队伍编号:	244
赛道:	A

基于时间序列聚类的小区分类及基站智能开关载频的研究

摘　要

随着移动通信技术的发展，基站作为移动通信的基础设施，其承载网络流量持续增加．本文建立基于流量时间序列特征的聚类模型对小区进行分类，基于基站智能开关载频算法对基站载频的流量阈值进行研究．该问题的研究能智能调节基站载频数，降低基站节能减排效率，符合中国能源结构的发展趋势．

针对问题一，主要需要对小区进行分类．本文建立了二级聚类模型，首先我们对各小区流量的消耗量进行分析，利用消耗量的评级对小区进行第一级聚类划分．对附件一中的数据预处理，选取 6 个与小区流量消耗有关的特征：“小区日均上行业务量 GB”、“小区上行业务量大于 1GB 的天数”、“小区上行业务量小于 0.2GB”、“小区日均值下行业务量”、“小区下行业务量大于 1GB 的天数”、“小区下行业务量小于 0.2GB 的天数”．基于流量消耗特征对小区进行一级聚类，将小区划分为“活跃类”、“较活跃类”、“较平淡类”、“平淡类”．接着，选取 12 个有关流量时间序列的特征：“上行昼夜差均值”、“上行周差均值”、“上行近似熵”、“上行样本熵”、“上行白天消耗流量序列方差”以及对应的下行流量数据特征．基于时间序列特征对小区进行二级聚类，最终得到 16 个小区类型，并对每一类小区特点进行说明．

针对问题二，需要给出基站载频的流量阈值的设置策略和具体结果．我们对基站能耗和用户服务这两个指标进行研究，基站能耗主要从基站业务负载和基站静态能耗上考虑，基站业务负载为小区用户使用的流量总和，基站静态能耗为维持基站运行的最小功耗．根据小区的业务量与基站分配的载频容量差值，确立了一种基站的动态分配载频机制，使基站分配给小区的载频数既能满足用户流量需求，也使基站能耗最低．基于基站能耗和服务质量这两个指标，采用线性加权法，对蜂窝网络整体综合评价，选择适当的权重，使得达到基站能耗和用户服务质量两者达到最优值．由于数据的限制，我们仿真计算时将模型简化为单基站多小区型．提取 186，221，546 等小区一周的数据，通过仿真得到载频阈值，最终确定了该基站开关流量阈值 2.485GB.

最后，本文对第二问的模型进行稳定性检验，对不减函数进行多种函数的取样测试．结果得到在选取不同函数时，网络整体的评价趋势与得分都较为相似，由此可以得出该模型具有一定的稳定性．另外，我们对模型进行了优、缺点评价．最后经过分析检验，本文的模型具有合理性和一定的现实意义．

关键词：时间序列；特征提取；聚类；载频分配；基站节能

目 录

一、问题重述

1.1 问题的背景

随着移动通信技术的发展，移动端设备的大范围，4G、5G 给人们带来了极大的便利．移动互联网的快速发展，人们对于移动流量的需求不断增加．在 2015 年思科发布的关于流量的预测报告中显示，直至 2020 年，移动通信网的每月流量将会增长到 30.6 Ebyte, 相比于 2016 年翻了接近 5 倍[1]．当今，移动互联网和物联网逐渐兴起，无线通信系统的业务类型重点逐渐从语音通话演变为数据业务，数据业务的需求急速提升，于此同时，无线通信系统也面临也更加严峻的能源消耗问题．基站作为承载网络流量的基础设施，其流量负荷问题变得越来越重要．一方面，在流量高峰期，大量基站呈现出负荷超过容量的问题，使得即使信号条件很好，网络速度也非常的慢，给用户带来了非常不好的体验．通常需要给基站增加载频的数量来扩容，使得基站可以承载更多的流量；另一方面，由于基站潮汐现象，使得在某些时段，用户数量会大幅度降低．

在基站低流量时段，如果仍然按照高容量时段的在载频数量来运行基站的配置，会极大的浪费资源和能量．特别是现在每个城市的基站数量巨大，而且随着 5G 的不断部署，基站数量还会大幅增加．同时，每个基站的流量高峰和低估的时段各不相同．如果所有基站都按照高容量时段来配置运行载频，则网络的能量消耗是非常巨大的．因此需要根据流量的变化，计算需要的载频数量，从而可以在不同时段打开或者关闭部分载频使得既可以满足对用户的服务，又可以尽可能低的消耗能量和资源．

由于基站数量巨大，无法通过人工实时关注每个基站的流量变化，需要给每个基站设置根据时段开关自动载频的程序．这样就需要知道一段时间内基站流量关于时段的变化值，特别是基站在每个小时的上下流量值．从而可以知道基站在每个时段需要的载频的数量，进而设置一定时间内基站载频自动开关的程序．

另外，无论基站流量随时段怎么波动，从长期来看，大部分基站的整体流量是呈逐渐增加趋势的．当整体流量增加到一定程序时，这种动态开关载频的方式已经无法满足基站在流量峰值时候的需求了，因此需要做物理扩容，新建扇区或者新建基站．由于规划的需要的时间非常长，所以需要提早预估出基站需要物理扩容的时间，从而可以更早的进行规划和设计．

1.2 问题的提出

基于针对初赛问题所作的分析和处理，再根据附件所给的数据，通过建立特征分析与数学建模帮助移动通信运营商解决下面的问题：

问题一：对附件 1 (见初赛题目) 的数据进行预处理，提取附件 1 关于上行和下行流量时间序列的特征，依据你们所提取的特征对附件 1 所给的小区进行分类 (类别不

超过 30 个)，并说明每一类的特点．

问题二：基站运行具有潮汐现象，如果按高容量时段的载频数量来运行基站的配置会造成浪费．根据流量变化设置自动开关限制部分载频的方法，可以有效优化基站的运行，节约能源．基站开关载频的流量阈值则成为研究的关键，过低会造成用户使用的体验感差，过高会造成资源的浪费．请给出你们阈值的设置策略和具体结果．

二、问题的分析

2.1 问题一的分析

问题一主要需要我们提取附件 1 中小区上行和下行流量的时间序列数据的特征，对小区进行分类．在小区的分类中并不依赖于其本身的特征属性，而是主要参考覆盖范围内的主导场景．不同基站由于其覆盖范围内用户、使用业务的差异性，其流量模式也存在极大的差异．比如，在工作区域，小区流量高峰一般出现在早上 7-11 点和下午 13-17 点处于上班时间；在餐饮区域，基站流量高峰一般出现在中午和晚上吃饭时间．但是，由于小区覆盖范围内场景的混合性以及不同场景的相似性，这种划分并不具备代表性．

对于第一问，我们提出了一种基于时间序列特征的聚类方式，从时间序列的分布、熵、稳定性、尺度变化提取时间序列的特征形成时间序列特征向量．

为了对小区进行更细一步的划分，我们先基于能量消耗模型对小区进行第一层聚类，选取“小区日均上行业务量 GB”、“小区上行业务量大于 1GB 的天数”、“小区上行业务量小于 0.2GB”、“小区日均值下行业务量”、“小区下行业务量大于 1GB 的天数”、“小区下行业务量小于 0.2GB 的天数”作为一级聚类特征．根据流量使用量，利用聚类算法，将小区流量使用划分活跃类、较活跃类、较平淡类、平淡类．基于第一层聚类后的小区，我们再使用时间序列特征的聚类方式对小区进行第二层聚类，选取“上行昼夜差均值”、“上行周差均值”、“上行近似熵”、“上行样本熵”、“上行白天消耗流量序列方差”、“下行夜晚消耗 1 流量序列方差”、“下行昼夜差均值”、“下行周差均值”、“下行近似熵”、“下行样本熵”、“下行白天消耗流量序列方差”、“下行夜晚消耗流量序列方差”作为二级聚类特征向量，利用聚类算法，对小区进行更细一步的划分．并将每一类的小区绘制出一周的流量趋势图，说明该类小区的流量趋势特点．

关于问题一的具体思路图如下：

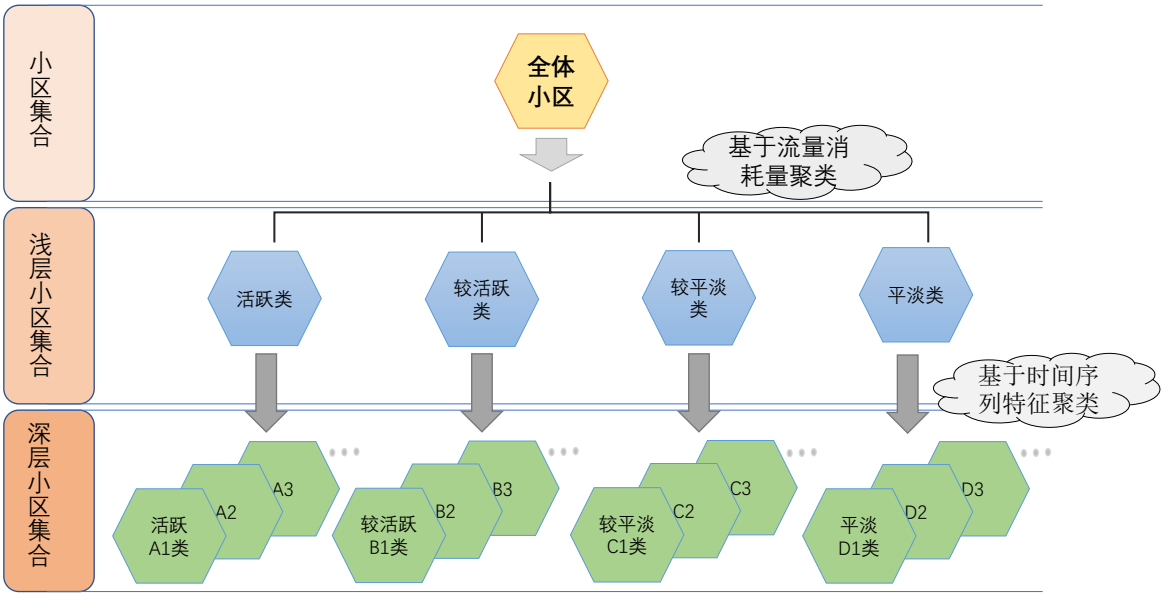

图 1 问题一的思路图

2.2 问题二的分析

问题二主要需要我们对基站阈值的设置进行研究，基站的能源消耗主要来着小区的流量使用．而流量本身存在“潮汐现象”，如果基站以高容量的载频来运行，则对基站能耗是一个很大问题；如果基站关闭一些载频数来降低能耗，在一定程度上，使得用户的服务质量下降．基站智能开关载频算法是我们解决问题的关键，寻找既能使用户达到满足状态，确保用户的服务质量，又能使基站开启较低的载频数，使基站的能耗消耗较低．

通过蜂窝网络结构模型，建立小区覆盖下的用户与基站的联系．基站能耗从基站业务负载和基站静态能耗上考虑，基站的业务负载为该基站下所有小区用户所需要满足的流量总和，基站静态能耗是指基站没有传输数据时，维持基站运行的最小功耗．小区的流量使用一般会有时间性，不同的时间下用户使用流量的需求不同．对此，我们确立了基站的动态分配载频机制．如果某一时刻小区的业务量高于基站分配小区的载频容量，那么说明基站需要分配格外的载频以满足需求；如果小区的业务量等于基站分配小区的载频容量，那么说明基站与小区业务量之间达到了平衡，无需额外的操作；如果小区的业务量小于基站分配的载频容量，说明基站可以关闭一部分的载频数．一方面除了考虑基站能耗，另一方面还要考虑用户的服务质量，指需要为用户提供更好的服务能力．

基于基站能耗和用户服务这两个指标，对蜂窝网络整体综合评价，采用线性加权法，作为目标函数．基于需求，改变权重系数来改变模型的适应性．基站根据小区业务量的变化，分配小区最合适的载频数，使得既能满足用户的流量需求，也能使基站能耗达到最低，使整个蜂窝网络能耗达到最低．

三、模型的假设

1、假设所调查的有关基站的数据都比较准确;

2、假设基站使用宏基站能耗模型;

3、假设小区所属基站不会因为人工或自然原因突然损坏;

4、假设附件中所给小区覆盖地区的流量使用不考虑新建基站的影响;

5、假设网络模型为蜂窝网络模型.

四、符号说明

符号	符号说明
F_n	表示第 n 个小区关于一级聚类的特征向量
G_n	表示第 n 个小区关于二级聚类的特征向量
BS_m	表示第 m 个基站
UE_n	表示第 n 个小区覆盖下用户
$P_m^{BS}(t)$	第 m 个基站在 t 时刻的发射功率
$T_m^{BS}(t)$	第 m 个基站在 t 时刻的业务负载
$P_{total}^m(t)$	第 m 个基站在 t 时刻的总功耗
f_{mn}	基站 m 分配给小区 n 的载频数
C_{mn}^f	基站 m 分配给小区 n 的载频容量
$d_n(t)$	小区 n 在时间点 t 的业务量
ζ	基站需要动态调整的载频数

五、基于二级聚类模型对小区的分类

5.1 相关名词解释

基站：公用移动通信基站是无线电台站的一种形式，是指在一定的无线电覆盖区中，通过移动通信交换中心，与移动电话终端之间进行信息传递的无线电收发信电台.

扇区：扇区是指覆盖一定地理区域的无线覆盖区，是对无线覆盖区域的划分.

载频：一个物理概念，是一个特定频率的无线电波，单位 Hz，是一种在频率、幅度或相位方面被调制以传输语言、音频、图象或其它信号的电磁波.

小区：为用户提供无线通信业务的一片区域，是无线网络的基本组成单位.

SINR：信号与干燥加噪声比 (Signal to Interference plus Noise Ratio) 是指接收到的有用信号的强度与接收到的干扰信号 (噪声和干扰) 的强度的比值.

