于低谷，这也比较符合实际情况，移动互联网迅速发展，特别是短视频爆发式发展，用户在晚上休闲娱乐时间多，网络流量大，而在凌晨用户休息时，网络流量最低。

### 3.3.2 平稳性检验

时间序列数据的平稳性对于模型的选择有着至关重要的影响，所以对时间序列进行后续分析之前，首先需要进行平稳性检验。

（1）自相关与偏相关

在实际过程中，研究者经常结合自相关图和偏自相关图来判断时间序列的平稳性，自相关系数 $\rho_l$ 的计算公式如式 3-1 所示，

$$\rho_l = \frac{Cov(X_k, X_{k-l})}{Var(X_K)} = \frac{\gamma_l}{\gamma_0} \tag{3-1}$$

其中，$X_k$ 是时间点 k 对应的随机变量，$X_{k-l}$ 是时间点 k 往前数 $l$ 期对应的随机变量，$\gamma_0$ 是时间序列 $X_k$ 的方差，$\gamma_l$ 是 $X_k$ 与 $X_{k-l}$ 之间的自协方差。

本文对小区流量数据的自相关和偏相关系数进行分析，如图 3.2 所示：

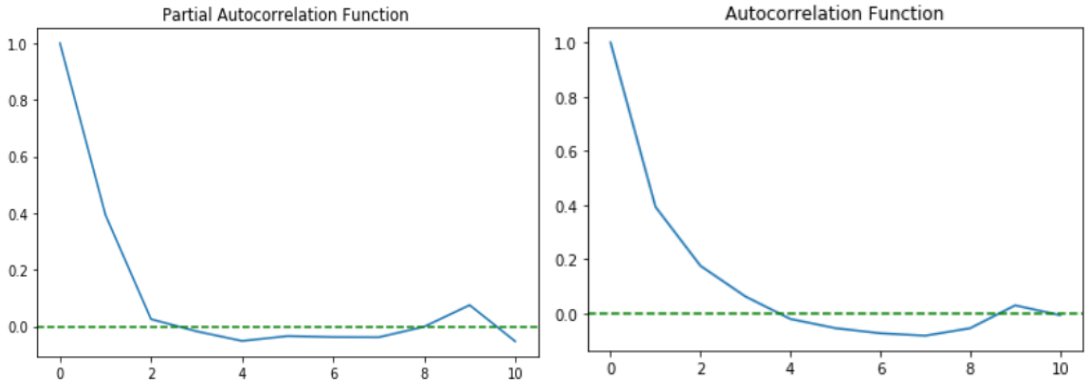



图 3.2 第 186 号小区的自相关图和偏相关图

从图 3.2 可发现，自相关出现截尾现象，偏相关出现拖尾现象，所以可初步认为该序列是平稳的。

（2）单位根检验

自相关图和偏相关图可以直观地反映一定的序列平稳性，但是比较主观，本文另外使用 ADF 检验方法。通过检验，其中 186 小区 ADF 输出结果为：ADF:-4.237613; p-value:0.000569。p 值远小于 0.05，说明拒绝原假设，该序列为平稳序列。本文将所有待预测小区进行平稳性检验，检验结果显示所有小区历史流量数据均为平稳序列。

### 3.3.3 小区上行和下行流量序列相关性检验

所给数据集中，小区流量分为上行流量和下行流量，本节中首先计算上行和下行流量序列的互相关系数(Cross-Correlation, CC)，互相关系数的计算如式 3-2 所示：

$$CC(\bar{x},\bar{y}) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{3-2}$$

互相关系数表示了不同流量序列之间的时空相关程度。如图 3.3 所示是随机 200 个小区上行和下行流量序列互相关系数频率直方图：

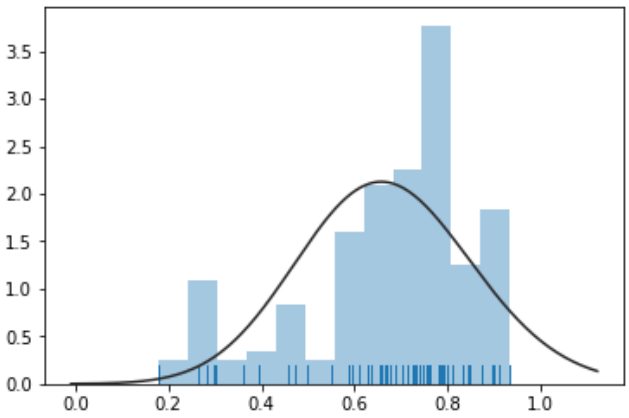



图 3.3 200 个小区上行和下行流量序列互相关系数频率直方图

由图3.3可看出，95%以上的小区上下行流量序列的相关性系数大于0.3，70%以上的相关性系数大于0.5。统计学中定义相关系数在0-0.3之间为微相关，0.3-0.5之间为实相关，0.5-0.8 之间为显著相关，0.8-1.0 之间为高度相关。所以小区的上下行流量具有较强的相关性，可以将上下行流量之和作为总流量序列进行后续分析。

### 3.3.4 不同小区流量序列相关性检验

小区流量本身具有地理位置属性，小区所处的地区属性不同，则该小区的人流量及与基站产生的交互也会不同。比如处于闹市区的小区每天的流量值必定大于位于郊区的小区，处于城市中央商务区的小区与处于住宅区的小区在白天和晚上的流量分布情况也有较大差别。而不同地理位置的小区之间的流量序列往往具有相关性。本文随机选取了数据集中部分小区，计算这些小区流量序列之间的互相关系数，如图 3.4 所示是某 40 个小区流量序列互相关系数频率直方图：

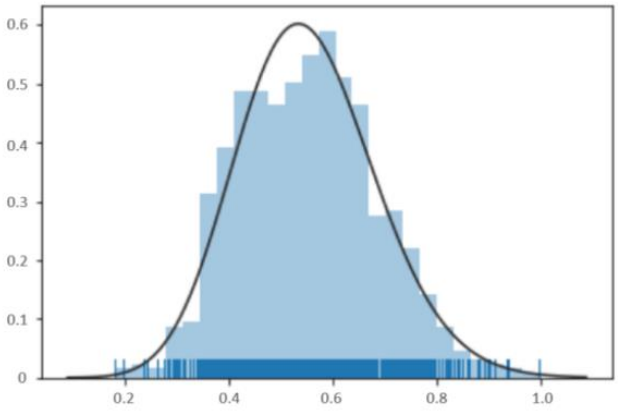



图 3.4 40 个小区流量序列互相关系数频率直方图

随机选取的这 40 个小区中，两两配对计算其互相关系数，由图 3.4 可知，98%以上小区对的流量序列互相关系数在 0.3 以上，即为实相关，60%左右的小区对的流量序列互相关系数在 0.5 以上，为显著相关。可以得出结论，不同小区的流量序列之间具有相关性。这也为问题一中对小区进行聚类提供了理论支撑。

### 3.3.5 小区周流量序列相关性检验

题中所给数据集包含十万多个小区 50 天的小时级流量数据，需处理的数据量过大，不利于特征提取及相关计算。本小节首先验证每个小区每周流量数据的相关性，再把小区每周相同时间点的流量数据取平均作为该时间点的流量数据，将 50 天的数据精简为 7 天的数据，缩短了流量序列长度，降低了处理复杂度。

以 186 号小区为例，由 3.3.1 节可视化处理可知，小区流量呈现近似周期性的变化。本节通过式 3-2 计算互相关系数，进一步说明小区七个周流量序列之间具有较强的相关性。图 3.5 所示为第 186 号小区周流量互相关系数频率直方图：

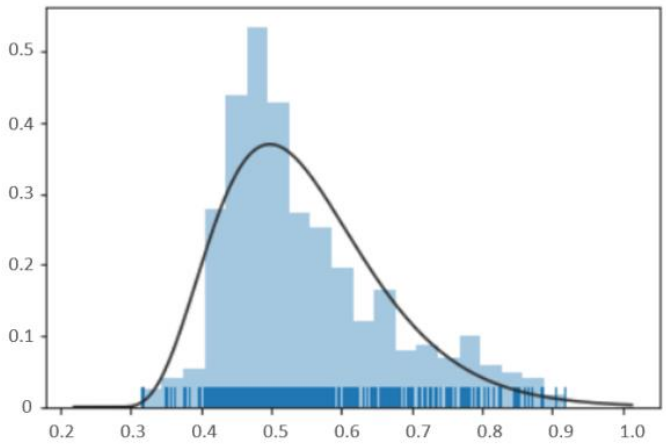



图 3.5 第 186 号小区周流量互相关系数频率直方图

由图 3.5 可知，几乎所有周流量序列互相关系数在 0.3 以上，即为实相关，60%左右的周流量序列互相关系数在 0.5 以上，为显著相关。于是得出结论，同一小区的周流量序列之间具有相关性。

本节对小区流量序列进行统计特性分析可知，小区的流量序列具有一定的自相似性与明显的周期性，同一小区的上下行流量序列之间、不同小区的流量序列之间、同一小区不同周流量序列之间都具有相关性。故而引进聚类的思想，将具有类似流量变化特性的小区聚为一个类簇，研究不同类簇中的小区流量序列特性。