QoS：服务质量 (Quality of Service) 指一个网络能够利用各种基础技术，为指定的网络通信提供更好的服务能力，是网络的一种安全机制，是用来解决网络延迟和阻塞等问题的一种技术.

蜂窝网络：蜂窝网络 (Cellular network)，又称移动网络 (mobile network) 是一种移动通信硬件架构，分为模拟蜂窝网络和数字蜂窝网络．由于构成网络覆盖的各通信基地台的信号覆盖呈六边形，从而使整个网络像一个蜂窝而得名．目前世界上的主流蜂窝网络类型有：GSM、WCDMA/CDMA2000(3G)、LTE/LTE-A(4G) 等.

频率复用因子: 频率复用因子表示一个频率复用簇 (Reuse Cluster) 当中的频点的数量，复用因子越大，表示复用距离越大.

5.2 基于流量消耗特征对小区的一级聚类

基于 python 中 pandas 库对附件一中的数据进行读取，采取初赛中相同的数据预处理操作，在初赛中，我们主要从以下四个方面对数据进行清洗：

1、对“上行业务量 GB”、“下行业务量 GB”中缺失值进行处理.

2、对其中错误的“日期”数据进行修改，例如，有些日期数据为“018-04-01”，这很明显是记录错误，应该是缺失了数字 2，为了统一，将此类日期错误修改为“2018/4/1”.

3、对“上行业务量 GB”、“下行业务量 GB”中异常值进行处理.

4、对每个小区数据中的重复值进行处理.

数据清洗后，我们先采用流量消耗模型对小区进行一级聚类，这个主要为了区分小区的流量使用量．选取 6 个有关小区流量使用量的特征，作为小区一级聚类的特征向量，即 $F_n = \{f_1, f_2, \cdots, f_6\}$，其中 $f_1, \cdots, f_6$ 具体含义如下表 1所示：

表 1 关于小区流量消耗的特征向量含义表

特征	含义
f_1	小区日均上行业务量 GB
f_2	小区上行业务量大于 1GB 的天数
f_3	小区上行业务量小于 0.2GB 的天数
f_4	小区日均下行业务量 GB
f_5	小区下行业务量大于 1GB 的天数
f_6	小区下行业务量小于 0.2GB 的天数

利用 python 编写代码，在清洗好的附件一中提取出这 6 个特征的数据，提取后的数据如下表 2所示：

表 2 各小区关于流量消耗特征的数据

小区编号	f_1	f_2	f_3	f_4	f_5	f_6
1	7.988	14	11	67.841	28	1
2	2.097	6	0	17.268	7	0
3	6.582	13	12	41.422	39	3
4	3.489	11	10	20.165	40	2
5	5.975	13	17	38.532	38	4
6	2.129	16	9	12.296	39	1
...	...	...	...	...	...	...
127192	1.789	17	0	13.276	21	0
...	...	...	...	...	...	...

将根据特征 F_n 提取好的数据，利用 Mini Batch K-Means 算法[6] 进行聚类．Mini Batch K-Means 是 K-Means 算法的一种优化方案，K-Means 是采用距离为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大．但是当数据量很大时，K-Means 是比较弱的，会比较耗费内存速度．而 Mini Batch K-Means 算法是 K-Means 算法的变种，采用小批量的数据子集减小计算时间，同时仍试图优化目标函数，这里所谓的小批量是指每次训练算法时所随机抽取的数据子集，采用这些随机产生的子集进行训练算法，大大减小了计算时间，与其他算法相比，减少了 K-Means 的收敛时间，小批量 K-Means 产生的结果，一般只略差于标准算法．其 Mini Batch K-Means 的算法如下：

算法：Mini Batch K-Means

输入：k, 小批量子集 b, 迭代次数 t, 数据集 X, 集群集合 C , 数据集合 X
初始化：随机抽取集群点 $\mathbf{c} \in C$, 数据 $\mathbf{x} \in X$
$\mathbf{v} \leftarrow 0$
for $i = 1$ to t **do**
  $M \leftarrow b$ //从 X 中随机选取的示例 b
  for $\mathbf{x} \in M$ **do**
    $\mathrm{d}[x] \leftarrow f(C, \mathbf{x})$ //缓存最接近 x 的中心
  end for
  for $\mathbf{x} \in M$ **do**
    $\mathbf{c} \leftarrow \mathbf{d}[x]$ //获取此 x 的缓存中心
    $\mathbf{v}[\mathbf{c}] \leftarrow \mathbf{v}[\mathbf{c}] + 1$ //更新每个中心的计数
    $\eta \rightarrow \frac{1}{\mathbf{v}[\mathbf{c}]}$ //获取每个中心的学习率
    $\mathbf{c} \leftarrow (1 - \eta)\mathbf{c} + \eta\mathbf{x}$ //采取梯度下降

end for

end for

由于在使用 Mini Batch K-Means 需要确保特征度量的比例尺度一致，所以采用最小最大化处理，将原始数据转化到 [0,1] 的范围，归一化公式如下：

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

其中，$x_{\max}$ 为样本数据最大值，$x_{\min}$ 为样本数据最小值.

基于上述分析，先调用 sklearn 库中 preprocessing 模块中的 *MinMaxScalar()* 函数对特征向量 F_n 中的数据进行归一化处理. 接着，对归一化后的小区特征数据进行 Mini batch K-Means 聚类，调用 sklearn.cluster 库中 *MiniBatchKMeans* 算法，构造聚类器，进行聚类，根据群惯性和轮廓系数来确定具体的 k 值就可以进行叶子分裂优化计算. 群惯性是一种最小化群内误差平方和 (SSE) 的迭代方法，其公式为:

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)} = ||x^i - \mu^j||_2^2, \tag{2}$$

其中 μ^j 为集群 j 的代表点 (重心)，如果样本 x^i 在集群 j 中, $w^{(i,j)} = 0$, 否则 $w^{(i,j)} = 1$. 而轮廓系数 (silhouette Coefficient) 是聚类效果好坏的一种评价方式. 对于其中的一个点来说，i 向量的轮廓系数为:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{3}$$

从上式可见，轮廓系数是介于 [-1,1]，越趋近于 1 代表内聚度和分离度都相对较优. 其中 $a(i)$ 表示 i 向量到同一簇内其他点不相识程度的平均值，$b(i)$ 表示 i 向量到其他簇内的平均不相识程度的最小值. 将所有点的轮廓系数求平均，就是该聚类结果总的轮廓系数.

利用 python 调用 Mini Batch K-means 算法和 sklearn.metrics 中 sihoutte_score 模块绘制出群惯性图 (SSE) 和轮廓系数图 (silhouette Coefficient)，如下图:

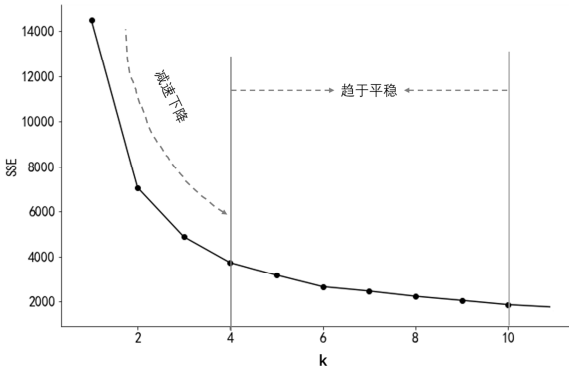

图 2 一级聚类群惯性 (SSE) 效果图

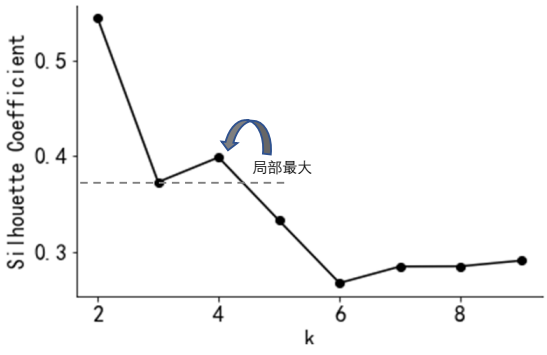

图 3 一级聚类轮廓系数效果图

从图 2中得出从 $k = 4$ 开始左右，曲线变化趋势较为平缓，结合一级小区轮廓系数图 3，综合考虑取 k=4.

最终聚类结果如下表：

表 3 一级聚类小区分类表

聚类后小区类别	小区数量
第一类小区	48229
第二类小区	67589
第三类小区	9071
第四类小区	8306

利用 python 编写代码，分别选取关于上行业务量的三个特征和下行业务量的三个特征，做出聚类效果图，其中图 4 是关于上行业务量三个特征的聚类效果图，图 5 是关于下行业务量三个特征的聚类效果图.

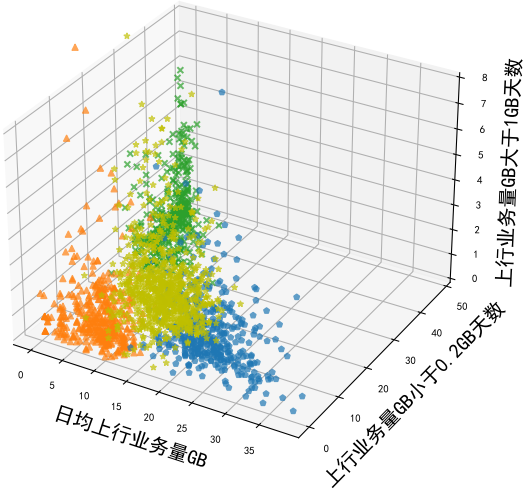

图 4 关于上行特征的三维效果图

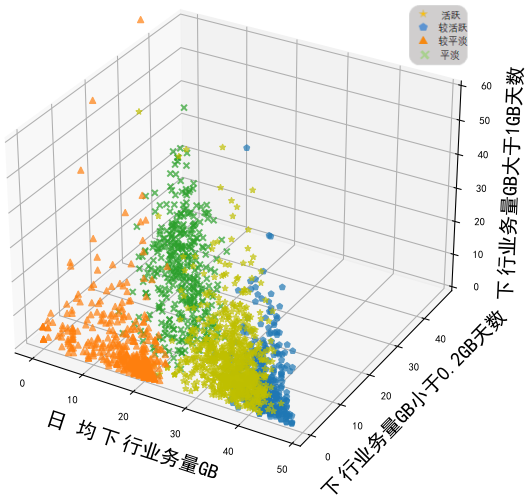

图 5 关于下行特征的三维效果图

从上图4、5中可以看出，活跃类和较活跃类小区主要集中在“上行业务量 GB”、“上行业务量 GB 大于 1GB 天数”较大的数值区间，“下行业务量 GB 小于 0.2GB 天数”较小的数值区间. 而较平淡类和平淡类小区则相反.

为了进行更细一步的说明，我们对一级聚类的小区特征做出了雷达图，如图 6所示：

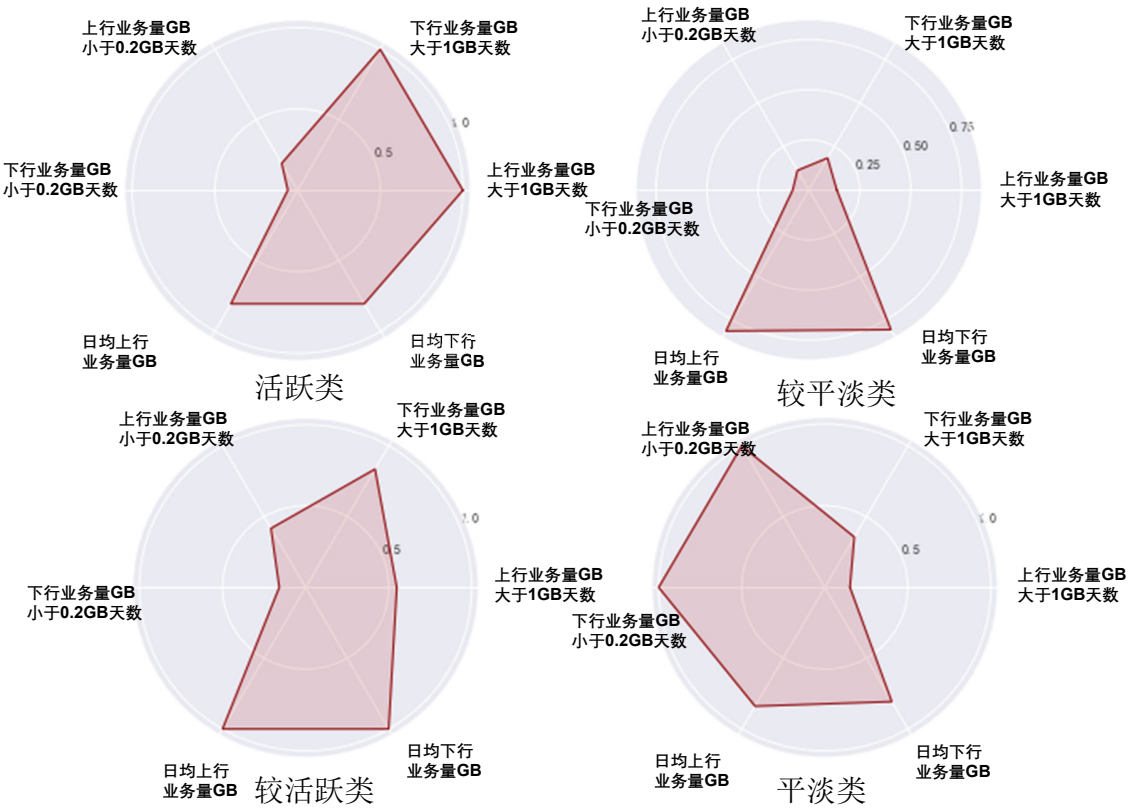

图 6 一级聚类小区特征雷达图

在上图中，我们基于流量消耗的 6 个特征将小区分为活跃类、较活跃类、较平淡类、平淡类．从雷达图，可以明显的看出活跃类和较活跃类小区的“日均上行业务量 GB”、“日均下行量 GB”、“下行业务量大于 1GB 天数”特征比较明显；相较于平淡类小区“下行业务量 GB 小于 0.2GB 的天数”、“上行业务量 GB 小于 0.2GB 的天数”特征比较明显．