本章对原始小区流量的数据处理和分析流程如图 3.6 所示：

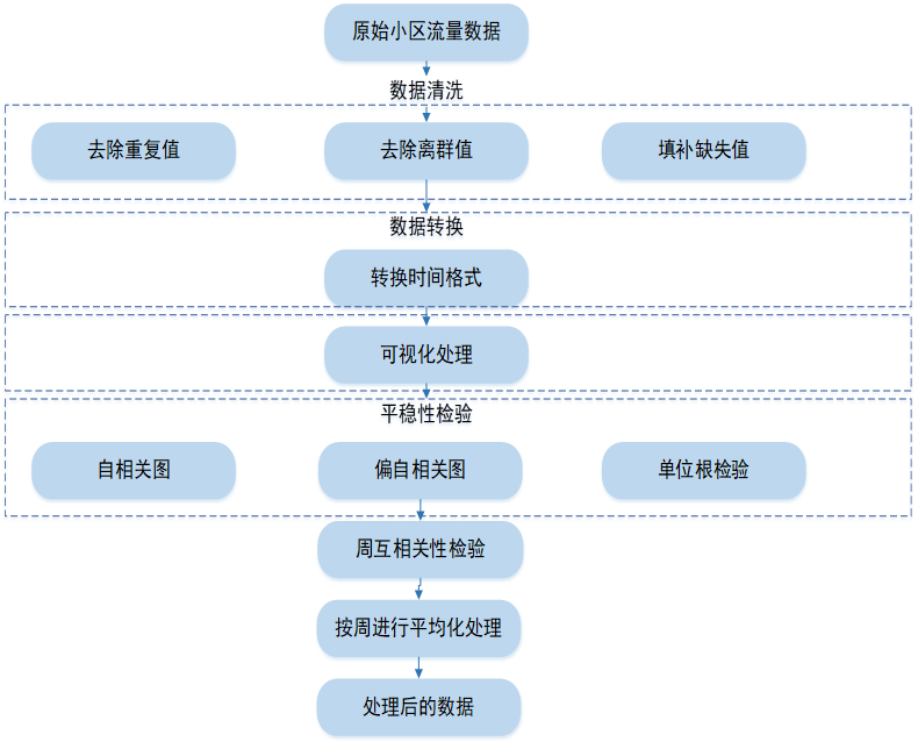



图 3.6 数据处理和分析流程

# 四、问题一：提取特征并对小区分类

## 4.1 问题一的求解思路

问题一是对小区流量数据提取特征进而分类的问题。由 3.3 节可知，小区流量数据是一个具有平稳性和周期性的时间序列，但是易受到外部多种因素的干扰，呈现非线性时间序列的特点。并且不同的小区流量序列之间具有较强的相关性，这种相关性在时域上体现为流量变化特性的相似性，因此可以将机器学习中的聚类算法引入到流量分析，利用聚类算法将小区流量序列按照相关性强弱程度聚成不同的类簇，每个类簇中的小区具有较强的相关性，即具有相似的流量变化特性。在本论文的分析中，已把小区流量数据处理为时间序列的形式，因此小区流量序列的聚类问题可转化为时间序列聚类问题。

时间序列中各点是按照一定顺序排列的，每个点与前后点之间存在增大或减小的变化趋势，小区流量序列的每个值不仅反应当下时间段的流量值大小，也包含有这个时间段基站流量上升或下降的变化趋势。因此可以将这个变化趋势作为特征。考虑到前后差分对数据变化的绝对幅度描述性较好，RANK 向量可以抓住数据变化的相对趋势，因此希望结合这两个特征的优点，更好地把具有相同特点的小区分为同一类。鉴于此，本文首先把小区流量时间序列转化为差分序列和 RANK 向量序列，将二者作为特征归一化并加权求和后进行聚类，基于聚类结果分析每一类小区的流量特点。

## 4.2 提取特征

### 4.2.1 差分特征

假设小区流量时间序列为 $\vec{x}=(x_1,x_2,\cdots,x_m)$，参考导数时间序列分段近似模型，该时间序列的导数估计，即差分特征为：

$$\hat{x}_h=\begin{cases} x_{h+1}-x_h, h=1 \\ (x_{h+1}-x_{h-1})/2, h\in[2,3,\cdots,m-1] \\ x_h-x_{h-1}, h=m \end{cases} \tag{4-1}$$

本文中，m=7*24，即为 168。$\hat{x}_h$ 表示原小区流量序列第 $h$ 个时间段的流量变化趋势。

### 4.2.2 RANK 向量特征

RANK 向量的含义如表 4.1 所示：

表 4.1 RANK 向量示意表

| 小区编号 | 2018-3-18 | 2018-3-24 | 2018-3-30 | 2018-4-5 | 2018-4-10 | RANK 向量 |
|---|---|---|---|---|---|---|
| 186 | 0.01 | 0.03 | 0.015 | 0.02 | 0.04 | [1,4,2,3,5] |

举例来说，表 4.2 是某小区几天内的流量数据，由此可以得出该小区流量的 RANK 向量为[1,4,2,3,5]，其中向量里面的“2”表示 2018-3-30 的流量在所有流量中的排名为第 2 名，以此类推。各维度的数值越高表示某时刻该小区的流量值越高。本文在数据处理部分，把每个小区的流量序列缩短为 7*24 长度的序列，因此对每个小区来说，其 RANK 向量的长度为 7*24，即 168，十万多个小区将会计算出十万多个 RANK 向量，统计并得出数据集中所有小区的 RANK 向量，作为小区流量数据的一个特征。

### 4.2.3 特征融合

在 4.2.1 和 4.2.2 节，本文已经对小区流量时间序列进行了特征提取，第 $p$ 号小区流量时间序列为 $x_p$，差分特征序列为 $\hat{x}_p(h)$，把 RANK 向量写为时间序列形式 $r_p(h)$，本节中先把提取特征后得到的的两个特征序列归一化，再加权融合后得到新的序列。归一化公式如式 4-2：

$$\begin{aligned} \bar{\hat{x}}_p(h) &= \frac{\hat{x}_p(h)}{\sum_{i=1}^{m} \hat{x}_p(h)} \\ \bar{r}_p(h) &= \frac{r_p(h)}{\sum_{i=1}^{m} r_p(h)}, h = 1, 2, \cdots m \end{aligned} \tag{4-2}$$

由于对所有特征序列都进行了归一化处理，为使公式更简洁，下面公式中的 $\hat{x}_p(h)$、$r_p(h)$ 即代表归一化后的序列。

再加权融合后得到新的序列，如式 4-3：

$$y_p(h)=\gamma\hat{x}_p(h)+(1-\gamma)r_p(h) \tag{4-3}$$

同理第 $q$ 号小区也可得到特征融合后的新序列：

$$y_q(h)=\gamma\hat{x}_q(h)+(1-\gamma)r_q(h) \tag{4-4}$$

由于两个特征序列都进行了归一化，而且可以在不同的方面表达小区流量序列的特点和变化趋势，本文中 $\gamma$ 取 0.5。

## 4.3 聚类算法

聚类的基本思想是：根据算法需要解决的实际问题应用场景以及已有数据集的特点，确定样本间距离的度量方式，按照这个距离度量方式将样本聚成不同的类簇，属于同一类的样本之间相似度尽量地高，而不同类的样本之间相似度尽量地低，并确定指标评价聚类有效性。本文的聚类过程如下所示。

（1）输入融合后的特征序列数据 X，最大迭代次数 $r$ 和质心轮换次数 $e$，最大类簇数目 $K_{\max}$；

（2）令 $k$=1；

（3）随机选取 $k$ 个质心；

（4）形成聚类类簇划分 U；

（5）根据公式(4-12 )计算每个聚类类簇的质心 $\mu_k^*$；

（6）根据公式（4-7）计算每个时间序列到已得的 $k$ 个类簇中心的距离，得到新的聚类类簇划分并记录；

（7）判断步骤(6)中所得聚类类簇划分与原有聚类类簇划分是否一致，如果不一致，则更新类簇划分并跳转步骤(8)，如果一致，则跳转步骤(9)；

（8）判断 $r$ 是否为 0，如果为 0，则跳转步骤(8)，如果不为 0，则令 $r$=$r$-1，并跳转步骤(4)；

（9）判断 $e$ 是否为 0，如果为 0，则计算每次聚类结果的 SSE，选择 SSE 最小的聚类结果，跳转步骤(9)；如果不为 0，改变初始的质心选择，$e$=$e$-1，跳转步骤(4)；

（10）计算 Gap 指标，判断是否满足 Gap 条件，若满足，则输出当前 $k$ 值和聚类划分结果，算法结束，否则进行步骤(11)；

（11）令 $k=k+1$，如果 $k \le 30$，跳转步骤(3)。

### 4.3.1 特征 TBD 距离计算

聚类算法的关键问题是如何定义样本间相似度，即如何计算样本间的距离[5]。在本文中，两个不同序列对应点之间的变化趋势的差异由式 4-5 衡量：

$$t(\vec{y}_p,\vec{y}_q)=\frac{\left|y_p(h)-y_q(h)\right|}{\left|y_p(h)\right|+\left|y_q(h)\right|},h\in[1,2,\cdots m] \tag{4-5}$$

式 4-5 可代表两个小区流量时间序列的变化趋势的不相似度，将这个趋势信息与经典距离计算方式结合，参照相关性计算公式以及 K-Shape 算法的距离公式，即可得出基于趋势信息的基站流量序列距离度量 TBD，如式 4-6 所示：

$$TBD(\vec{y}_p,\vec{y}_q)=2-\frac{\vec{y}_p\cdot T\cdot\vec{y}_q{}^T}{\left\|\vec{y}_p\right\|\cdot\left\|\vec{y}_q\right\|} \tag{4-6}$$

其中 $\left\|\vec{y}_p\right\|$ 是 $\vec{y}_p$ 的范数，矩阵 $T=diag(2-t(\vec{y}_p,\vec{y}_q))$。令 $\vec{s}=\vec{y}_p\cdot T$，距离度量转换为式 4-7：

$$TBD(\vec{y}_p,\vec{y}_q)=2-\frac{s\cdot\vec{y}_q{}^T}{\left\|\vec{y}_p\right\|\cdot\left\|\vec{y}_q\right\|} \tag{4-7}$$

式 4-7 成为度量样本间相似度的指标。

### 4.3.2 更新聚类中心

在更新聚类中心的迭代步骤中，本论文采取斯坦纳序列(Steiner's sequence)的方式：找到一个序列，使得其到该簇内所有时间序列的平方距离之和最小，则这个序列可被称为这个簇的中心。新的聚类中心 $\vec{\mu}_k^*$ 满足：

$$\vec{\mu}_k^*=\arg\min_{\vec{\mu}_k}\sum_{\vec{x}_i\in Y_k}(TBD(\vec{x}_i,\vec{\mu}_k))^2 \tag{4-8}$$