5.3 基于时间序列特征对小区的二级聚类

为了更进一步对小区分类进行研究，我们采用基于时间序列特征聚类模型[3]，对分类后的小区再次进行二级聚类．时间序列聚类是一种完全根据数据自身所提供的信息进行分类的一种方法．不同小区相当于不同类别的场景，场景之间的差异性主要包括昼夜差、周差、熵、方差等特征，基于这些特征要求对时间序列进行处理．我们选择 12 个能反映小区特性的特征构成特征向量，作为小区二级聚类的特征向量，即 $G_n = \{g_1, g_2, g_3, \cdots, g_{12}\}$，关于 $g_1, g_2, \cdots, g_{12}$ 的具体含义如下表 4所示:

表 4 关于小区流量时间序列特征的特征向量含义表

特征	含义	特征	含义
g_1	上行昼夜差均值	g_7	下行昼夜差均值
g_2	上行周差均值	g_8	下行周差均值
g_3	上行近似熵	g_9	下行近似熵
g_4	上行样本熵	g_{10}	下行样本熵
g_5	上行白天消耗流量序列方差	g_{11}	下行白天消耗流量序列方差
g_6	下行夜晚消耗流量序列方差	g_{12}	下行夜晚消耗流量序列方差

其中，近似熵是一种用于量化时间序列波动的规律性和不可预测性的非线性动力学参数，用一个非负数来表示一个时间序列的复杂性，反映了时间序列中新信息发生的可能性，越复杂的时间序列对应的近似熵越大．而样本熵是基于近似熵的一种用于度量时间序列复杂性的改进方法．

在上表中，昼夜差[3] 表示一天最大最小的落差占最大值的百分比，即

$$DI = \frac{1}{D}\frac{v_{\max}(d) - v_{\min}(d)}{v_{\max(D)}}, \tag{4}$$

在上式中，DI 表示昼夜差，D 表示所有天数，$v_{\max(d)}$ 和 $v_{\min(d)}$ 分别表示第 d 天的最大最小数据量．由于最小值一般出现在夜间，最大值出现在白天，所以可以使用这样的方式来表示流量昼夜差距．

周差[3] 表示的是工作日与周末总值之间的落差占工作日总值的百分比，即

$$WI = \frac{\frac{1}{D}\sum_{d=1}^{D} v_{sum}(d) - \frac{1}{W}\sum_{w=1}^{W} v_{sum}(w)}{\frac{1}{D}\sum_{d=1}^{D} v_{sum}(d)} \tag{5}$$

其中，WI 表示周差，w 表示所有周数，$v_{mum}(d)$ 表示第 d 天的所有流量之和，$v_{mum}(w)$ 表示第 w 个周末的流量总数．

利用 python，在附件一中提取出这些特征，提取后的部分数据如下表 5所示：

表 5 各小区关于时间序列特征的数据

小区编号	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
1	0.71	-1.9e-4	0.03	0.04	0.58	0.41	6.27	-3e-3	3.99	4.59	0.58	0.98
3	0.53	9.7e-5	0.08	0.03	0.21	0.19	3.39	5.3e-4	4.94	1.69	0.80	1.35
9	0.17	3.1e-4	5e-3	2e-3	0.15	0.14	1.62	2e-3	0.68	0.09	0.61	0.63
10	0.10	3.2e-4	2e-3	9e-4	1e-7	0.04	0.65	2e-3	0.10	0.01	0.61	0.60
18	0.04	2e-3	1e-4	5e-4	1e-8	0.10	0.22	3e-3	7e-3	4e-3	0.45	0.65
…	…	…	…	…	…	…	…	…	…	…	…	…
172	0.02	-3.5e-4	1.5e-5	9e-5	0.15	0.19	0.18	-0.03	0.012	0.002	0.63	1.64
…	…	…	…	…	…	…	…	…	…	…	…	…

在 5.2 中，我们已经将小区根据流量消耗特征分为了 4 类，即活跃类、较活跃类、较平淡类、平淡类．基于这四类，再对每一类进行时间序列特征聚类，对每一类进行划分．先绘制出各类小区的群惯性系数和轮廓系数图，如下：

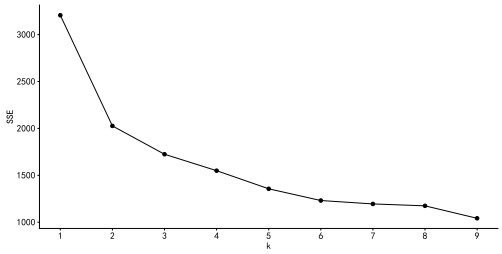

图 7 活跃类小区群惯性效果图

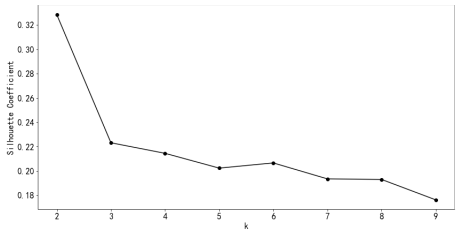

图 8 活跃类小区轮廓系数效果图

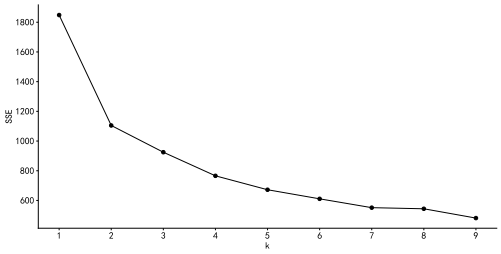

图 9 较活跃类小区群惯性效果图

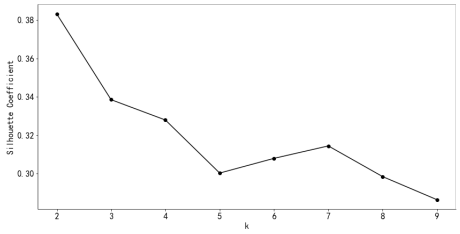

图 10 较活跃类小区轮廓系数效果图

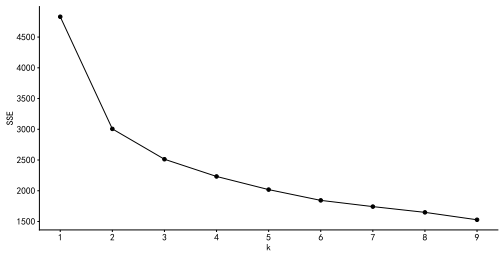

图 11 较平淡类小区群惯性效果图

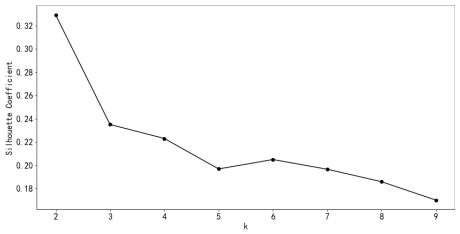

图 12 较平淡类小区轮廓系数效果图

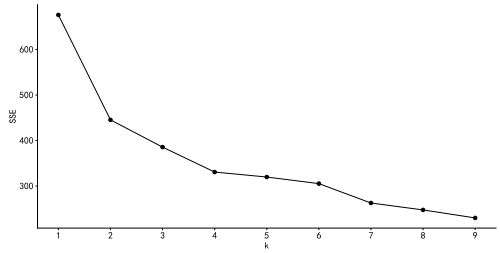

图 13 平淡类小区群惯性效果图

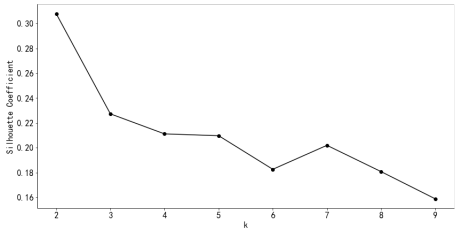

图 14 平淡类小区轮廓系数效果图

观察上述各类小区的群惯性效果图和轮廓系数效果图的趋势，我们确定各类二级聚类的 k 值如下表：

表 6 各类小区二级聚类的 k 值

小区类别	k 值
活跃类	3
较活跃类	3
较平淡类	6
平淡类	4

基于每个小区类别得到的 k 值，利用 Mini Batch K-Means 聚类，最终得到 16 个小区类别的聚类中心，如下表：

表 7 各类小区的聚类中心

聚类	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}
第一类	0.1125	0.9990	0.0155	0.0135	0.0774	0.9639	0.0073	0.0062	0.5406	0.6237	0.4238	0.3770
第二类	0.0846	0.9988	0.0091	0.0081	0.0573	0.9637	0.0037	0.0031	0.1608	0.6915	0.1195	0.3074
第三类	0.0874	0.9989	0.0094	0.0086	0.0595	0.9638	0.0040	0.0032	0.3534	0.6732	0.2565	0.3404
第四类	0.0751	0.9983	0.0089	0.0055	0.0641	0.8991	0.0042	0.0041	0.5412	0.5241	0.4192	0.3742
第五类	0.0634	0.9983	0.0061	0.0043	0.0560	0.8991	0.0033	0.0033	0.1273	0.6219	0.0963	0.3198
第六类	0.0425	0.9982	0.0032	0.0021	0.0375	0.8991	0.0015	0.0015	0.2773	0.4609	0.1928	0.2624
第七类	0.1373	0.9983	0.0189	0.0121	0.1186	0.8992	0.0100	0.0099	0.3262	0.5569	0.2375	0.3457
第八类	0.0483	0.9983	0.0035	0.0025	0.0420	0.8991	0.0018	0.0015	0.3374	0.6602	0.2483	0.3900
第九类	0.2911	0.9983	0.0953	0.0593	0.2594	0.8993	0.0635	0.0633	0.3249	0.5573	0.2399	0.3309
第十类	0.0691	0.9977	0.0106	0.0091	0.0611	0.9950	0.0079	0.0056	0.0347	0.1313	0.1389	0.0354
第十一类	0.0555	0.9990	0.0111	0.0105	0.0449	0.9963	0.0054	0.0053	0.1470	0.5938	0.0907	0.3842
第十二类	0.0805	0.9990	0.0132	0.0127	0.0715	0.9963	0.0138	0.0159	0.5169	0.3974	0.2335	0.2988
第十三类	0.1016	0.9233	0.0101	0.0139	0.0679	0.7976	0.0092	0.0092	0.3756	0.6731	0.2391	0.3590
第十四类	0.1064	0.9240	0.0135	0.0174	0.0697	0.7979	0.0103	0.0112	0.5815	0.5897	0.3867	0.3790
第十五类	0.0607	0.9216	0.0057	0.0109	0.0435	0.7983	0.0059	0.0061	0.2616	0.5166	0.1700	0.2188
第十六类	0.1188	0.9233	0.0142	0.0179	0.0785	0.7976	0.0124	0.0124	0.1539	0.7074	0.1106	0.3212

从上表可以看出不同类别之间的小区，各特征的聚类中心差距比较大，说明了这些不同类别的小区之间存在较大的差异.

5.4 各类别小区的特征描述

基于二级聚类模型，我们将小区划分为 16 类，并提取了每一类中的一个小区，绘制出一星期的流量使用趋势图. 下面对每一类小区的使用流量进行分析和特点描述：

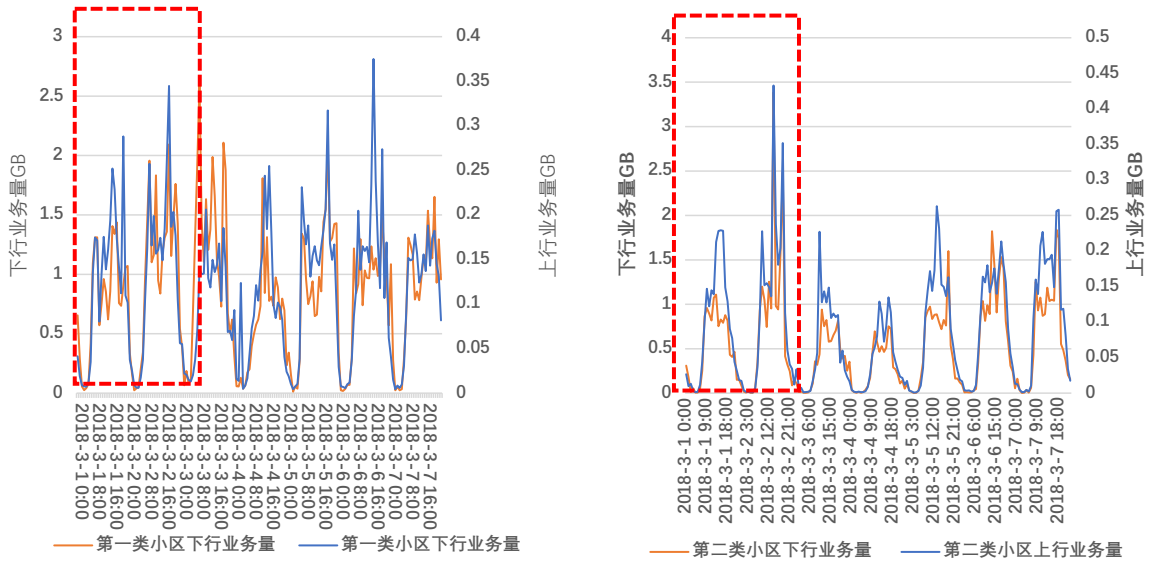

图 15 第 1-2 类小区使用流量趋势图

从图 15中可以看出，第一类小区的下行业务量 GB 较大，大部分时间段都超过了 1GB 以上，应该属于活跃类小区中的一种．从时间序列上看，第一类小区的流量使用主要集中在一天的 8-21 点，中午时间段会有较小的波动，晚上 21 点以后基本趋近于 0，比较符合工作区、上班族这一类的趋势．第二类小区的下行业务量 GB 也较大，大部分时间段也超过了 1GB 以上，也属于活跃类小区的一种．流量使用主要集中在 9-18 点, 中午时间段使用流量比较多，午夜流量使用几乎为 0，而且有些时候每天流量使用会相差较大．

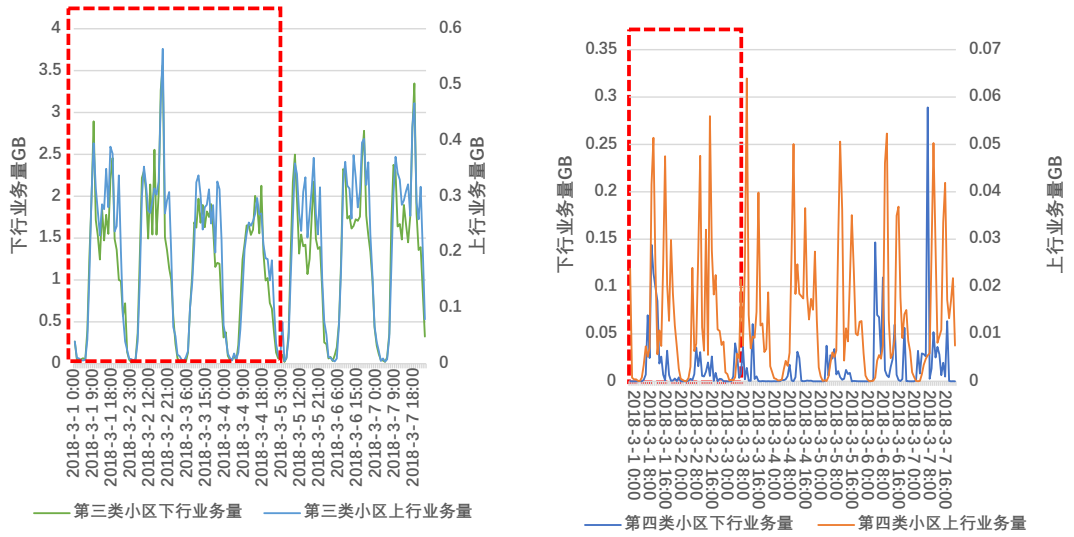

图 16 第 3-4 类小区使用流量趋势图

从图 16中可以看出，第三类小区的流量使用较大，属于活跃类中的一种．从时间序列上看，流量使用先从 8-11 点上升，11-13 点稍微下降，从 13-17 点又上升，18 点之后下降趋近于零，且每天变化趋势不大，较为稳定，应该属于某些企业上班区域．第

四类小区相对来说流量使用较小，日流量使用集中在 8-16 点，且数值不高，说明该地区，使用流量需求比较小，而且下行流量使用波动比较大，下行流量周差异较大，而上行业务量周差异较为稳定.

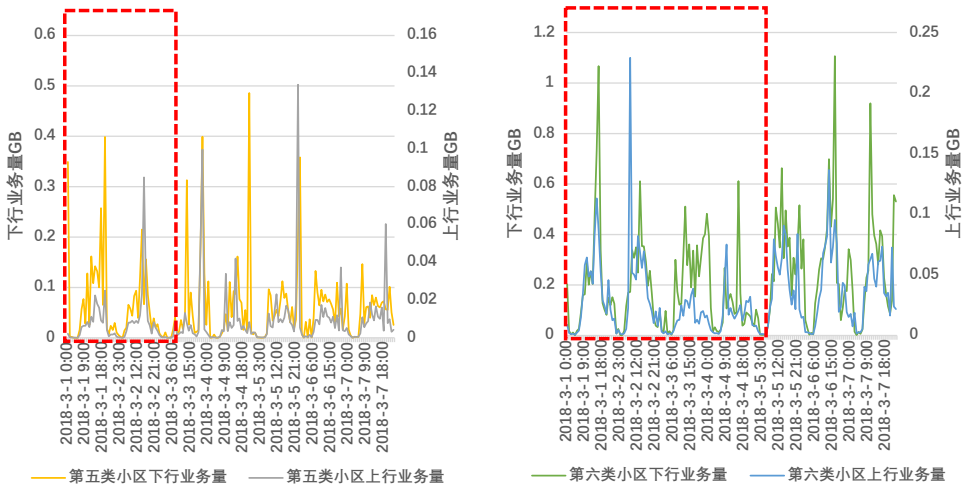

图 17 第 5-6 类小区使用流量趋势图

从图 17中观察第四、五类小区使用流量趋势．第四类小区流量使用量不大，流量使用集中在 9-18 点之间，凌晨会有较小的波动．这一类小区有一点很明显的特点，前五天的业务量大于后两天的业务量，存在着周差．第六类小区流量使用相对较小，流量峰值集中在 12 点左右，流量使用主要集中在 9-18 点．从流量趋势可以看出，有两天的流量使用明显下降，说明存在周差.

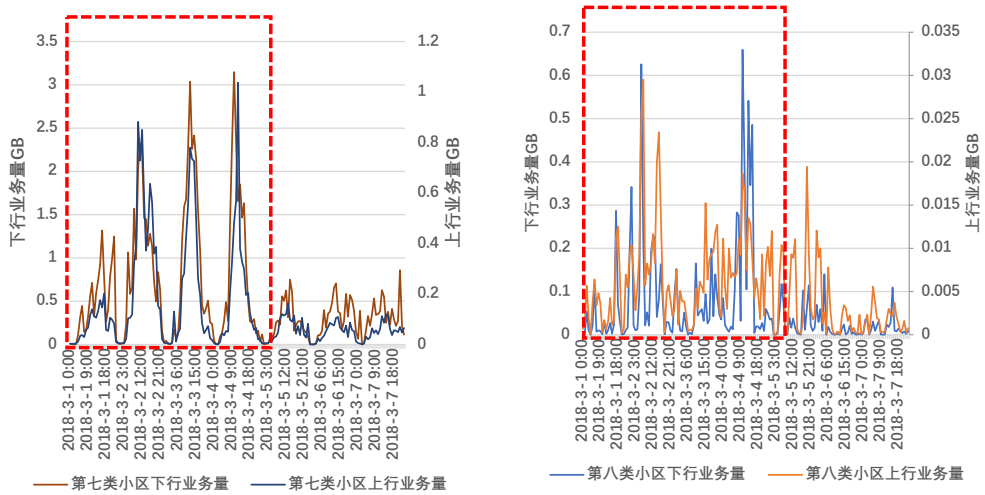

图 18 第 7-8 类小区使用流量趋势图

从图 18中观察第七、八类小区使用流量趋势．第七类小区流量小区使用量较大，但是波动性较大，很明显的可以看出，该小区每日的需求量不同，其中 3 月 2 日至 3 月 4 日的需求明显高于后三天的流量需求，存在很明显的周差，日均值差距也较大．在使用较多的天数，每日的流量需求集中在 8–15 点，在 12 点左右达到峰值；在需求量

不是很大的天数，流量没有明显的趋势．第八类小区波动性比较大，没有较明显的流量趋势，从时间序列上看，每个时间段没有明显的特点，比较杂乱，每日之间流量使用也有差距，例如 3 月 7 日的流量使用趋势曲线明显低于前几天的趋势．

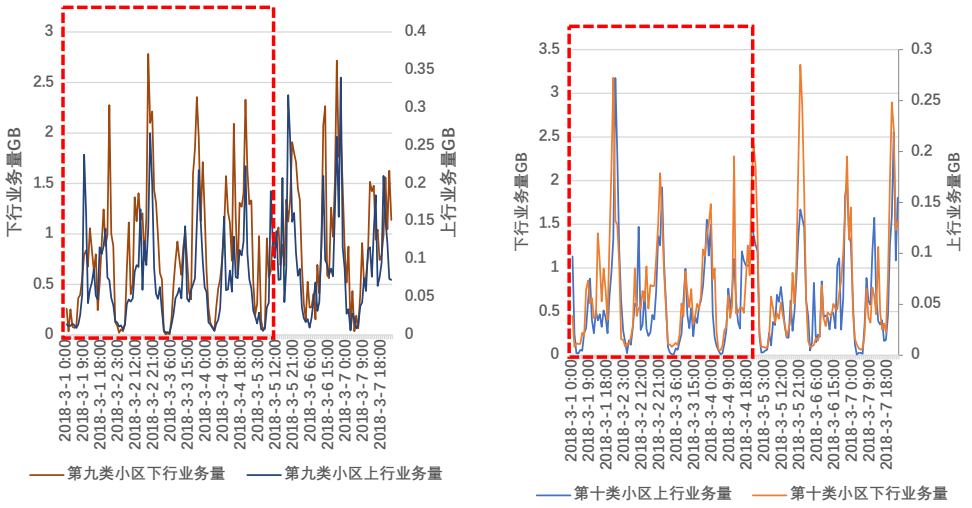

图 19 第 9-10 类小区使用流量趋势图

从图 19中可以看出，第九类小区流量使用相对较多，大部分时间端达到 1GB 以上，从时间序列上看，流量主要集中在 9-22 点，其中 9-11 点和 18-21 点呈上升趋势．每日的流量使用量也基本较稳定，没有较大的上升下降趋势，第九类小区应该属于商场、超市区域．第十类小区流量相对比较活跃，每日流量趋势 9-18 点相对来说使用较小，但在 18 点之后流量使用急速上升，在 22 点左右达到峰值，流量使用量主要集中在后半段时间．每日流量使用量相对比较稳定，波动幅度不是很大．该类小区应该有较多的娱乐设施，适合年轻人夜晚娱乐、释放压力．

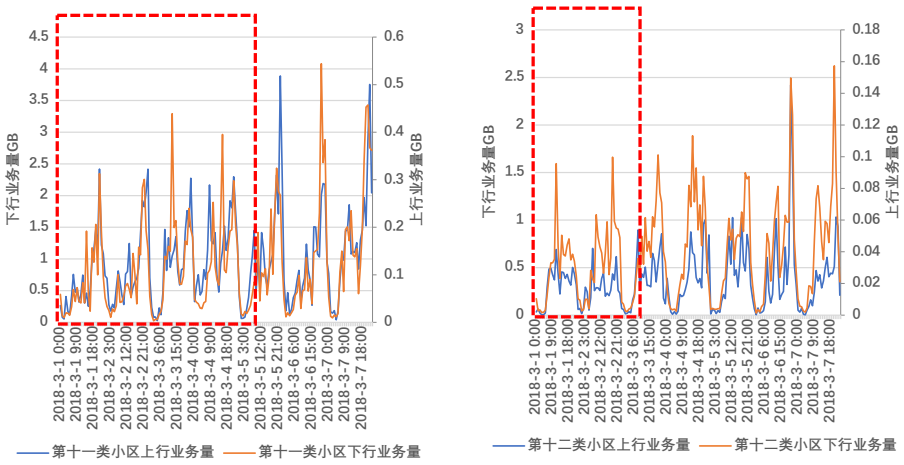

图 20 第 11-12 类小区使用流量趋势图

从图 20中观察第十一、十二类小区使用流量趋势．第十一类小区的流量相对较大，结合时间序列，一天内的流量消耗主要集中在 16-22 点，9-16 点，流量消耗较小，超

过 1.5G 的流量主要集中在 22 点左右．每日的流量消耗存在较小的波动，3 月 7 号的流量消耗明显高于 3 月 1 号的流量消耗，存在周差现象．第十二类小区流量使用集中在 9-20 点时间段，在这个时间段内存在较小的波动，一般 12 点和 18 点流量使用达到峰值．从 20 点左右开始下降趋势．从周差上分析，前五天的流量消耗相对比较稳定，后两天的流量消耗明显比之前大．

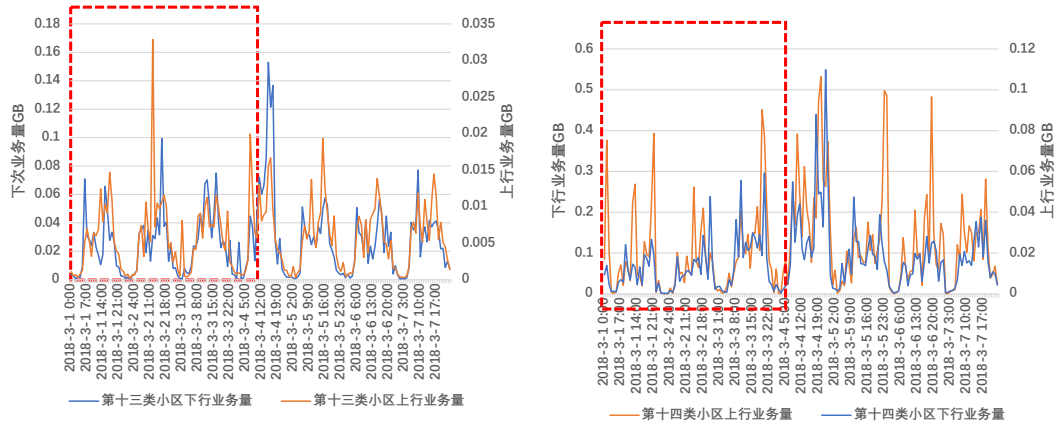

图 21　第 13-14 类小区使用流量趋势图

从图 21中观察第十三、十四类小区使用流量趋势．第十三类小区流量消耗相对较小, 流量使用集中在 7-21 点，中午时间段会体现下降趋势，22 点后流量使用接近于 0．每天流量使用量比较稳定．第十四类小区流量使用较小，说明小区用户流量使用量需求不是很大，流量使用主要集中在 7-21 点，中午区间有下降趋势，波动性较大．每日的流量使用量也存在波动性，有些天数明显比较大．