其中 $\vec{x}_i$ 是第 $k$ 个类簇 $Y_k$ 中的序列。根据 TBD 表达式可进一步得出：

$$\vec{\mu}_k^*=\arg\max_{\vec{\mu}_k}\sum_{\vec{x}_i\in p_k}(\sum_{l\in[1,m]}s_{il}\cdot\mu_{kl})^2 \tag{4-9}$$

即：

$$\begin{aligned}\vec{\mu}_k^* &= \arg\max_{\vec{\mu}_k} \sum_{\vec{x}_k \in p_k} (\vec{s}_i^T \cdot \vec{\mu}_k)^2 \\ &= \arg\max_{\vec{\mu}_k} \vec{\mu}_k^T \sum_{\vec{x}_k \in p_k} (\vec{s}_i \cdot \vec{s}_i^T) \cdot \vec{\mu}_k\end{aligned} \tag{4-10}$$

上式中，只有 $\vec{\mu}_k$ 没有被归一化，令 $\vec{\mu}_k = \vec{\mu}_k \cdot Q$，其中 $Q = I - \frac{1}{m}O$，$I$ 是单位矩阵，$O$ 是全 1 矩阵，为了让 $\vec{\mu}_k$ 单位化，式 4-10 可做如下转换：

$$\begin{aligned}\vec{\mu}_k^* &= \arg\max_{\vec{\mu}_k} \frac{\vec{\mu}_k^T \cdot Q^T \cdot \sum_{\vec{x}_i \in p_k} (\vec{s}_i \cdot \vec{s}_i^T) \cdot Q \cdot \vec{\mu}_k}{\vec{\mu}_k^T \cdot \vec{\mu}_k} \\ &= \arg\max_{\vec{\mu}_k} \frac{\vec{\mu}_k^T \cdot R \cdot \vec{\mu}_k}{\vec{\mu}_k^T \cdot \vec{\mu}_k}\end{aligned} \tag{4-11}$$

其中：

$$R = Q^T \cdot \sum_{\vec{x}_i \in p_k} (\vec{s}_i \cdot \vec{s}_i^T) \cdot Q \cdot \vec{\mu} \cdot Q \tag{4-12}$$

经过上述转换，式 4-8 优化为瑞利商数最大化求解问题，可以找到最大的 $\vec{\mu}_k^*$，即为实对称矩阵 R 的最大特征值所对应的特征向量。

### 4.3.3 聚类评价指标

由于聚类算法采用的是启发式的方法进行求解，无法保证求解得到的结果是最优的。为了对聚类结果的质量进行度量，定义误差平方和（SSE）作为优化的目标函数：

$$SSE = \sum_{i=1}^{K} \sum_{y \in Y_i} TBD(\mu_i, \mathrm{y}) \tag{4-13}$$

其中，K 是预先给定的簇数目，$\mu_i$ 是第 i 类簇的质心，计算公式如式 4-12。

初始质心的选择会对最后的求解结果产生很大的影响，在本文中,为了消除该影响，在同样的序列样本中多次运行聚类，在每次运行中随机选择一组不同的样本点作为初始质心，然后从多次运行的结果中选择具有最小 SSE 的聚类结果。

类簇的数目对聚类结果有较大的影响，本文中使用 GAP 统计量[4][6]。评估不同数目的簇对结果的影响，最终选择最优的簇数目。

### 4.3.4 Gap 统计量获得最优类簇数

Gap 统计量可以对样本数据的内部离散特性进行定量分析，确定最优类簇数。给定样本数据集$\{x_{ij}\}, i=1,2,\cdots,n, j=1,2,\cdots,m$，其中样本数据集中包含有 $n$ 个样本点，$m$ 表示每个样本点具有 $m$ 维特征属性。对于聚类结果中第 $p$ 类样本的集合$C_p$，所有属于该类的样本点的两两之间距离总和表示为：

$$D_p = \sum_{i,i' \in C_p} d_{ii'} \tag{4-14}$$

其中$d_{ii'}$表示样本点 $i$ 和 $i'$ 之间的距离度量，本文中使用 TBD 距离。

簇内离散度$W_k$定义为对于第 $p$ 类样本的集合$C_p$分别计算该类中所有样本点对于类质心的距离平方和，最后对所有的类进行加权平均。$W_k$ 公式表示为：

$$W_k = \sum_{p=1}^{k} \frac{1}{2n_p} D_p \tag{4-15}$$

簇内离散度$W_k$会随着簇数目 $k$ 的增加而单调减小，并且会有一个从快速减小到减小速度逐渐平缓的变化过程，簇内离散度$W_k$减小速度转变时的 $k$ 就是最优的簇数目。

定义如下 Gap 统计量：

$$Gap_n(k) = E_n^* \{\log(W_k)\} - \log(W_k) \tag{4-16}$$

Gap 统计量的含义如下:首先生成参考数据点数目为 $n$ 的参考零分布，并由此计算得到$\log(W_k)$的期望值$E_n^* \{\log(W_k)\}$衡量了样本点的簇内离散度的下降速度与参考数据点的内部离散度的下降速度的差。结合前述分析可知，最优簇数目 $k$ 将使$Gap_n(k)$取得最大值。

对于求解$E_n^* \{\log(W_k)\}$，通常做法是基于 Monte Carlo 方法重复生成 B 份参考零分布，将这 B 份参考零分布的簇内离散度$\log(W_k^*)$的平均值作为对期望值$E_n^* \{\log(W_k)\}$的估计，此处参考零分布数据的生成采用在观测样本点范围内随机

产生 $n$ 个数据点的方式。由于在实际中使用有限样本平均值对期望值进行估计，为了消除实验统计误差，引入修正因子 $s_k$：

$$s_k = \sqrt{(1+\frac{1}{B})}sd_k \tag{4-17}$$

其中 $sd_k$ 是这 B 份参考零分布簇内离散度 $\log(W_k^*)$ 的标准差：

$$sd_k = \sqrt{\frac{1}{B}\sum_b \left\{\log(W_{kb}^*) - \bar{l}\right\}^2},\ \bar{l} = \frac{1}{B}\sum_b \log(W_{kb}^*) \tag{4-18}$$

最优簇数目是满足 $Gap(k) \geq Gap(k+1) - s_{k+1}$ 的最小 $k$ 值。

利用 Gap 统计量对样本数据集中簇数目 k 进行确定的算法步骤如下：

1. 分别令簇数目 $k = 1,2,\ldots,K$（由于题干要求，本文中 K=30），利用聚类算法对给定的样本数据进行聚类，计算样本的簇内离散度 $W_k$。

2. 依据给定的规则生成 B 份参考数据集，分别令 $k = 1, 2, \ldots, K$，利用聚类算法对每一份生成的参考数据集进行聚类。因此对于第 b 份参考数据集及簇数目 k，都可以计算得到一个相应的簇内离散度量 $W_{kb}^*$，并依此计算得到 Gap 统计量：

$$Gap(k) = \frac{1}{B}\sum_b \log(W_{kb}^*) - \log(W_k) \tag{4-19}$$

3. 针对上一步计算得到的当簇数目为 $k$ 时的簇内离散度，计算它的标准差 $sd_k$ 及相应的修正参数 $s_k$。

4. 由 $k$ 从小到大增加的顺序进行搜索测试，选择满足 $Gap(k) \geq Gap(k+1) - s_{k+1}$ 的最小的 $k$ 值作为最优簇数目。

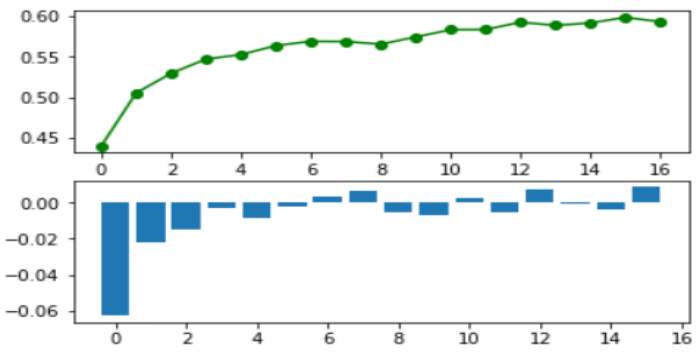
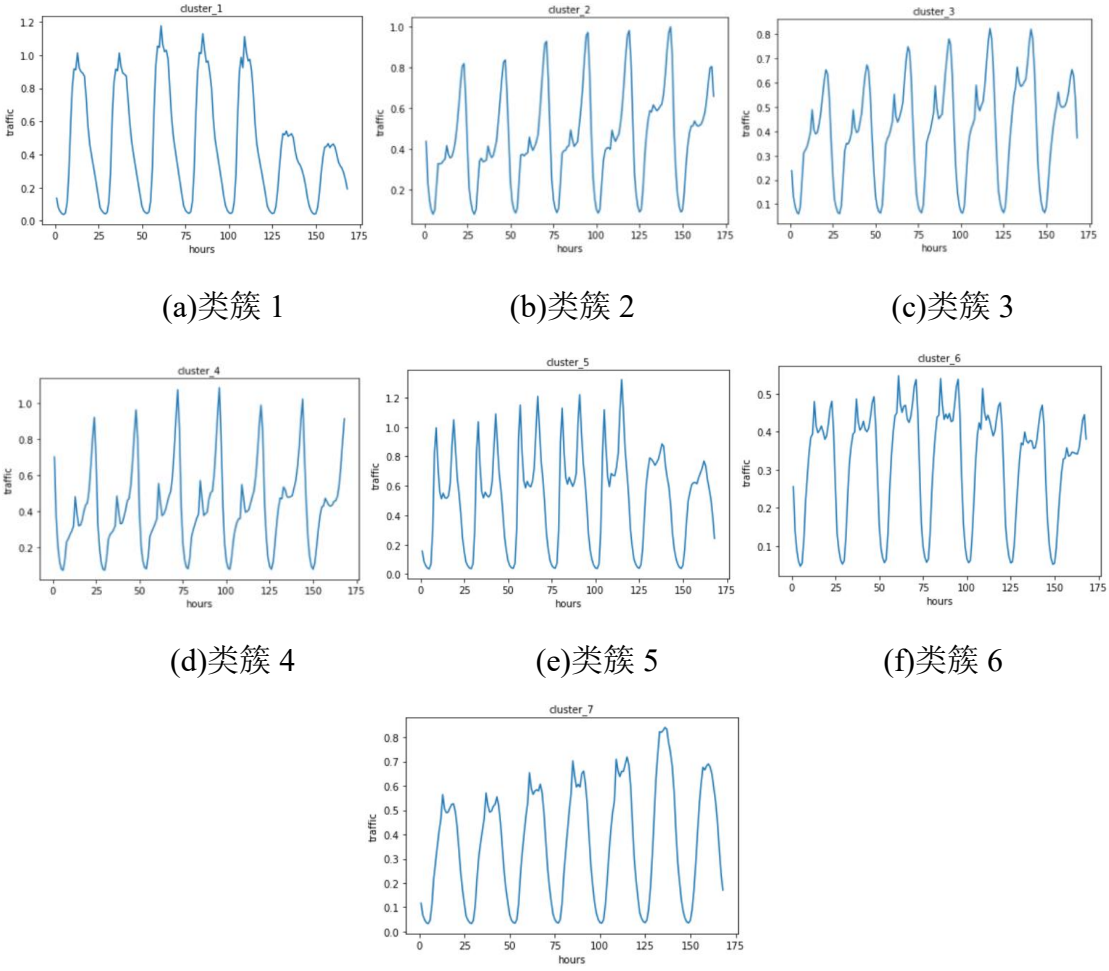