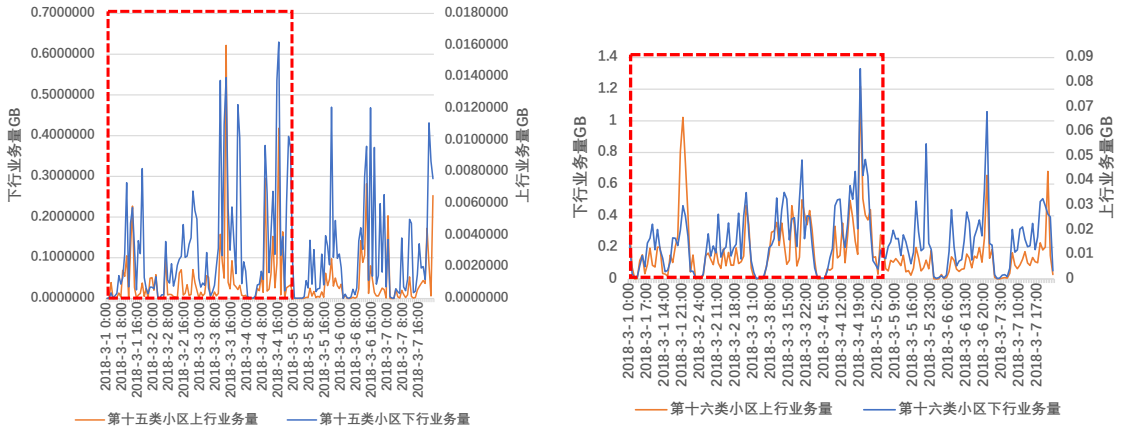

图 22　第 15-16 类小区使用流量趋势图

从图 22中观察第十五、十六类小区使用流量趋势．第十五类小区使用量比较少，流量需求较小，流量使用主要集中在白天时段，且波动性较大，夜晚流量消耗接近于零，但也存在波动性，没有较明显的规律．第十六类小区流量使用集中在 7-14 点和 15–22 点，在 20 点左右存在峰值，22 点后流量使用接近于 0．每日流量使用有相似的趋势，没有较大的变化．

六、基站智能开关载频节能机制研究

在初赛题目中，我们主要对各小区长短期上下行流量进行了预测，而移动流量的使用和通信离不开基站，随着无线通行业务的持续增长，基站的数目也在不断增加．在移动通信网络中，基站是网络能源消耗的主要因素，降低基站的能耗是提升整个网络能效水平的关键方向．无线通信网络中基站的配置是依据最大负载情况来确定的，但无线网络业务量会随着时间不断变化，存在“潮汐”现象，如果在低负载时期，基站按高容量时段的载频数量来运行，会给无线通信网络造成不必要的能源消耗．根据流量业务量的变化来设置自动开关载频的方法，可以有效优化基站的运行，节约能源．在基站的节能技术中，基站自动开关载频技术易于实施，只要改变基站的载频数量符合用户流量需求，无需对硬件进行改变，节能效果明显等优点成为节能研究的重点方向．

基于初赛所建立的对各小区的流量预测算法，得到了未来时段的流量使用情况，为了进一步优化支援分配方式，提升资源利用效率，以降低网络能效为目标，我们对基站智能开关载频的节能机制进行研究．

6.1 蜂窝网络结构模型

蜂窝网络组成主要有以下三部分：移动站、基站子系统、网络子系统．移动站就是我们的网络终端设备，比如手机或者一些蜂窝工控设备．基站子系统包括移动基站、无线收发设备、专用网络、无限的数字设备等．基站子系统可以看作无线网络与有线网络之间的转换器．如下图为一个简单的蜂窝网络结构：

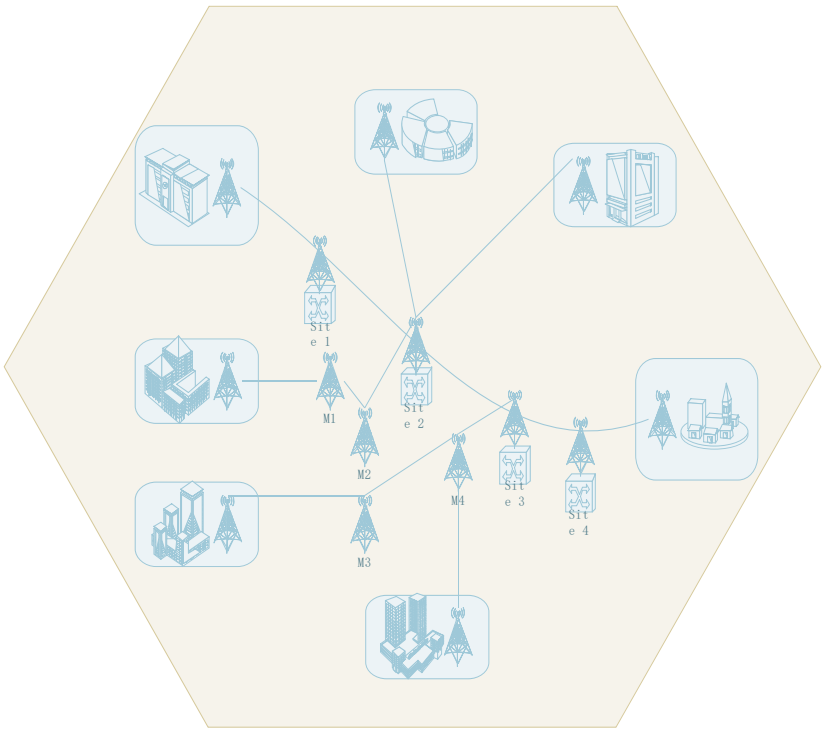

图 23 蜂窝网络结构模型图

6.2 基于线性加权的基站智能开关载频算法

6.2.1 网络和能耗模型中变量的确定

基于简单蜂窝网络，设网络模型中有 M 个基站，将全部的基站集合记为 $BS = \{BS_1 BS_2 \cdots BS_M\}$，第 m 个基站为 BS_m．设小区内的用户 (流量使用者) 为 UE，记第 n 个小区的用户为 UE_n．首先将网络表示成基站、用户以及它们连接关系的二部图，在此基础上对用户服务质量和基站节能建立线性加权的基站智能开关载频算法．假设网络频率复用因子为 1，则在 t 时刻用户 UE_n 接入 BS_m 时，UE_n 的 $SINR$ 值[8] 可表示如下：

$$SINR_{m,n}(t) = \frac{P_m^{BS}(t) \cdot h_m^2}{\sum\limits_{i=1, i\neq m}^{M} P_i^{BS}(t) h_i^2 + P_N}, \tag{6}$$

上式 (6) 中，$P_m^{BS}(t)$ 表示第 m 个基站在时间点 t 时的发射功率，h_m 表示第 m 个基站的信道系数；$\sum\limits_{i=1, i\neq m}^{M} P_i^{BS}(t)$ 表示在时间点 t 时的干扰信号功率，h_j 表示干扰信道系数；P_N 表示随机噪声功率．

设 $d_n(t)$ 表示小区 n 在时间点 t 的业务量大小，用 UE_m^{BS} 表示与 BS_m 通信的 UE．定义在时间点 t 的 BS_m 业务负载为 $T_m^{BS}(t)$, 那么 $T_m^{BS}(t)$ 可表示为 UE_m^{BS} 的业务量总和，即 $T_m^{BS}(t) = \sum\limits_{UE \in UE_m^{BS}} d_n(t)$．因为 $P_m^{BS}(t)$ 即在时间点 t 的 BS_m 发射功率与 BS_m 表示为 $T_m^{BS}(t)$ 的函数[2]，如下：

$$P_m^{BS} = g(T_m^{BS}(t)), \tag{7}$$

其 $g(\cdot)$ 是一个不减函数，其表示基站的业务负载越大，其发射功率也会增大．

本文采用线性基站能耗模型[7]，基站的能耗分为两个部分，即静态和动态部分．静态部分指基站没有传输数据时，维持基站运行的最小功耗；动态部分指当基站负载上升，基站发射功率也会随之提高．基站 BS_m 在时刻 t 的总功耗[7] $P_{total}^m(t)$ 可以表示为：

$$P_{total}^m(t) = \begin{cases} N_{TRX}(P_0 + \Delta P \cdot P_m^{BS}(t)), & 0 < P_m^{BS}(t) < P_{max} \\ N_{TRX} \cdot P_{sleep}, & P_m^{BS}(t) = 0 \end{cases} \tag{8}$$

上式 (8) 中 N_{TRX} 代表基站收发信机个数，P_0 代表基站处于活跃状态，发射功耗接近于零时的最小功耗，P_m^{BS} 是基站 BS_m 的发射功耗，Δp 是与负载相关的斜率系数，P_{sleep} 表示休眠功耗，P_{max} 表示最大负载下的最大 RF 输出功率．通过参考相关文献[7]，关于 Macro BS(宏基站) 和 Pico BS(皮基站) 的基站类型，具体的参数设定如表 8所示：

表 8 基站能耗模型参数

基站类型	N_{TRX}	P_0	Δ_P	P_{sleep}	P_{max}
Macro BS	6	130.0 W	4.7	75 W	20W
Pico BS	2	6.8W	4.0	4.3 W	0.13 W

6.2.2 基站载频分配方案

设 f_{mn} 表示基站 m 分配给小区 n 的载频数，C_{mn}^{f} 表示基站 m 分配给小区 n 的载频容量[5]．由于载频容量与载频数之间存在关联，分配给载频数越多的小区，会有更多的载频容量．故 C_{mn}^{f} 可表示为 f_{mn} 的函数，如下：

$$C_{mn}^{f} = \Psi(f_{mn}), \tag{9}$$

其中，$\Psi(\cdot)$ 是个不减函数，表示所给小区的载频数越多，载频容量越大．

由上述知 $d_n(t)$ 表示小区 n 在时间点 t 的业务量，C_{mn}^{f} 表示基站 m 分配给小区 n 的载频容量．在任意时间点 t 的业务量 $d_n(t)$ 可以根据初赛的预测模型容易得到．我们进行以下讨论：

1、如果 $d_n(t)>\Psi(f_{mn})$，说明小区 n 在时间点 t 的业务量大于基站 m 分配给小区 n 的载频容量，说明基站需要开启分配更多的载频，才能满足小区 n 的需求．

2、如果 $d_n(t+1) = \Psi(f_{mn})$, 说明小区 n 在时间点 t 的业务量与基站 m 分配给小区 n 的载频容量相同，说明基站的载频数能满足需求，不需要额外分配载频．

3、如果 $d_n(t+1) < \Psi(f_{mn})$, 说明小区 n 在时间点 $t+1$ 的业务量比基站 m 分配给小区 n 的载频容量小，说明基站不需要分配这么多的载频给小区，可以关闭一些载频的使用．

基于上面的分析，设 ζ 为基站需要动态调整的载频数，那么 ζ 的数学公式如下：

$$\zeta = \begin{cases} k, & \Psi(f_{mn}+k) > d_n(t) > \Psi(f_{mn}+k-1) \\ 0, & d_n(t) = \Psi(f_{mn}) \\ -k, & \Psi(f_{mn}-k) > d_n(t) > \Psi(f_{mn}-1-k) \end{cases} \tag{10}$$

上式公式表明，基站 BS_m 根据小区第 t 时间点的业务量需求，动态分配载频，使其恰好满足．结合基站载频分配方案，第 m 个基站在时间点 t 的发射功率 $P_m^{BS}(t)$ 转变为 $P_M^{BS}(t) = g(T_m^{BS}(t) + \Psi(\zeta))$.

其基站载频分配算法流程图如下：

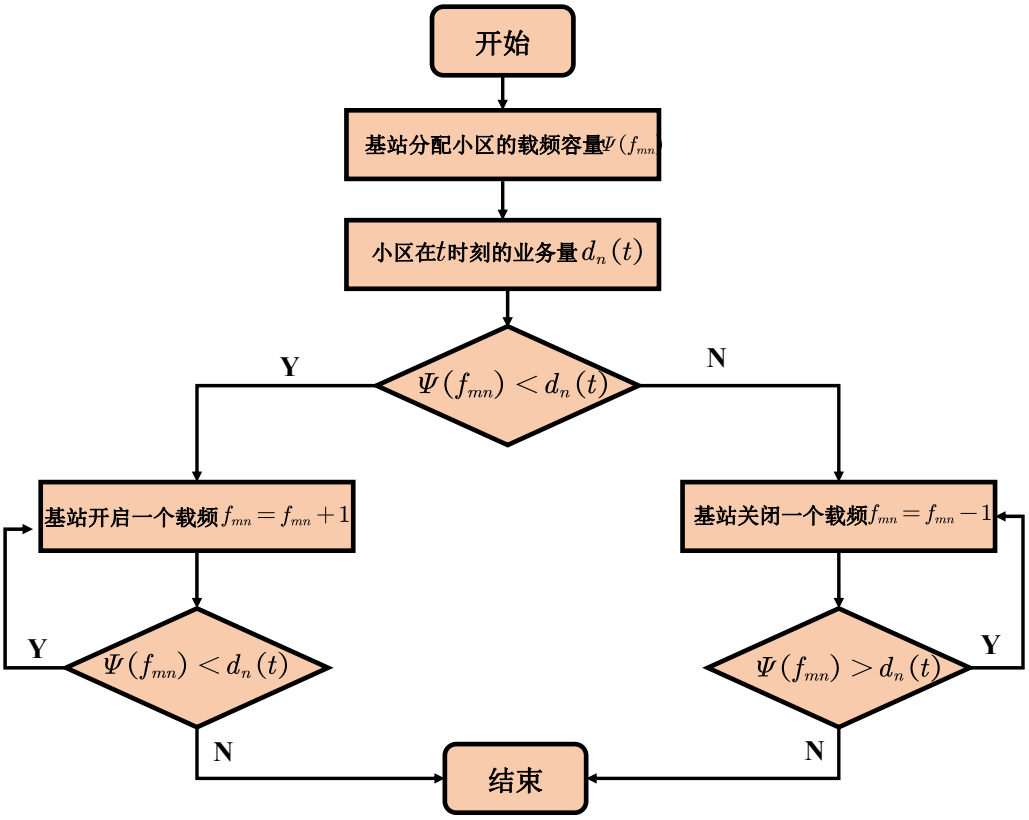

图 24 基站载频分配算法流程图

6.2.3 基站智能开关载频算法的确立

由于基站 BS_m 与小区 UE_n 的连接状态不好确立，我们使用连接矩阵 $X = x_{m,n}$ 来表示基站 BS_m 与 UE_n 的连接关系，即

$$x_{m,n} = \begin{cases} 1, & BS_m与UE_n连接 \\ 0, & BS_m与UE_n未连接 \end{cases} \tag{11}$$

通过 0-1 变量，$T_m^{BS}(t)$ 可以表示为 $T_m^{BS}(t) = \sum_{n=1}^{N} x_{m,n} d_n(t)$.

在蜂窝网络中用 *SINR* 表示用户使用流量的 QoS(服务质量) 值[2]，蜂窝网络的能耗用 BS_m 的业务负载来表示．一般来说，为了提高各小区覆盖下的用户使用需求，基站会往往分配给更多的载频数给小区，来连接以保证用户能有较高的通信质量和使用体验，所以，UE_m 会有比较高的 *SINR* 值．但是，基站总是分配给小区较多的载频，或者以高容量的载频数量来运行基站的配置，这样会导致能耗大幅度提升．相反地，如果基站给小区分配的载频数一直较小，或者以低容量的载频数来运行基站的配置，以减少能耗损失，用户的满足程度下降，*SINR* 值就会较少．所以，在无线通信中要综合考虑通信质量和能耗，以满足达到满足 QoS 情况下节省能耗.

通过上述分析，既要满足基站在时刻 t 的能耗 $P_{total}^{m}(t)$ 尽可能低，又要使用户 QoS 值尽可能高，考虑由于 *SINR* 和基站能耗之间有较强的关联性，适合采用指标间关联性

较强的线性加权和法 (linear weighted sum method)，线性加权法是一种评价函数方法，是按各目标的重要性赋予它相应的权系数，然后对其线性组合进行寻优的求解多目标规划问题的方法.

由于 t 时刻 UE_n 的 $SINR$ 值与 t 时刻基站的能耗 $P^m_{total}(t)$ 有不同的单位和数量级，这两个指标之间没有可比性，或者说指标间存在不可公度性. 所以首先对 $SINR_{m,n}(t)$ 和 $P^m_{total}(t)$ 进行标准化操作，标准化可以解决指标大小排序大小排序产生的不合理性，具体操作如下：

$$\overline{SINR_{m,n}(t)} = \frac{SINR_{m,n}(t) - \min\{SINR_{m,n}(t)\}}{\max\{SINR_{m,n}(t)\} - \min\{SINR_{m,n}(t)\}}, \tag{12}$$

$$\overline{P^m_{total}(t)} = \frac{P^m_{total}(t) - \min\{P^m_{total}(t)\}}{\max\{P^m_{total}(t)\} - \min\{P^m_{total}(t)\}} \tag{13}$$

基于采用线性加权法，综合考虑在 t 时刻的 UE_m 的 $SINR$ 值和 BS_m 的总功耗，当 UE_n 和 BS_m 相连时，则可把这条通路的连通指标设为

$$y_{m,n}(t) = \omega_1\overline{SINR_{m,n}(t)} + \omega_2\overline{P^m_{total}(t)}, \tag{14}$$

其中 ω_1、ω_2 表示权重系数. 同时反映用户服务质量和基站功耗情况，并且可以通过改变两个指标的权重系数改变模型的适用性. 由于 $\overline{SINR_{m,n}(t)}$ 表示 t 时刻用户的服务质量 Qos, 其值越高越好；而 $\overline{P^m_{total}(t)}$ 表示 t 时刻基站的总功耗. 故 ω_2 是负值参数. 因此，蜂窝网络整体评价指标 Y

$$\max \quad Y = \sum_{m=1}^{M}\sum_{n=1}^{N} y_{m,n}(t)\cdot x_{m,n}.$$

$$\text{s.t.}\begin{cases} y_{m,n}(t) = \omega_1\overline{SINR_{m,n}(t)} + \omega_2\overline{P^m_{total}(t)}, \\ P^m_{total}(t) = \begin{cases} N_{TRX}(P_0 + \Delta P\cdot P^{BS}_m(t)) \ , & 0 < P^{BS}_m(t) < P_{max} \\ N_{TRX}\cdot P_{sleep} \ , & P^{BS}_m(t) = 0 \end{cases} \\ SINR_{m,n}(t) = \frac{P^{BS}_m(t)\cdot h^2_m}{\sum\limits_{i=1,i\neq m}^{M} P^{BS}_i(t)h^2_i + P_N}, \\ P^{BS}_M(t) = g(T^{BS}_m(t) + \Psi(\zeta)), \\ T^{BS}_m = \sum\limits_{n=1}^{N} x_{m,n}d_n(t), \\ \zeta = \begin{cases} k, & \Psi(f_{mn}+k) > d_n(t) > \Psi(f_{mn}+k-1) \\ 0, & d_n(t) = \Psi(f_{mn}) \\ -k, & \Psi(f_{mn}-k) > d_n(t) > \Psi(f_{mn}-1-k) \end{cases} \\ x_{m,n} = \begin{cases} 1, & BS_m\text{与}UE_n\text{连接} \\ 0, & BS_m\text{与}UE_n\text{未连接} \end{cases} \\ \sum\limits_{m=1}^{M} x_{m,n} = 1. \end{cases}$$

综上所述，基于基站智能开关载频算法，将整个过程表示为基站的动态分配载频机制．为了使整个蜂窝网络整体评价指标 Y 尽可能大，蜂窝网络中每个基站都应该分派最合适的载频数，使得既能满足用户流量需求，也能使得整体蜂窝网络的能耗达到最低．

6.3 仿真结果及分析

收集小区一周的流量数据，利用 matlab 编写代码，进行仿真．对不同阈值的仿真结果如下图25、26所示：

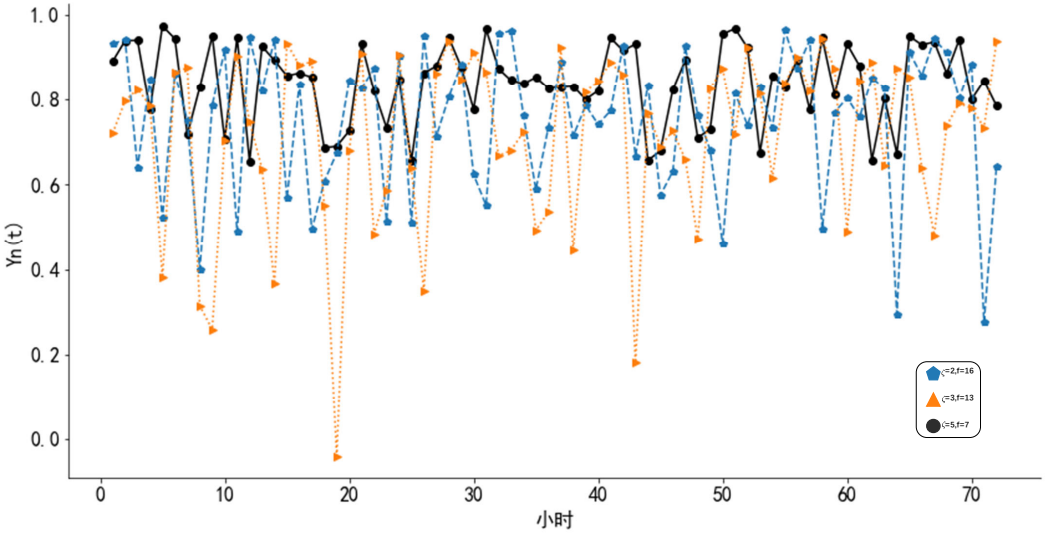

图 25 基站载频数对整体网络功耗影响 (前 70 小时)

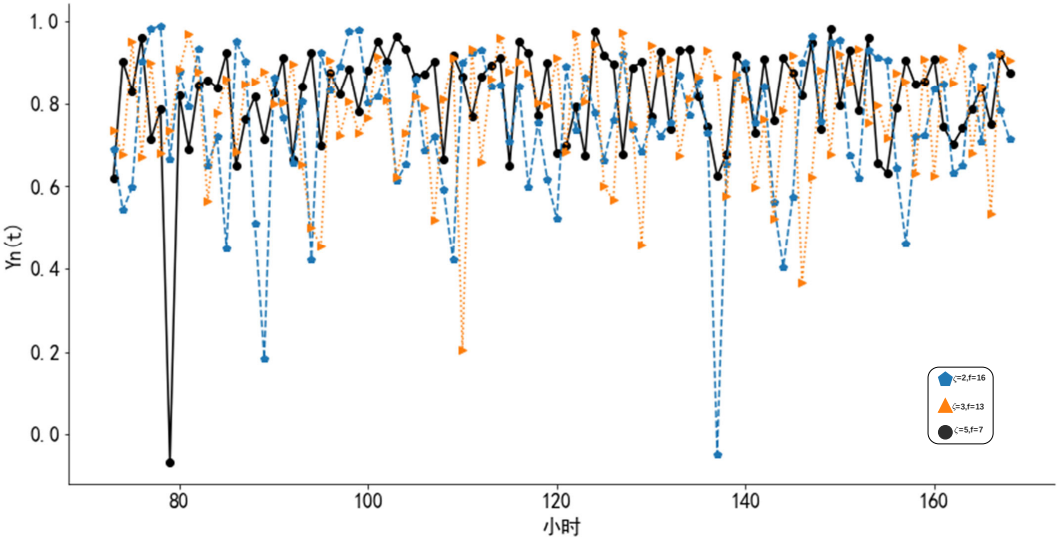

图 26 基站载频数对整体网络功耗影响 (后 80 小时)

从上图25、26可以看出，整体网络功耗收基站分配小区的起始载频数 f_{mn} 和基站动态分配载频数 ζ 有关，随着这两个变量的变化波动．为了确定最佳能达到网络能耗

最低且满足用户服务质量的基站载频数，通过计算机深度搜索，确定最佳的 ζ．最终确定当 $f_{mn}=7$、$\zeta=5$ 的时候，整体网络功耗波动最小，说明此时基站的载频数达到阈值，基站分配给小区的载频容量既能恰好满足用户的流量需求，且能达到用户的服务质量．在仿真中，由于基站载频容量与载频数存在函数关系 $\Psi(\cdot)$，$\Psi(\cdot)$ 是一个不减函数，为了计算方便，我们采用 $\Psi(x)=\ln(x+1)$ 作为函数关系，此时 $\Psi(\zeta+f_{mn})=2.485$，说明在仿真中所使用的基站开关载频的流量阈值为 2.485GB．

七、模型的检验

7.1 基站智能开关载频算法稳定性检验

对于问题二的求解，我们提出了基站智能开关载频算法，其中有两个不减函数 $g(\cdot)$ 和 $\Psi(\cdot)$．由于问题二没有给相关基站的数据，只有各个小区的流量数据，对此我们使用的不减函数是 $\ln(x+1)$．为了检验是否存在偶然性，我们对其作稳定性检验，任意选取其他不减函数，观察其结果，如下图：

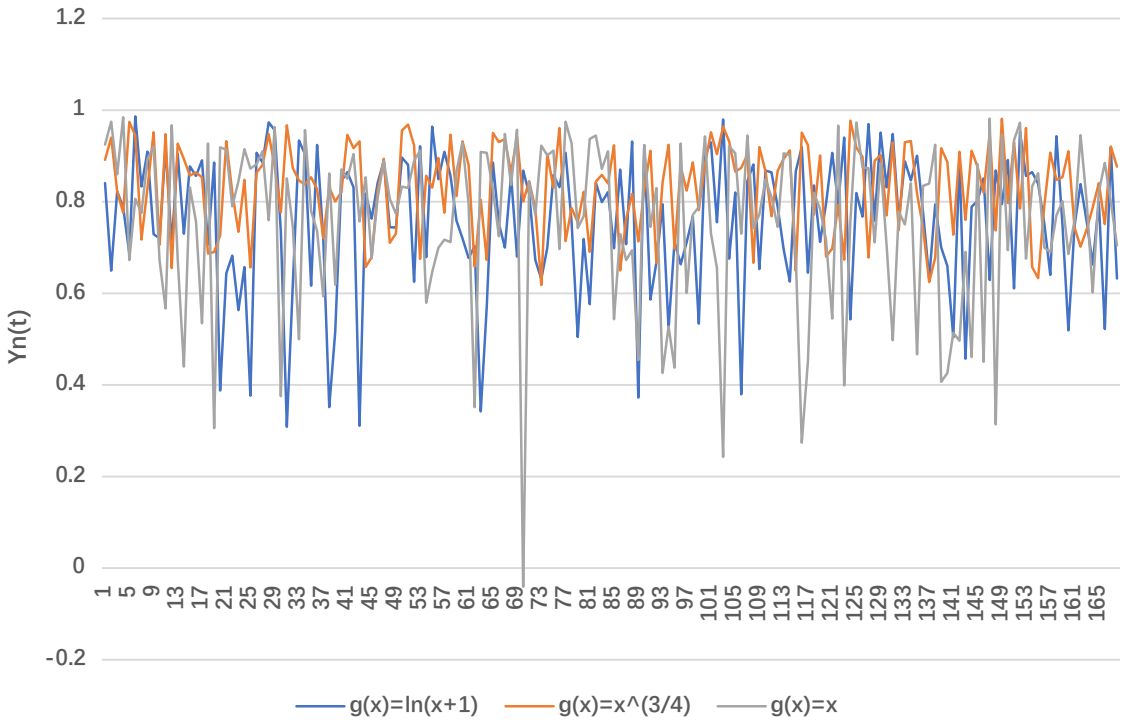

图 27　不同 $g(x)$ 对算法稳定性检验

从上图中看出，不同 $g(x)$ 对整体网络功耗的影响不大，且通过验证也是 $\zeta=5$ 和 $f_{mn}=7$ 的时候，即基站分配小区起始数为 7 的时候，基站动态调整载频为 5 的时候，整体网络功耗最小．从不同不减函数 $g(x)=\ln(x+1)$、$g(x)=x^{\frac{3}{4}}$、$g(x)=x$ 选取上，说明模型的稳健性较好．

八、模型的评价

8.1 模型的优点

1、本文使用二级聚类模型对小区进行分类，第一级使用流量消耗特征对小区进行聚类，第二级使用时间序列特征对小区进行聚类，考虑小区特征比较完善，对小区的分类也比较详细.