图 4.1 基于特征的 Gap 统计量

从图 4.1 可知，满足式 4-20 的最小的 $k$ 值为 7，因此最优簇数目确定为 7。

## 4.4 聚类结果分析

聚类得到的结果如图 4.2 所示：




(a)类簇 1　(b)类簇 2　(c)类簇 3

(d)类簇 4　(e)类簇 5　(f)类簇 6

(g)类簇 7

图 4.2 7 类小区聚类结果

分析图 4.3 发现，本文中聚类算法把小区流量时间序列划分成了 7 类，每个类簇的类簇中心曲线可以反映出这一类簇的小区流量特点。这七类小区在午夜至凌晨的时间段内流量水平较低，这与大部分人们的生活规律相符合。第一类小区每天的流量峰值在十二点附近，上午十点至晚上八点的流量水平较高，工作日相比周末流量值更高。第二类小区的流量峰值在晚上十点附近，白天流量值较低，晚上八点至十二点的流量水平较高，工作日和周末的流量值接近，周日的流量值稍低一些。第三类小区的流量峰值在晚上八点附近，中午十二点左右也有一个流

量的小峰值，在晚上七点至晚上十一点流量水平较高。第四类小区的流量峰值在晚上十点附近，晚上八点到晚上十二点的流量水平较高，其余时间流量水平较低。第五类小区的流量曲线呈现“U”型，在上午九点和晚上八点出现双峰值，其余时段的流量水平较低。第六类小区除了午夜至凌晨的时间段，其余时段的流量水平一直较高，在中午十二点和晚上十点各出现一个小峰值。第七类小区的整体流量水平较高，与第六类小区所呈现的流量曲线类似，区别在于，第六类小区的两个峰值更加明显，除峰值外的其他时段流量值更低，第七类小区的整体流量水平更高，峰值不明显，在中午十二点出现一个较小的流量峰值。

表 4.3 各类小区流量特点分析

| 类簇 | 本类簇小区流量特点 | 本类小区可能的地理区域属性 |
|---|---|---|
| 类簇 1 | 上午十点至晚上八点的流量水平较高，工作日流量高于周末 | 商务区，公司聚集地 |
| 类簇 2 | 白天流量值较低，晚上八点至十二点的流量水平较高，工作日和周末的流量值接近 | 住宅区 |
| 类簇 3 | 在晚上七点至十一点处于较高流量水平 | 商圈 |
| 类簇 4 | 晚上八点到晚上十二点的流量水平较高，其余时间流量水平较低 | 住宅区，商圈 |
| 类簇 5 | 在上午九点附近和晚上八点附近出现两个明显尖峰值 | 一线城市的地铁站 |
| 类簇 6 | 总体流量水平较高，在中午十二点和晚上十点各出现一个小峰值 | 大学校园 |
| 类簇 7 | 总体流量水平比类簇 6 更高，峰值不明显 | 商业区 |

由于小区流量数据反应了用户与基站之间的流量交互情况，故而大致可以反映出各个小区的覆盖范围内的人类活动情况。各类小区的流量特点和小区可能所处地理区域属性如表 3.3 所示。小区的地理区域属性与所属类簇并不是一一对应的关系，因为有些小区的区域位置属性存在重叠的现象，比如一个小区覆盖范围内既有住宅区，也有商务区以及购物中心等，所以小区流量变化会受到这些地理区域属性的综合影响，呈现出各种不同的流量特点。

# 五、问题二：基于流量预测的基站动态载频开关方案

## 5.1 问题二的求解思路

问题二考虑基站运行的潮汐现象，旨在设计一种基于流量预测的动态载频开关的基站配置方案以调整网络容量，提高网络能效。基于问题一中对各个小区流量特征的聚类结果，该部分对每一类簇中的小区流量负载进行 4 个等级的划分，包括超低负载，低负载，中负载以及高负载水平[7]。根据网络负载等级划分结果，对每个负载等级预设置相应的基站载频开关比例，即阈值。考虑到载频的关闭与启动操作可能耗费额外的时间和能耗[8][9][10]，因此为了避免频繁地进行智能载频关断操作和网络初始化，本文限制操作执行的间隔时间。根据潮汐效应的特征，在非工作时间段，由于负载较少，可以设置较长的更新间隔；在工作时间段，由于负载大且具有时变性，更新间隔较短。根据一天的网络负载变化用 Q 个时间点进行划分，形成 Q-1 个时间段。在设定好的载频开关操作时间点，根据下一时间段的网络流量预测情况以及实时负载情况综合判断，基站依照相应的流量负载等级中预设置的阈值自动控制并执行智能载频关断操作，直到下一个操作时间点重新进行开关设定，假设执行时间可以忽略不计。

本文联合考虑某小区用户终端功耗和基站功耗，分别对小区上行信道和下行信道建立能效模型，并设计二者的权重能效模型。以提高网络能效为目标，在小区内用户流量实时动态随机变化的情况下，灵活地确定合适的基站载频开启比例，改变网络流量负载，以此在保障用户服务质量的前提下降低系统能耗。

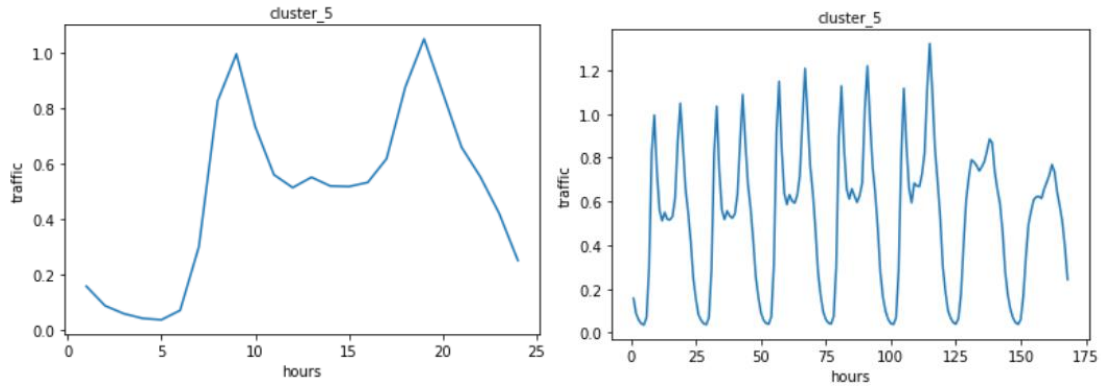
基于短期流量预测模型，对未来一天的小区流量进行可视化，并根据流量负载水平等级，关闭部分比例的基站载频，通过可视化一天内关闭载频的数目直观地展示动态开关载频方案的具体实施效果。为了进一步验证该方案的可行性和有效性，本文在同一网络场景和相同的参数设置下与固定按高负载时段载频配置方案进行系统总能耗和系统能效两方面对比验证。

## 5.2 模型建立与求解

### 5.2.1 载频动态开关时间点设置

根据潮汐效应特征，在合适的时间点进行动态载频开关的操作十分必要。如果操作过于频繁，需要不断进行网络初始化，会增加额外的网络开销；如果操作更新间隔太长，则会影响网络的工作状态和用户的服务质量，无法满足时变的流量负载需求[11]。因此本文结合工作和非工作时段分别设置不同的载频动态操作时间间隔，在工作时段，本文设置载频开关操作间隔为 1 小时，在非工作时段，设置操作间隔为 2 小时。

具体的潮汐现象展示如下，基于问题 1 中对十多万个小区的聚类结果，以类簇 5 为例，图 5-1 分别是该类簇一天内和一周七天内的网络流量变化趋势：




(a) 类簇 5 一天 24 小时的网络流量变化　　(b) 类簇 5 一周内的网络流量变化

图 5-1 潮汐效应网络流量变化图

由图 5-1 (a)可以看出，在工作时间 8:00-11:00，用户由于办公、视频等各类业务对通信流量的需求十分巨大；11:00-15:00 时网络流量稍有下降；18:00-21:00 流量再次达到新高峰，需要开启大部分载频用于增加网络容量，满足用户通信需求。在非工作时间大量用户产生的业务量降低，通信流量需求较低，可关闭部分载频以减少能耗。图 5-1 (b)展示了类簇 5 中所有小区一周内的平均流量变化趋势，可以看出由于“潮汐效应”的影响，用户流量存在周期性和较大幅度的波动。