2、本文使用 Mini Batch K-Means 进行聚类，降低了聚类的计算时间.

3、提出了一种基站智能开关载频的方法，能根据小区的业务量需求，基站自动开关载频数，来满足小区的流量需求.

4、在基站载频阈值的研究中，充分考虑了用户的服务质量，在保证用户服务质量的基础上，保证基站能耗尽可能少，降低了基站能耗，对网络的能耗有较大的提高.

5、本文考虑了整个蜂窝网络结构模型，而非对单一的基站进行研究，所建立的模型能使整个蜂窝网络能耗最低.

8.2 模型的缺点

1、在基站阈值的仿真中获取的数据来着其他不同的参考论文，可能存在有点偏差.

2、在基站阈值仿真中有些变量受人为因素确定.

参考文献

[1] Cisco.Cisco visual networking index: Global mobile data traffic forecast update,2015-2020.2016.

[2] 王丽君, 韩涛, 薛曼琳, 俞侃. 基于基站选择性休眠的非线性加权算法研究 [J]. 武汉理工大学学报,2014,36(12):142-146.

[3] 纪勇. 基于时间序列特征的基站聚类 [J]. 电子世界,2016(14):165-167.

[4] 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.

[5] 李玮婷. 基于 WCDMA 的无线接入网节能技术研究 [D]. 电子科技大学,2012.

[6] D. Sculley.Web-Scale K-Means Clustering. Raleigh, 2010.

[7] Auer G, Blume O, Giannini V, et al. Energy efficiency analysis of the reference systems,areas of victories and target breakdown[R]. FP7 project EARTH - Deliverable D2.3 ,2010.

[8] HuangY,Al-QahtaniF,ZhongC,etal.PerformanceAnalysisofMultiuserMultipleAntennaRelayingNetworkswithCochannelInterferenceandFeedbackDelay[J].IEEETransactionsonCommunications,2014,62(1):59-73.

[9] 周紫阳. 蜂窝网络上的移动宽带数据流量特征分析 [D]. 南京理工大学,2018.

[10] 赵捷. 基于业务特征的蜂窝网络资源分配研究 [D]. 北京邮电大学,2018.

附　录

附录 1： 特征提取代码 (jupyter notebook 编写，由.ipynb 格式导出)

```
import pandas as pd
# -*- coding: utf-8 -*-
import json
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
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
sns.set(font_scale=1) #sns字体大小
sns.set(font='SimHei') # 解决Seaborn中文显示问题
# Set default font size
#设置默认字体大小
plt.rcParams['font.size'] = 16
sns.set(style="darkgrid") #这是seaborn默认的风格
sns.set(font_scale = 2)

pd.set_option('display.max_columns', 60)


path = r'G:\MathorCup\赛道A\附件1: 训练数据\CleanTrainData.csv'
reader = pd.read_csv(path,encoding='GB18030')
reader.columns=['索引', '日期','时间', '小区编号', '上行业务量GB',
    '下行业务量GB', '时间序列']

data = reader

data['时间序列'] = pd.to_datetime(data['时间序列'])
data['week'] = data['时间序列'].dt.weekday

def Day_And_Night_Difference(d):

    num_day =
```

```
        d.drop_duplicates('日期',keep='first',inplace=False).shape[0] #
        确定天数
    time_day_list =
        list(d.drop_duplicates('日期',keep='first',inplace=False)['日期'])
        # 存在的日期
    sta_num_shang = []
    sta_num_xia = []
    for i in time_day_list:
        day_max = d[d['日期']==i]['上行业务量GB'].max()
        day_min = d[d['日期']==i]['上行业务量GB'].min()
        sta_num_shang.append(day_max-day_min)

        day_max_1 = d[d['日期']==i]['下行业务量GB'].max()
        day_min_1 = d[d['日期']==i]['下行业务量GB'].min()
        sta_num_xia.append(day_max_1-day_min_1)
    Day_and_night_difference_s = sum(sta_num_shang)/num_day
    Day_and_night_difference_x = sum(sta_num_xia)/num_day
    return Day_and_night_difference_s,Day_and_night_difference_x

def Week_Difference(d):
    num_week = list(d['week'].value_counts().index) #数据中存在的星期一至日
    p1 =
        d.drop_duplicates('日期',keep='first',inplace=False)['week'].value_counts()

    work_day = []
    week_day = []
    D = 0
    W = 0

    work_day_xia = []
    week_day_xia = []

    for i in num_week:
        if i < 5:
            work_day.append(d[d['week']==i]['上行业务量GB'].sum())
                #工作日流量和
            work_day_xia.append(d[d['week']==i]['下行业务量GB'].sum())
                #工作日流量和
            D = D + p1[i]
        if i > 4:
            week_day.append(d[d['week']==i]['上行业务量GB'].sum())
```

```
            #周末流量和
            week_day_xia.append(d[d['week']==i]['下行业务量GB'].sum())
            #周末流量和
            W = W + p1[i]

    if D == 0:
        D = 1
    if W == 0:
        W = 1
    Week_difference = (sum(work_day)/D -
        sum(week_day)/W)/sum(work_day)/D
    Week_difference_xia = (sum(work_day_xia)/D -
        sum(week_day_xia)/W)/sum(work_day)/D

    return Week_difference,Week_difference_xia


def Var_Day(d):
    var_mon = d[d['时间']>=6][d[d['时间']>=6]['时间']<=12] #6点至12点数据
    var_after = d[d['时间']>=13][d[d['时间']>=13]['时间']<=18]
        #13点至18点数据
    var_day =
        var_mon['上行业务量GB'].var()+var_after['上行业务量GB'].var()
    var_day_xia =
        var_mon['下行业务量GB'].var()+var_after['下行业务量GB'].var()
    return var_day,var_day_xia


def Var_Night(d):
    var_n = d[d['时间']>=18][d[d['时间']>=18]['时间']<=24] #18点至24点数据

    var_night = var_n['上行业务量GB'].var()
    var_night_xia = var_n['下行业务量GB'].var()
    return var_night,var_night_xia

Community = []
DAND = []
WD = []
VD = []
VN = []
DAND_xia = []
```

```
WD_xia = []
VD_xia = []
VN_xia = []
com_list =
   data.drop_duplicates('小区编号',keep='first',inplace=False)['小区编号']
com_list = list(com_list)
com_list.sort(reverse=False)


for i in com_list:
   d = data[data['小区编号']==i]
   d['时间'] = d['时间序列'].dt.hour

   Day_and_night_difference_s,Day_and_night_difference_x =
      Day_And_Night_Difference(d)
   Week_difference,Week_difference_xia = Week_Difference(d)
   var_day,var_day_xia = Var_Day(d)
   var_night,var_night_xia = Var_Night(d)

   Community.append(i)
   DAND.append(Day_and_night_difference_s)
   WD.append(Week_difference)
   VD.append(var_day)
   VN.append(var_night)
   DAND_xia.append(Day_and_night_difference_x)
   WD_xia.append(Week_difference_xia)
   VD_xia.append(var_day_xia)
   VN_xia.append(var_night_xia)
   print(i)

whole_data = pd.DataFrame()
whole_data['Community']=Community
whole_data['Day_and_night_difference_s']=DAND
whole_data['Week_difference']=WD
whole_data['var_day']=VD
whole_data['var_night']=VN
whole_data['Day_and_night_difference_x']=DAND_xia
whole_data['Week_difference_xia']=WD_xia
whole_data['var_night_xia']=VN_xia
whole_data['var_day_xia']=VD_xia
```

```
whole_data.to_csv('whole_data3.csv')

def save_csv(SourceData,Name):
   SourceData.to_csv(Name, encoding='GB18030')

def SampEn(U, m=2, r=3):
   def _maxdist(x_i, x_j):
      return max([abs(ua - va) for ua, va in zip(x_i, x_j)])
   def _phi(m):
      x = [[U[j] for j in range(i, i + m - 1 + 1)] for i in range(N -
         m + 1)]
      C = [len([1 for j in range(len(x)) if i != j and _maxdist(x[i],
         x[j]) <= r]) for i in range(len(x))]
      return sum(C)
   N = len(U)
   if _phi(m)==0:
      return 0
   else:
      return -np.log(_phi(m + 1) / _phi(m))


def ApEn(U, m=2, r=3):
   def _maxdist(x_i, x_j):
      return max([abs(ua - va) for ua, va in zip(x_i, x_j)])
   def _phi(m):
      x = [[U[j] for j in range(i, i + m - 1 + 1)] for i in range(N -
         m + 1)]
      C = [len([1 for x_j in x if _maxdist(x_i, x_j) <= r]) / (N - m
         + 1.0) for x_i in x]
      if (N-m+1.0)==0:
         return 0
      else:
         return (N - m + 1.0)**(-1) * sum(np.log(C))
   N = len(U)
   return abs(_phi(m+1) - _phi(m))


CleanData = reader

CleanData = pd.read_csv('./CleanTrainData.csv', encoding='GB18030')
```

```
CleanData.drop(['Unnamed: 0'],axis=1,inplace=True)
CleanData.columns=['日期','时间', '小区编号', '上行业务量GB',
   '下行业务量GB', '时间序列']


Dup_data_Sum_group =
   CleanData.groupby(by=['小区编号','日期'])['上行业务量GB',
   '下行业务量GB'].sum()
Dup_data_Sum = Dup_data_Sum_group.reset_index(drop=True)

Dup_data_one = CleanData.drop_duplicates(subset=['小区编号',
                                '日期'],keep="first").reset_index(drop=True)

no_Dup =
   CleanData.drop_duplicates(subset=['小区编号','日期'],keep=False)

Dup_data_one['上行业务量GB'] = Dup_data_Sum['上行业务量GB']
Dup_data_one['下行业务量GB'] = Dup_data_Sum['下行业务量GB']

TrainDataResult = pd.concat([Dup_data_one, no_Dup])
CleanDataSum=TrainDataResult.drop(['时间'],axis=1)
CleanDataSum.sort_values(by=['小区编号','日期'] , inplace=True,
   ascending=True)

Dup_data_Mean_group =
   CleanDataSum.groupby(by=['小区编号'])['上行业务量GB',
   '下行业务量GB'].mean()
Dup_data_Mean = Dup_data_Mean_group.reset_index(drop=True)

Dup_data_one = CleanDataSum.drop_duplicates(subset=['小区编号'],
                                  keep="first").reset_index(drop=True)

no_Dup = CleanDataSum.drop_duplicates(subset=['小区编号'],keep=False)

Dup_data_one['上行业务量GB'] = Dup_data_Mean['上行业务量GB']
Dup_data_one['下行业务量GB'] = Dup_data_Mean['下行业务量GB']

TrainDataResult = pd.concat([Dup_data_one, no_Dup])
CleanDataDayMean=TrainDataResult.drop(['日期'],axis=1)
CleanDataDayMean.sort_values(by=['小区编号'] , inplace=True,
   ascending=True)
```

```
CleanDataDayMean=TrainDataResult.drop(['时间序列'],axis=1)

save_csv(CleanDataDayMean,'上下行业务量日均值.csv')

upY=CleanDataSum['上行业务量GB']>=1
dnY=CleanDataSum['下行业务量GB']>=1
upX=CleanDataSum['上行业务量GB']<=0.2
dnX=CleanDataSum['下行业务量GB']<=0.2
Date=CleanDataSum['日期']
UpY=pd.concat([Date,CleanDataSum['小区编号'],upY,dnY,upX,dnX],axis=1)

UpY.columns=['日期','小区编号','上行业务量GB大于1GB天数','下行业务量GB大于1GB天数',
                      '上行业务量GB小于0.2GB天数','下行业务量GB小于0.2GB天数']

Dup_data_sum_group =
   UpY.groupby(by=['小区编号'])['上行业务量GB大于1GB天数',
    '下行业务量GB大于1GB天数','上行业务量GB小于0.2GB天数'
                               ,'下行业务量GB小于0.2GB天数'].sum()
Dup_data_sum = Dup_data_sum_group.reset_index(drop=True)

Dup_data_one = UpY.drop_duplicates(subset=['小区编号'],
                            keep="first").reset_index(drop=True)

no_Dup = UpY.drop_duplicates(subset=['小区编号'],keep=False)

Dup_data_one['上行业务量GB大于1GB天数'] =
   Dup_data_sum['上行业务量GB大于1GB天数']
Dup_data_one['下行业务量GB大于1GB天数'] =
   Dup_data_sum['下行业务量GB大于1GB天数']
Dup_data_one['上行业务量GB小于0.2GB天数'] =
   Dup_data_sum['上行业务量GB小于0.2GB天数']
Dup_data_one['下行业务量GB小于0.2GB天数'] =
   Dup_data_sum['下行业务量GB小于0.2GB天数']

TrainDataResult = pd.concat([Dup_data_one, no_Dup])
UpYSum=TrainDataResult.drop(['日期'],axis=1)
UpYSum.sort_values(by=['小区编号'] , inplace=True, ascending=True)

save_csv(UpYSum,'上下行大于1GB小于0.2GB的天数.csv')
```

```
Zone=np.unique(CleanData['小区编号'].values)

CleanDataSum['UpApEn'] = CleanDataSum['UpSampEn'] =
    CleanDataSum['UpFuzEn'] = 0
CleanDataSum['DnApEn'] = CleanDataSum['DnSampEn'] =
    CleanDataSum['DnFuzEn'] = 0

ApEnUp=[]
ApEnDn=[]
SampEnUp=[]
SampEnDn=[]

for i in Zone:
    ZoneData=CleanDataSum[CleanDataSum['小区编号']==i]
    TempUp=ZoneData['上行业务量GB']
    TempDn=ZoneData['下行业务量GB']
    UP=TempUp.values
    DN=TempDn.values
    ApEnUp.append(ApEn(UP,2,3))
    SampEnUp.append(SampEn(UP,2,3))
    ApEnDn.append(ApEn(DN,2,3))
    SampEnDn.append(SampEn(DN,2,3))

Data=np.vstack([Zone,ApEnUp,ApEnDn,SampEnUp,SampEnDn])
Data=pd.DataFrame(Data.T)
Data.columns=['ZoneNum','ApEnUp','ApEnDn','SampEnUp','SampEnDn']
Data.to_csv('./Data.csv')

UPDN = pd.read_csv('./上下行大于1GB小于0.2GB的天数.csv',
    encoding='GB18030')
DayMean = pd.read_csv('./上下行业务量日均值.csv',encoding='GB18030')
WholeData = pd.read_csv('./whole_data.csv')
DataEN = pd.read_csv('./Data.csv')
WholeData=WholeData.rename(columns={'Community':'小区编号'})
Data=pd.merge(UPDN,DayMean,on='小区编号')
Data=pd.merge(Data,WholeData,on='小区编号')
Data=pd.merge(Data,DataEN,on='小区编号')
DataEN.drop(['Unnamed: 0'],axis=1,inplace=True)
DataEN=DataEN.rename(columns={'ZoneNum':'小区编号'})
Data.fillna(method='ffill',inplace=True)
Data=Data.rename(columns={'上行业务量GB':'日均上行业务量GB',
```

```
                                ’下行业务量GB’:’日均下行业务量GB’})
Data=Data.replace(np.inf, 0)
save_csv(Data,’特征矩阵.csv’)
```