### 5.2.2 网络流量负载水平等级划分

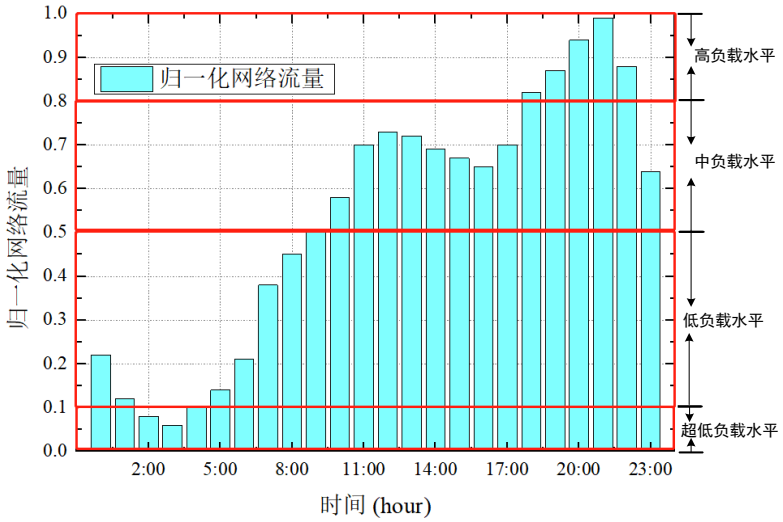



图 5-2 第 3 类簇质心一天内网络流量变化数据

图 5-2 为类簇 3 质心一天内小时级的归一化网络流量数据。设整个网络的归一化网络流量 $L \in [0,1]$，通过设置三个归一化流量负载门限 $\beta_1$， $\beta_2$， $\beta_3$，将该类簇中的流量负载水平划分为 4 个等级，包括超低负载，低负载，中负载及高负载水平，其中 $\beta_1$， $\beta_2$， $\beta_3$ 分别取值 10%，50%，80%，划分情况如表 5-1 所示：

表 5-1 网络负载水平等价划分

| 网络负载水平 | 归一化网络负载值 |
|---|---|
| 超低负载水平 | $L \in [0, \beta_1)$ |
| 低负载水平 | $L \in [\beta_1, \beta_2)$ |
| 中负载水平 | $L \in [\beta_2, \beta_3)$ |
| 高负载水平 | $L \in [\beta_3, 1]$ |

### 5.2.3 载频动态开关阈值设置

基本上，1 个移动基站分 3 个扇区，每个扇区的载波（载频）配置最大为 12 个。依照通过预测的网络流量水平和基站负载情况，对处于每个负载水平等级的时段，设置相应的基站载频开关比例参数，即阈值 z，具体的比例参数 $z_1, z_2, z_3, z_4$

分别对应超低负载、低负载、中负载和高负载水平四个等级。其中，z 的取值范围是[0,1]，若 z 取值 25%，则代表最多可以关闭基站中 25%的载频部分。具体阈值设置如表 5-2 所示：

表 5-2 载频开关阈值设置

| 下一操作时间点小区归一化流量预测 | 关闭载频比例参数（阈值） |
|---|---|
| [0,10%] | $z_1$:100% |
| [10%,50%] | $z_2$:50% |
| [50%,80%] | $z_3$:25% |
| [80%,1] | $z_4$:0 |

在某一载频动态操作时间点，获取当前时段小区内用户流量、载频状态信息以及载频关闭的比例参数 $z_n$，依照下一时段的流量预测水平 $L_p$，确定下一时段载频关闭的比例参数 $z_p$。如果 $z_p < z_n$，说明下一时段小区流量负载提升，需要开启部分载频；如果 $z_p > z_n$，下一时段小区流量负载降低，需要关闭更多的载频；$z_p = z_n$ 说明下一时段小区流量负载水平和当前一致，保持当前载频关闭比例不变。

### 5.2.4 系统能效模型

假设本文的系统模型是建立在全双工双层异构蜂窝网络的基础上[12]，由宏基站和 N 个小型基站组成，如图 5-3 所示。每个小区定义为由一个小型基站和基站覆盖范围内的多个终端用户组成的区域。

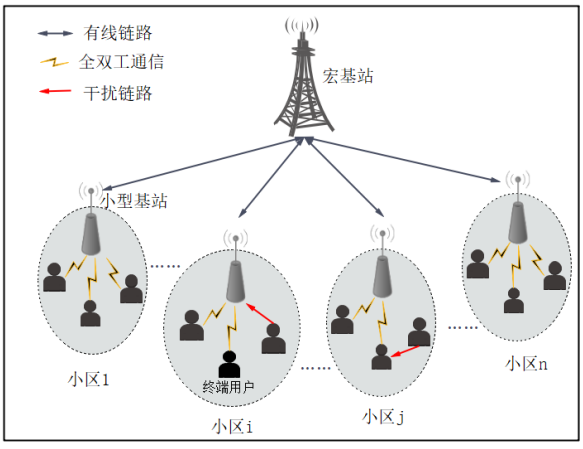


图 5-3 异构蜂窝网络模型

以 186 号小区为例，定义该小区内部的小型基站为 $f$，基站 $f$ 的载频集合为 $C=\{c_1,c_2,...,c_k\}$，某载频 $c_i$ 所服务的用户集合为 $U_{c_i}$。

### 5.2.4.1 上行信道能效模型

当基站和用户进行全双工通信时，用户 $u$ 与基站 $f$ 分配的载频 $c_i$ 间上行信道的信干噪比 $SINR_{u,c_i}^{UL}$ 为：

$$SINR_{u,c_i}^{UL}=\frac{P_u R_{u,c_i}^{-\alpha}}{\sum_{m=1,m\neq u}^{U_{c_i}} P_m R_{m,c_i}^{-\alpha}+P_{c_i}^{\max} h_{f,u}+N_0 W} \tag{5-1}$$

其中 $P_u$ 表示用户终端 u 的传输功率；$P_{c_i}^{\max}$ 表示载频 $c_i$ 的最大传输功率；$h_{f,u}$ 表示用户终端 u 受到来自基站 f 的干扰增益；$R_{u,c_i}^{-\alpha}$ 表示用户终端 u 通过分配的载频 $c_i$ 到基站 f 之间的路径损耗，$\alpha$ 表示路损指数；$\sum_{m=1,m\neq u}^{U_{c_i}} P_m R_{m,c_i}^{-\alpha}$ 表示用户终端 u 受到来自载频 $c_i$ 服务的其他用户的干扰；$N_0$ 表示高斯白噪声信道的功率谱密度；载频带宽为 $W$ Hz。

用 $R_{u,c_i}^{UL}$ 表示载频 $c_i$ 服务的用户终端 u 的上行信道容量为

$$R_{u,c_i}^{UL}=W\log(1+SINR_{u,c_i}^{UL}) \tag{5-2}$$

用户终端的功耗模型为

$$P_{UE}=P_{CU}+\zeta P_u \tag{5-3}$$

其中，$P_{CU}$ 表示用户终端的固定消耗；$\zeta$ 表示功率消耗系数。

用户 u 的能效为

$$\eta_u=R_{u,c_i}^{UL}/P_{UE} \tag{5-4}$$

则上行信道的总能效为异构蜂窝网络中所有用户能效之和，上行信道的能效模型为

$$\eta^{UL} = \sum_{i=0}^{k} \tau_i \cdot \sum_{u=1}^{U_{c_i}} \frac{R_{u,c_i}^{UL}}{P_{UE}} = \sum_{i=0}^{k} \tau_i \cdot \sum_{u=1}^{U_{c_i}} \frac{W \log(1+SINR_{u,c_i}^{UL})}{P_{CU}+\zeta P_u} \tag{5-5}$$

$\tau_i$ 表示载频 $c_i$ 开启时为 1，关闭时为 0。于是，有

$$\sum_{i=0}^{k} \tau_i = z \cdot k \tag{5-6}$$

式（5-6）用来计算基站 f 开启的载频数目，表示当前基站可负载流量，其中 k 为基站 f 总载频数， z 为载频开关比例。

### 5.2.4.2 下行信道能效模型

当基站 f 和用户终端 u 进行全双工通信时，基站所分配的载频 $c_i$ 到用户的下行信道的信干噪比 $SINR_{f,c_i}^{DL}$ 为（忽略其他基站对基站 f 的通信干扰）：

$$SINR_{c_i,u}^{DL} = \frac{P_{c_i} R_{c_i,u}^{-\alpha}}{\sum_{j=1,j\neq i}^{k} \tau_j P_{c_j} R_{c_j,u}^{-\alpha} + P_{c_i}^{\max} h_{f,u} + N_0 W} \tag{5-7}$$

其中，$\sum_{j=1,j\neq i}^{k} \tau_j P_{c_j} R_{c_j,u}^{-\alpha}$ 表示载频 $c_i$ 受到其他载频 $c_{j=1,j\neq i}$ 与用户 u 通信时的干扰。

用 $R_{c_i,u}^{DL}$ 表示载频 $c_i$ 服务的用户终端的 u 的下行信道容量，表示为：

$$R_{c_i,u}^{DL} = W \log(1+SINR_{c_i,u}^{DL}) \tag{5-8}$$

异构蜂窝网络内该小区载频的总功耗为

$$P_{BS} = \sum_{i=1}^{k} P_{c_i} = \sum_{i=1}^{k} \tau_j \cdot \left( P_{const} + \rho_{c_i} \frac{P_{c_i}^{\max}}{\varphi_{c_i}} \right) \tag{5-9}$$