附录 2： 聚类代码 (jupyter notebook 编写，由.ipynb 格式导出)

```
import pandas as pd
import json
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

import warnings
warnings.filterwarnings(’ignore’)
plt.style.use(’ggplot’)
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
# Set default font size
#设置默认字体大小
plt.rcParams[’font.size’] = 16

# Display up to 60 columns of a dataframe
# 最多显示60列
pd.set_option(’display.max_columns’, 60)

reader =
   pd.read_csv(’G:\MathorCup\复赛\特征矩阵.csv’,encoding=’GB18030’)

from sklearn import preprocessing
from sklearn.cluster import MiniBatchKMeans
from sklearn.metrics import silhouette_score

min_max_scaler = preprocessing.MinMaxScaler()
x = min_max_scaler.fit_transform(reader.iloc[:,1:7])

SSE = []
for k in range(1,30):
   estimator = MiniBatchKMeans(n_clusters=k) # 构造聚类器
```

```
    estimator.fit(x)
    SSE.append(estimator.inertia_)
X1 = range(1,30)
plt.rcParams['font.size'] = 16
fig, ax = plt.subplots(figsize=(14,7))
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X1,SSE,'o-')
plt.savefig('误差平方和.pdf')

from sklearn.metrics import silhouette_score

Scores = []
for k in range(2,10):
    estimator = MiniBatchKMeans(n_clusters=k)
    estimator.fit(x)
    Scores.append(silhouette_score(x,estimator.labels_,metric='euclidean'))

X = range(2,10)
plt.xlabel('k')
plt.ylabel('Silhouette Coefficient')
plt.plot(X,Scores,'o-',c='y')
plt.savefig('轮廓系数.pdf')

kmeans = MiniBatchKMeans(n_clusters=4)
kmeans.fit(x)
y_kmeans = kmeans.predict(x)
plt.style.use('ggplot')

pd.DataFrame(kmeans.cluster_centers_).to_csv('./cluster_centers.csv')
reader['聚类'] = y_kmeans
reader['聚类'].value_counts()
reader.to_csv('./第一层聚类.csv',encoding='GB18030')
reader=pd.read_csv('./第一层聚类.csv',encoding='GB18030')
reader.drop('Unnamed: 0',axis=1,inplace=True)
reader[reader['聚类']==0]
reader0=reader[reader['聚类']==0]
min_max_scaler = preprocessing.MinMaxScaler()
x0 = min_max_scaler.fit_transform(reader0.iloc[:,7:-1])
```

```
SSE = []
for k in range(1,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x0)
   SSE.append(estimator.inertia_)
X1 = range(1,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X1,SSE,'o-',c='y')
plt.savefig('误差平方和.pdf')

from sklearn.metrics import silhouette_score

Scores = []
for k in range(2,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x0)
   Scores.append(silhouette_score(x0,estimator.labels_,metric='euclidean'))

X = range(2,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('Silhouette Coefficient')
plt.plot(X,Scores,'o-',c='y')
plt.savefig('轮廓系数.pdf')

kmeans0 = MiniBatchKMeans(n_clusters=3)
kmeans0.fit(x0)
y_kmeans0 = kmeans0.predict(x0)
plt.style.use('ggplot')

pd.DataFrame(kmeans0.cluster_centers_).to_csv('./layer0_cluster_centers.csv')
reader0['Layer0聚类'] = y_kmeans0


reader0.to_csv('./第0种第二层聚类.csv',encoding='GB18030')


reader1=reader[reader['聚类']==1]
```

```
min_max_scaler = preprocessing.MinMaxScaler()
x1 = min_max_scaler.fit_transform(reader1.iloc[:,7:-1])

SSE = []
for k in range(1,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x1)
   SSE.append(estimator.inertia_)
X1 = range(1,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X1,SSE,'o-',c='y')
plt.savefig('误差平方和.pdf')

from sklearn.metrics import silhouette_score

Scores = []
for k in range(2,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x1)
   Scores.append(silhouette_score(x1,estimator.labels_,metric='euclidean'))

X = range(2,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('Silhouette Coefficient')
plt.plot(X,Scores,'o-',c='y')
plt.savefig('轮廓系数.pdf')

kmeans1 = MiniBatchKMeans(n_clusters=6)
kmeans1.fit(x1)
y_kmeans1 = kmeans1.predict(x1)
plt.style.use('ggplot')

pd.DataFrame(kmeans1.cluster_centers_).to_csv('./layer1_cluster_centers.csv')
reader1['Layer1聚类'] = y_kmeans1

reader1.to_csv('./第1种第二层聚类.csv',encoding='GB18030')
```

```
reader2=reader[reader['聚类']==2]
min_max_scaler = preprocessing.MinMaxScaler()
x2 = min_max_scaler.fit_transform(reader2.iloc[:,7:-1])

SSE = []
for k in range(1,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x2)
   SSE.append(estimator.inertia_)
X1 = range(1,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X1,SSE,'o-',c='y')
plt.savefig('误差平方和.pdf')

from sklearn.metrics import silhouette_score

Scores = []
for k in range(2,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x2)
   Scores.append(silhouette_score(x2,estimator.labels_,metric='euclidean'))

X = range(2,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('Silhouette Coefficient')
plt.plot(X,Scores,'o-',c='y')
plt.savefig('轮廓系数.pdf')

kmeans2 = MiniBatchKMeans(n_clusters=3)
kmeans2.fit(x2)
y_kmeans2 = kmeans2.predict(x2)
plt.style.use('ggplot')


pd.DataFrame(kmeans2.cluster_centers_).to_csv('./layer2_cluster_centers.csv')
reader2['Layer2聚类'] = y_kmeans2
```

```
reader2.to_csv('./第2种第二层聚类.csv',encoding='GB18030')


reader3=reader[reader['聚类']==3]
min_max_scaler = preprocessing.MinMaxScaler()
x3 = min_max_scaler.fit_transform(reader3.iloc[:,7:-1])

SSE = []
for k in range(1,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x3)
   SSE.append(estimator.inertia_)
X1 = range(1,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('SSE')
plt.plot(X1,SSE,'o-',c='y')
plt.savefig('误差平方和.pdf')

from sklearn.metrics import silhouette_score

Scores = []
for k in range(2,10):
   estimator = MiniBatchKMeans(n_clusters=k)
   estimator.fit(x3)
   Scores.append(silhouette_score(x3,estimator.labels_,metric='euclidean'))

X = range(2,10)
plt.style.use('Solarize_Light2')
plt.xlabel('k')
plt.ylabel('Silhouette Coefficient')
plt.plot(X,Scores,'o-',c='y')
plt.savefig('轮廓系数.pdf')


kmeans3 = MiniBatchKMeans(n_clusters=4)
kmeans3.fit(x3)
y_kmeans3 = kmeans3.predict(x3)
plt.style.use('ggplot')
```

```
pd.DataFrame(kmeans3.cluster_centers_).to_csv('./layer3_cluster_centers.csv')
reader3['Layer3聚类'] = y_kmeans3



reader3.to_csv('./第3种第二层聚类.csv',encoding='GB18030')
```

附录 3： 仿真代码 (matlab 编写)

FuncPBS.m

```
 function PBS = FuncPBS(k,d,t,TBS,f)
 PBS = G(TBS+FuncPsi(FuncZeta(k,d,t,f)));
 end
\begin{matlab}
\textbf{FuncPsi.m}
\begin{matlab}
 function psi = FuncPsi(t)
 if t>=0
    psi=log(t+1);
 else
    psi=0;
 end
 end
```

FuncPtotal.m

```
 function Ptotal = FuncPtotal(k,d,t,TBS,f)
 Psleep = 75;
 NTRX = 6;
 Pmax = 20;
 P0 = 130;
 deltap = 4.7;
 if FuncPBS(k,d,t,TBS,f) == 0
    Ptotal=NTRX*Psleep;
 elseif FuncPBS(k,d,t,TBS,f) > 0 && FuncPBS(k,d,t,TBS,f) < Pmax
    Ptotal=NTRX*(P0+deltap*FuncPBS(k,d,t,TBS,f));
 else
    Ptotal=NTRX*(P0+deltap*Pmax);
 end
```

```
end
```

FuncSINR.m

```
function sinr = FuncSINR(k,d,t,h,TBS,PBSj,f)
sinr = (((FuncPBS(k,d,t,TBS,f)*h^2)/(PBSj))*(rand/10+1));
end
```

FuncZeta.m

```
%fmn 基站m分配给小区n的载频数
function zeta = FuncZeta(k,d,t,f)
if FuncPsi(f+k)>d(t) && FuncPsi(f+k-1)<d(t)
   zeta=k;
elseif d == FuncPsi(f)
   zeta = 0;
elseif FuncPsi(f-k)>d(t) && FuncPsi(f-k-1)<d(t)
   zeta = -k;
else
   zeta=0;
end
end
```

FZ.m

```
x=[];
y=[];
t=[];

% Y = y(t)*x';
% y=omega1*mapminmax(FuncSINR(d,t,h,x))+omega2*mapminmax(Ptotal);
% mapminmax(FuncSINR)
Data1=xlsread('./小区一周流量汇总.xlsx');

% h 干扰性系数
h = 1.1;
dj=Data1(:,1);
SINR=[];
for k = 1:5
   for f = 1:100
      for t = 1:168
         TBS = sum(Data1(t,:));
         for i = 1:10
```

```
            d=Data1(:,i);
            PBSj=0;
            for j = 1:10
               if j==i
               else
                  dj=Data1(:,j);
               end
               PBSj=PBSj+FuncPBS(k,dj,t,TBS,f);
            end
            SINR(k,f,t)=FuncSINR(k,d,t,h,TBS,PBSj,f);
         end
      end
   end
end

Ptotal=[];
for k=1:5
   for f=1:100
      for t=1:168
         TBS=sum(Data1(t,:));
         for i=1:10
            d=Data1(:,i);
            Ptotal(k,f,t)=FuncPtotal(k,d,t,TBS,f);
         end
      end
   end
end
```

G.m

```
function g = G(x)
g=log(x+1);
end
```

plt.m

```
mapminmax(Ptotal, 0, 1);

for k=1:5
   for f=1:100
      SINRBar(k,f,:)=sqrt(sqrt(mapminmax(reshape(SINR(k,f,:),[1,168]),0,1)));
   end
```

```
end

for k=1:5
   for f=1:100
      PtotalBar(k,f,:)=mapminmax(reshape(Ptotal(k,f,:),[1,168]),0,1);
   end
end

plot(1:168,reshape(SINRBar(1,1,:),[1,168]))

plot(1:168,reshape(PtotalBar(1,1,:),[1,168]))

sa=[];
for k=1:5
   for f=1:100
      sa(k,f,:)=(reshape(SINRBar(k,f,:),[1,168])-...
                 reshape(PtotalBar(k,f,:),[1,168])*0.1);
   end
end

plot(1:168,reshape(sa(1,1,:),[1,168]))

summ=[];
for k=1:5
   for f=1:100
      summ(k,f)=sum(reshape(sa(k,f,:),[1,168]));
   end
end

value=max(max(summ));
[Row,Col]=find(value==summ);

plot(1:168,reshape(sa(5,7,:),[1,168]))

result=reshape(sa(5,:,:),[100,168]);

csvwrite('result4.csv',result)
```