其中， $P_{const}$ 表示载频 $c_i$ 处于工作状态时的固定功耗； $\rho_{c_i}$ 表示当前载频 $c_i$ 的负载因子，且 $\rho_{c_i} = a_i / A_{c_i}^{\max}$ ， $a_i$ 为基于预测的下一时间点小区网络流量， $A_{c_i}^{\max}$ 为载频 $c_i$ 可负载最大网络流量， $\varphi_{c_i}$ 表示载频 $c_i$ 的直流到射频的转换因子。

下行信道模型表示为

$$\eta^{DL}=\sum_{i=1}^{k}\tau_j\cdot\sum_{u=1}^{U_f}\frac{R_{c_i,u}^{DL}}{P_{BS}}=\sum_{i=1}^{k}\tau_j\cdot\sum_{u=1}^{U_f}\frac{W\log(1+SINR_{c_i,u}^{DL})}{\sum_{i=1}^{k}P_{c_i}} \tag{5-10}$$

### 5.2.4.3 系统总功耗模型

基于用户终端的功耗模型和活跃载频的功耗模型，本文定义系统总功耗为：

$$\begin{aligned}P_{total}&=P_{UE}+P_{BS}=P_{CU}+\zeta P_u+\sum_{i=1}^{k}P_{c_i}\\&=P_{CU}+\zeta P_u+\sum_{i=1}^{k}\tau_j\cdot\left(P_{const}+\rho_{c_i}\frac{P_{c_i}^{\max}}{\varphi_{c_i}}\right)\end{aligned} \tag{5-11}$$

### 5.2.4.4 系统权重能效模型

基于上行信道能效和下行信道能效模型，本文定义二者的权重能效模型，可表示为：

$$\begin{aligned}\eta&=\omega\eta^{UL}+(1-\omega)\eta^{DL}\\&=\omega\sum_{i=0}^{k}\tau_i\cdot\sum_{u=1}^{U_f}\frac{W\log(1+SINR_{u,c_i}^{UL})}{P_{CU}+\zeta P_u}+(1-\omega)\sum_{i=1}^{k}\tau_j\cdot\sum_{u=1}^{U_f}\frac{W\log(1+SINR_{c_i,u}^{DL})}{\sum_{i=1}^{k}P_{c_i}}\end{aligned} \tag{5-12}$$

其中$0<\omega<1$表示能效权重因子。

根据不同时段小区流量负载水平的变化，动态设置基站载频的开关比例能有效地提高网络能效，即能效值$\eta$会随着$\sum_{i=0}^{k}\tau_i$值的不同而动态变化。

## 5.2.5 动态载频效果仿真

为了验证本文基于不同网络流量负载水平等级中设置的载频开关比例参数，即阈值 z 对提升整个系统能效的有效性，以下内容分别从活跃载频数目，系统总能耗以及系统能效等性能指标来进行衡量和分析。

考虑某场景下包括一个覆盖范围为 50m 的小型基站和多个终端用户。区域内的用户都与基站进行全双工通信，具体仿真参数设置[7]如表 5-3：

表 5-3 系统能效参数设置

| 参数名称 | 参数值 | 参数名称 | 参数值 |
| --- | --- | --- | --- |
| 基站的载频配置数目 k | 36 | 路损指数 $\alpha$ | 3.4 |
| 载频带宽 $W$ /MHz | 10 | 载频 $c_i$ 最大传输功率 $P_{c_i}^{\max}$ / dBm | 21 |
| 高斯白噪声信道的功率谱密度 $N_0$ /W/Hz | $10^{-11}$ | 载频 $c_i$ 直流到射频转换因子 $\varphi_{c_i}$ | 0.045 |
| 用户传输功率 $P_u$ /dBm | 23 | 载频 $c_i$ 可负载最大网络流量 $A_{c_i}^{\max}$ | 2 |
| 干扰增益 $h_{f,u}$ / dBi | 5 | 载频 $c_i$ 工作状态固定功耗 $P_{const}$ /W | 4.8 |
| 用户固定消耗功率 $P_{CU}$ /W | 0.01 | 效率权重因子 $\omega$ | 0.6 |
| 功率消耗系数 $\zeta$ | 0.3 | | |

（1）活跃载频数目

以 186 号小区为例，图 5-4 展示了基于短期预测模型得到的 186 号小区 4 月 20 号网络预测流量随时间的变化情况，根据不同流量负载水平对本日流量进行四个等级的划分。

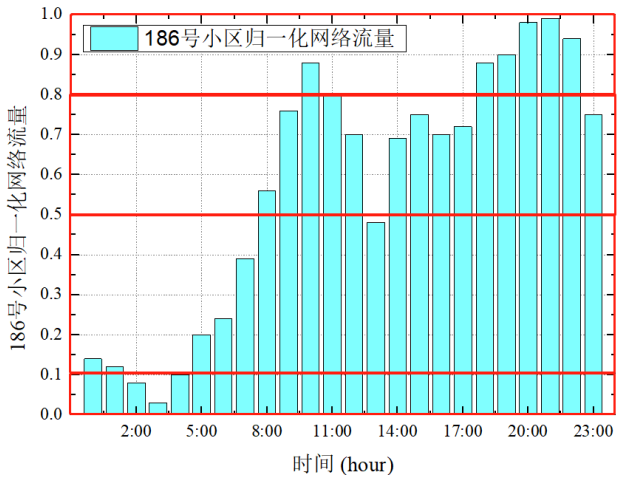



图 5-4 186 号小区一天内网络流量随时间的变化情况

可以看出一天之内只有少数时间（10:00-11:00、18:00-22:00）流量处于高负载水平，如果固定按高负载时段的载频数量来运行基站配置会造成浪费。因此，需要根据不同的网络负载水平关闭相应数目的载频来降低系统能耗。图 5-5 展示了 186 号小区 4 月 20 号基于流量预测的载频关闭数目。

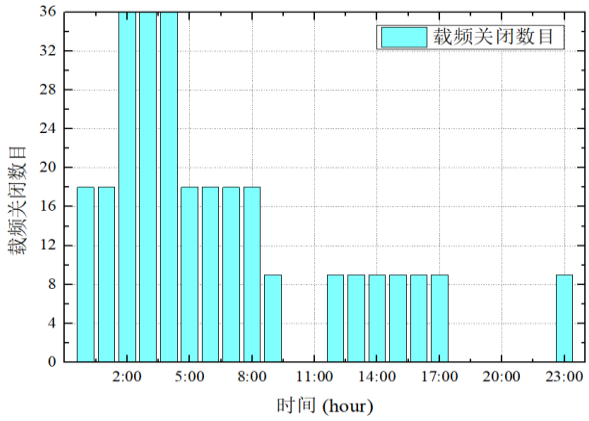



图 5-5 186 号小区基站一天内关闭载频数目随时间的变化情况

通过利用基于预测的动态载频开关算法进行设置，当网络负载处于超低负载水平时，如 2:00-4:00，基站中所有载频处于关闭状态；当网络负载处于高负载水平时，基站中所有载频处于活跃状态。大部分时段都会有一定数量的载频处于关闭状态，只有少数时段，如 18:00-22:00 时所有载频处于活跃状态。因此根据流量变化设置自动开关限制部分载频的能够很大程度上降低网络总能耗。

（2）系统总能耗

从系统总能耗的角度分析，将本文提出的基于预测的动态载频开关方案与固定按高负载时段载频配置方案进行比较。图 5-6 显示了一天内两种方案下系统总功耗的对比情况，网络总功耗随着网络负载水平的不同而变化，在 2:00-4:00 时间段，由于网络负载水处于一天之内的最低水平，所以网络总功耗较低，而在 18:00-22:00 期间，由于网络负载升至高负载水平，网络总功耗达到一天的峰值。

由系统总功耗对比情况可以看出通过根据网络不同时段的负载水平进行基站载频开关设置，将活跃载频数目动态调整，能够使得系统整体的功耗水平降低，特别是在网络处于超低负载水平的 2:00-4:00，系统总功耗水平有着明显的降低。

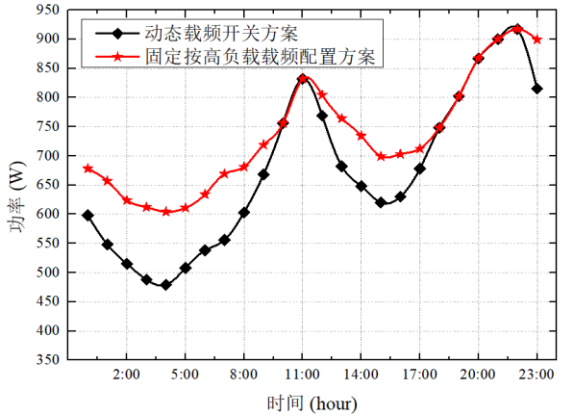



图 5-6 两种方案下的网络功耗随时间的变化情况

通过使用系统总能耗降低比率 $\phi$ 可以衡量动态载频开关方案的有效性，系统能效提升比率 $\phi$ 定义为：

$$\phi=\frac{P_{\text{total}}-P'}{P'} \tag{5-13}$$

其中 $P_{\text{total}}$ 表示动态载频开关方案下的系统总能耗，$P'$ 表示固定按高负载时段载频配置方案下的系统总能耗。图 5-7 显示了系统总能耗降低比率随时间的变化情况，在 2:00-4:00 时段之间，网络负载处于较低水平，动态载频开关能大幅降低系统总能耗，节能效果更加显著。随着网络负载水平的提高，部分关闭的载频被打开，系统总能耗降低比率逐渐下降；18:00-22:00 时网络处于高负载水平，为了保障用户服务质量，需要将全部载频开启，此时系统总能耗降低比率为 0。

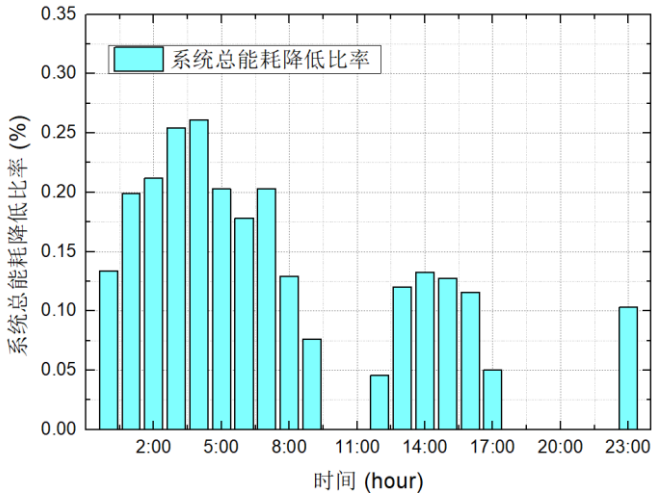



图 5-7 系统总能耗降低比率随时间的变化情况

（3）系统能效

从系统能效的角度分析，将本文提出的基于预测的动态载频开关方案与固定按高负载时段载频配置方案进行比较。图 5-8 显示了在一天内两种方案下系统能效的对比情况，系统能效随着网络负载水平和系统总能耗的不同而变化，在 2:00-5:00 期间达到最低水平，在 19:00-22:00 期间达到最高水平。从整体上看，相比于固定按高负载时段载频配置方案，采用基于预测的动态载频开关方案能显著提升系统能效。

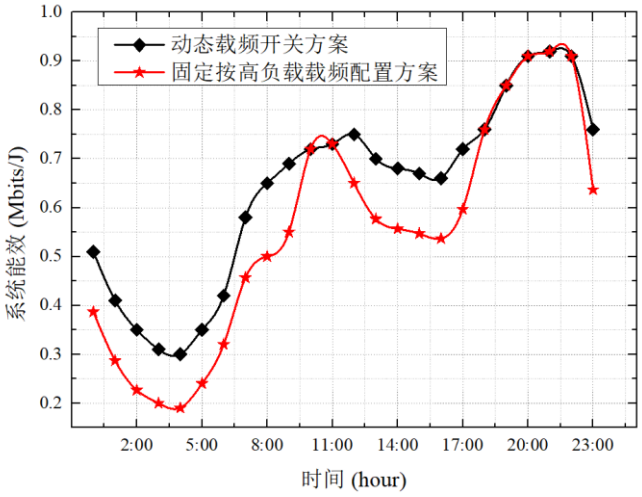



图 5-8 两种方案下的系统能效随时间的变化情况

通过使用系统能效提升比率$\psi$可以衡量动态载频开关方案的有效性，系统能效提升比率$\psi$定义为：

$$\psi=\frac{\eta-\eta'}{\eta'} \tag{5-14}$$

其中$\eta$表示动态载频开关方案下的系统能效，$\eta'$表示固定按高负载时段载频配置方案下的系统能效。图 5-9 显示了系统能效提升比率随时间的变化情况，可以看出在 2:00-4:00 时段之间，动态载频开关方案对系统能效有着大幅的提升。随着网络负载水平的提高，部分关闭的载频被打开，系统能效提升比率逐渐下降；在 18:00-22:00 时段网络处于高负载水平，所由载频处于开启状态，此时系统能效提升比率为 0。

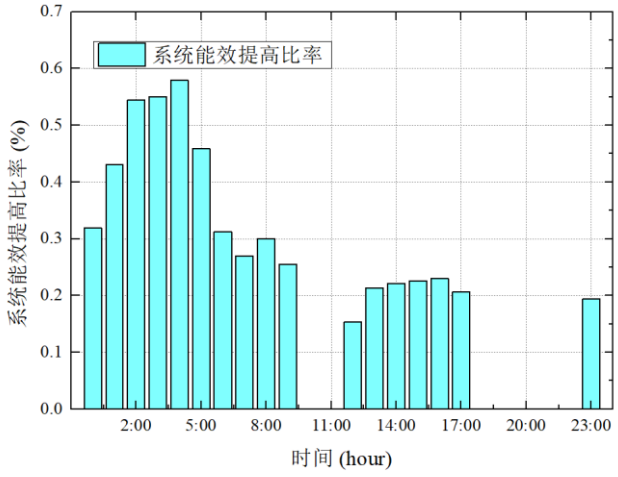



图 5-9 系统能效提升比率随时间的变化情况

# 参考文献


[1] 廖书妍.数据清洗研究综述[J].电脑知识与技术,2020,16(20):44-47.

[2] 赵一凡,卞良,丛昕.数据清洗方法研究综述[J].软件导刊,2017,16(12):222-224.

[3] 夏慧维. 基于决策树集成和宽度森林的网络流量分析与预测研究[D].南京邮电大学,2020.

[4] Qi C, Zhao Z, Li R, et al. Characterizing and modeling social mobile data traffic in cellular networks[C]//Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd. IEEE, 2016: 1-5.

[5] 蒋品. 基于机器学习的蜂窝网络基站流量分析与预测研究[D].北京邮电大学,2019.

[6] 张雁钦. 移动通信网流量数据分析及预测研究[D].中国科学技术大学,2016.

[7] 任嘉鹏. 基于机器学习的流量预测及基站休眠方法研究[D].吉林大学,2020.

[8] Heyman D, Lakshman T M Tabatabai A, et al. Modeling teleconference traW c from VBR video coders[C]. Proceedings of ICC/SUPERCOMM'94 Tnternational Conference on Communications. IEEE, I994,3: 1744-1748.

[9] K. Chang, K. Chu, H. Wang, Y. Lin and J. Pan, "Energy Saving Technology of 5G Base Station Based on Internet of Things Collaborative Control," in IEEE Access, vol. 8, pp. 32935-32946, 2020.

[10] H. Wang, Z. Zhao, X. Cheng, J. Ying, J. Qu and G. Xu, "Base Station Sleeping Strategy for On-Grid Energy Saving in Cellular Networks With Hybrid Energy Supplies in IoT Environment," in IEEE Access, vol. 6, pp. 45578-45589, 2018.

[11] G. Jang, N. Kim, T. Ha, C. Lee and S. Cho, "Base Station Switching and Sleep Mode Optimization With LSTM-Based User Prediction," in IEEE Access, vol. 8, pp. 222711-222723, 2020.

[12] G. Zhang, Y. Cao and D. Li, "Energy Cost Reduction for Hybrid Energy Supply Base Stations with Sleep Mode Techniques," 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, 2018, pp. 1-6.

# 附录

使用软件名称：spyder
命令：python xx.py

问题一源程序：

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt

#############################################################################
# 读取数据
train = pd.read_csv('train.csv', encoding='gbk')
new_col = ['DATE', 'HOUR', 'NAME', 'LABEL', 'LABEL2']
train.columns = new_col

#删除训练数据中重复列
train = train.drop_duplicates(subset=['DATE','HOUR','NAME'], keep='first')
train['HOUR'] = train['HOUR'].apply(lambda x: int(x.split(':')[0]))
train['DAY'] = train['DATE'].apply(lambda x: int(x.split('/')[-1][-2:]))
train['MON'] = train['DATE'].apply(lambda x: int(x[5]))
train.head()

# 表示上行流量和下行流量总和
train['LABEL'] = train['LABEL'] + train['LABEL2']
train = train.drop('LABEL2',axis=1)
train.head()

# 生成 weekday 标签，并且使用 1、2、3、4、5、6、7 分别表示周一至周天
train['WEEKDAY'] =    (train['DAY'] + (train['MON']-3)*31 + 3) % 7
train = train.drop(['DAY', 'MON'],axis=1)
train['WEEKDAY'] =    train['WEEKDAY'].apply(lambda x: 7 if x==0 else x)
train.head()

# 生成 train_1.csv 文件，用以下一步生成生成的七天小时级流量序列
train.to_csv('./train_1.csv', index=False)
train.info()


#############################################################################
# 读取数据
train = pd.read_csv('train_1.csv', encoding='gbk')
```

```
train.head()
train_name = train.NAME.unique()  # 用 numpy 数组形式生成保存每个小区的名字

# 对每个小区生成七天小时级平均流量
result = []
for i in train_name:
    data = train[train.NAME==i]
    print(i)
    res_temp = np.zeros(168)
    if len(data)<168:
        result.append(res_temp)
        continue
    for ii in range(7):
        data1 = data[data.WEEKDAY==ii+1]
        if len(data1)==0 : continue
        for jj in range(24):
            data2 = data1[data1.HOUR==jj]['LABEL'].values
            if len(data2)==0 : continue
            data3 = np.mean(data2)
            res_temp[ii*24+jj] = data3
    result.append(res_temp)

# 将生成的七天小时级流量序列保存到 test.csv 文件中
name = list(range(7*24))
test=pd.DataFrame(columns=name,data=result)
print(test.info())
test.to_csv('./test.csv', index=False)

#############################################################################
# 读取生成的七天小时级流量序列数据
data = pd.read_csv('test.csv', encoding='gbk')
name = list(range(1, 7*24+1))
test=pd.DataFrame(columns=name,data=data)
print(test.info())

# 进行缺失值处理操作
test_temp = test.replace(0,np.nan)
data_fina1 = test_temp.dropna()
data_fina1.info()

# 提取特征
# 差分特征
res_diff = []
leng = len(data_fina1)
```

```
for i in range(leng):
        dp = data_fina1.iloc[i,:]
        b = dp.diff()
        b[1] = dp[168] - dp[1]
        res_diff.append(b.values)

# 归一化
for i in range(len(res_diff)):
        Min = np.min(res_diff[i])
        Max = np.max(res_diff[i])
        res_diff[i] = (res_diff[i] - Min) / (Max - Min)

# RANK 向量
res_rank = []
leng = len(data_fina1)
for i in range(leng):
        dp = data_fina1.iloc[i,:]
        res = dp.rank().values
        res_rank.append(res)

# 归一化
for i in range(len(res_rank)):
        Min = np.min(res_rank[i])
        Max = np.max(res_rank[i])
        res_rank[i] = (res_rank[i] - Min) / (Max - Min)

# 进行特征融合
feature    = np.asarray(res_rank) * 0.6 +    np.asarray(res_diff) * 0.4


##########################################################################################
###############
# 分析最佳的 K 取值
# SSE = []
# for i in range(30):
#         km = KMeans(n_clusters=i+1,random_state=100)
#         km.fit(feature)
#                 #获取 K-means 算法的 SSE
#         SSE.append(km.inertia_)
#         print(i, SSE[i])
#
# import matplotlib.pyplot as plt
# plt.plot(range(1,31),SSE[0:30],marker="o")
# plt.xlabel("K",size=10)
```

```
# plt.ylabel("SSE",size=10)

# 利用 GAP 统计量计算最佳 k 值
import scipy
import scipy.cluster.vq
import scipy.spatial.distance
import numpy as np
EuclDist = scipy.spatial.distance.euclidean
def gapStat(data, resf=None, nrefs=10, ks=range(1,10)):
    '''
    Gap statistics
    '''
    # MC
    shape = data.shape
    if resf == None:
        x_max = data.max(axis=0)
        x_min = data.min(axis=0)
        dists = np.matrix(np.diag(x_max-x_min))
        rands = np.random.random_sample(size=(shape[0], shape[1], nrefs))
        for i in xrange(nrefs):
            rands[:,:,i] = rands[:,:,i]*dists+x_min
    else:
        rands = refs
    gaps = np.zeros((len(ks),))
    gapDiff = np.zeros(len(ks)-1,)
    sdk = np.zeros(len(ks),)
    for (i,k) in enumerate(ks):
        (cluster_mean, cluster_res) = scipy.cluster.vq.kmeans2(data, k)
        Wk = sum([EuclDist(data[m,:], cluster_mean[cluster_res[m],:]) for m in
xrange(shape[0])])
        WkRef = np.zeros((rands.shape[2],))
        for j in xrange(rands.shape[2]):
            (kmc,kml) = scipy.cluster.vq.kmeans2(rands[:,:,j], k)
            WkRef[j] = sum([EuclDist(rands[m,:,j],kmc[kml[m],:]) for m in range(shape[0])])
        gaps[i] = scipy.log(scipy.mean(WkRef))-scipy.log(Wk)
        sdk[i] = np.sqrt((1.0+nrefs)/nrefs)*np.std(scipy.log(WkRef))

        if i > 0:
            gapDiff[i-1] = gaps[i-1] - gaps[i] + sdk[i]
    return gaps, gapDiff


#############################################################################
# 加载数据
```

```
def loadDataSet(fileName):
    data = np.loadtxt(fileName, delimiter='\t')
    return data


# 欧氏距离计算
def distEclud(x, y):
    return np.sqrt(np.sum((x - y) ** 2))  # 计算欧氏距离


# 为给定数据集构建一个包含 K 个随机质心的集合
def randCent(dataSet, k):
    m, n = dataSet.shape
    centroids = np.zeros((k, n))
    for i in range(k):
        index = int(np.random.uniform(0, m))  #
        centroids[i, :] = dataSet[index, :]
    return centroids


# K 均值聚类算法
def KMeans(dataSet, k):
    m = np.shape(dataSet)[0]  # 行的数目
    # 第一列存样本属于哪一簇
    # 第二列存样本的到簇的中心点的误差
    clusterAssment = np.mat(np.zeros((m, 2)))
    clusterChange = True

    # 第 1 步 初始化 centroids
    centroids = randCent(dataSet, k)
    while clusterChange:
        clusterChange = False

        # 遍历所有的样本（行数）
        for i in range(m):
            minDist = 100000.0
            minIndex = -1

            # 遍历所有的质心
            # 第 2 步 找出最近的质心
            for j in range(k):
                # 计算该样本到质心的欧式距离
                distance = distEclud(centroids[j, :], dataSet[i, :])
                if distance < minDist:
```

```
                    minDist = distance
                    minIndex = j
            # 第 3 步：更新每一行样本所属的簇
            if clusterAssment[i, 0] != minIndex:
                clusterChange = True
                clusterAssment[i, :] = minIndex, minDist ** 2
        # 第 4 步：更新质心
        for j in range(k):
            pointsInCluster = dataSet[np.nonzero(clusterAssment[:, 0].A == j)[0]]  # 获取簇类所有的点
            centroids[j, :] = np.mean(pointsInCluster, axis=0)  # 对矩阵的行求均值

    print("Congratulations,cluster complete!")

    return centroids, clusterAssment


#多维特征数据进行聚类分析
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=7)
kmeans.fit(feature)
labels = kmeans.predict(feature)
centroids   = kmeans.cluster_centers_

# 求每个类簇的流量的七天平均值
res = []
for i in range(7):
    print(i)
    ans = []
    for j in range(len(data_fina1)):
        if labels[j]==i :
            ans.append(data_fina1[data_fina1.index==index[j]].values)
    res.append(np.mean(ans, axis=0))


# 画图表示
plt.plot(range(1,49), res[0][0][0:48])
plt.title("cluster_1",size=10)
plt.xlabel("hours",size=10)
plt.ylabel("traffic",size=10)
```

问题二源程序：

```
import pandas as pd
import numpy as np
import os

from pylab import *
import matplotlib.pyplot as plt
mpl.rcParams['font.sans-serif'] = ['SimHei']

# # 读取 186 小区的流量序列数据
# train = pd.read_csv('data_186.csv', encoding='gbk')
# data = train.LABEL1.values[72:96] + train.LABEL2.values[72:96]
# train.info()
# train.head()
# plt.plot(train.LABEL1.values[0:168]*10)
# plt.plot(train.LABEL2.values[0:168])
# plt.title("186 号小区的上行和下行流量序列",size=10)
# plt.xlabel("小时",size=10)
# plt.ylabel("流量",size=10)
#
# # 求上行流量和下行流量的自相关性
# u1 = np.append(train.LABEL1.values[0:168], train.LABEL2.values[0:168])
# # np.around(np.corrcoef(u1), decimals=3)
# np.corrcoef(u1)
#
# # 计算前一周和后一周的自相关系数
# u1 = np.append(train.LABEL1.values[0:168], train.LABEL2.values[168:168+168])
# np.corrcoef(u1)

# 设置阈值参数
b1, b2, b3 = 0.1, 0.5, 0.8 # 设置流量阈值比例，分别对应超低，低，中和高
z1, z2, z3, z4 = 1, 0.5, 0.25, 0   # 设置载频最多可休眠的比例

# 获取 186 小区载频开关实际情况
# data_3_label12.csv 存放 186 小区对应的 NAME、LABEL1 和 LABEL2 标签
data_temp = pd.read_csv('data_3_label12.csv', encoding='gbk')
data_temp.head()
data_label1 = data_temp.LABEL1.values[24:48]
data_label2 = data_temp.LABEL2.values[24:48]
data = data_label1 + data_label2

# 设置系统能效上行通道参数设置
K = 16 # 最大载频数
Hfu = pow(10, -4) + pow(10, -0.9)*5   # 高斯白噪声和基站干扰
Uf = 10 # 用户总数
```

```
Ruc =    0.2# 路径损耗
Puc = 0.01 # 用户固定消耗功率
Xu =    0.3         # 用户功率消耗系数
Pumax = 10    # 用户最大发射功率
traffic_upmax =    max(data_label1) # 上行最大
# Pcmax = 21 # 载频最高传输功率

# 计算上行功率和能效
traffic_dwmax = max(data_label2) # 下行最大负载
traffic_max = max(data) # 最大流量
print(traffic_max)
plt.plot(data)
t = np.zeros(24) # 载频开关实际情况
for i in range(24):
        if data[i]>=b3*traffic_max: t[i] = 0
        elif data[i]<b3*traffic_max and data[i]>=b2*traffic_max: t[i] = 0.25
        elif data[i]<b2*traffic_max and data[i]>=b1*traffic_max: t[i] = 0.5
        else: t[i] = 1

# 上行功率能耗
Pu =    Pumax * data_label1 / traffic_upmax # 用户发射功率，与负载变化相关
Pd1 = Ruc * Pu * Uf + Hfu
Pue = Puc + Xu * Pu
# print(Pue)
# plt.plot(data_label1)
plt.plot(range(len(Pue)), Pue,marker = "o" ) # 一个用户 24 小时的上行能耗
# Pdu_all = Pu

# 上行信道能效
SIRUL = Ruc * Pu / Pd1
ans_uci = pow(10,1) * np.log2(1+SIRUL) # 上行信道容量
ans1 = Uf * ans_uci / Pue
Nul = K * (1-t) * ans1
plt.plot(Nul,marker = "o")

# 设置系统能效下行通道参数设置
Hfu = pow(10, -4) + pow(10, -0.9)*5    # 高斯白噪声和基站干扰
Umax = 10 # 用户总数
Rcu =    0.2# 路径损耗
Pconst = 4.8 # 载频工作固定损耗
Xc = 0.045 # 直流到射频转换因子

# 下行功率能耗
Pcmax = pow(10, -0.9) #21 # 载频最高传输功率
```

```
Pc =    Pconst + Pcmax * data_label2 / traffic_dwmax # 基站发射载频功率，与负载变化相关
Pbs = K * (1-t) * Pc #  基站总功耗
plt.plot(data_label2, marker = "o" )
figure()
plt.plot(Pbs,marker = "o" )

# 下行信道能效
SINDL = Pc * Rcu / (Hfu + Pbs)
ans_ciu = pow(10,1) * np.log2(1+SINDL)
ans2 = Uf * ans_ciu / Pbs
print(ans2)
print(1-t)
Ndl = K * (1-t) * ans2
plt.plot(Ndl,marker = "o")
figure()
plt.plot(Ndl+Nul,marker = "o")
```