UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LARGE-SCALE DATA PROCESSING FOR DETECTING ACTIVITY ZONES IN MILAN

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

By

ARSALAN DARBANDI
Norman, Oklahoma
2017

LARGE-SCALE DATA PROCESSING FOR DETECTING ACTIVITY ZONES IN MILAN

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

_____
Dr. Ali Imran, Chair

_____
Dr. Samuel Cheng

_____
Dr. Gregory G. MacDonald

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my family.

I would like to express my deepest gratitude to my advisor, Dr. Ali Imran, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate his vast knowledge and skill in many areas who has led me to the field of big data and data science. His enthusiasm for research has motivated me to work hard all the time.

I would like to thank the other members of my committee, Dr. Samuel Cheng, and Dr. Macdonald for guiding my research for the past several years and helping me to develop my background in data mining, and machine learning.

Many thanks to members of BSONLab, Hasan Farooq, Ahmad Asghar, Azar Taufique, Umair Hashmi, Haneya Qureshi, Sinasi Cetinkaya, and other students in the department of telecommunication and electrical and computer engineering at OU-Tulsa. My research would not have been possible without their helps.

I want to thank Murray Patricia from writing services at OU- Tulsa for editing my thesis. I am grateful to have wonderful supporting friends and delightful people like Renee Wagenblatt and Krista Pettersen, Lawson A. Lee, and Joshua M. Davis in my life.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am very much thankful to my girlfriend for her love, understanding, prayers and continuing support to complete this research work. Also, I express my thanks to my sisters, brother, for their support and

valuable prayers. My Special thanks goes to my sister for the keen helped to complete this thesis successfully. Finally, there are my friends. We were not only able to support each other by deliberating over our problems and findings, but also happily by talking about things other than just our papers.

Table of Contents

List of Tables

List of Figures

Abstract

Small cell networks are complements for existing networks to improve quality of service (QoS), capacity, and coverage. The primary purpose of this thesis is to mine mobile network data to provide an algorithm that mobile network operators can use to determine the best small cell network topology automatically instead of manually. The main drawbacks for deploying topology manually is the cost and time the effort consumes. Therefore, we have developed our algorithm based on a real dataset collected from the city of Milan, Italy, to show our approach for automating the task of identifying the best small cell network topology to implement for specific situations.

First, we designed an algorithm to adjust all types of call detail records (CDRs) for small cells at mmWave frequencies. Moreover, the information produced by this algorithm together with spatio-temporal mobile data reflected a pattern of user activity in our sample city.

Second, we compared k-means, density-based spatial clustering of applications with noise (DBSCAN), hierarchical algorithm, and two more clustering algorithms to find the best clustering method for small cell network topology. In addition, we developed an ant colony optimization algorithm to produce spatio-temporal mobile dataset and provide a novel small cell network planning solution.

Finally, we ascertained the best topology by using machine learning clustering and an optimization algorithm. Our topology came up with 2097 mmWave cell sites that covered 1,853.28 sq. km, 424 small cell sites for that covered 11,276.3 sq. km, and 25 macro cell sites that covered 22,090 sq. km.

Chapter 1: Introduction

1.1 Background

Wearable technologies and smartphones have dramatically evolved. Over the years, those technologies have grown to use more and more data. Global mobile data traffic has grown 63% from 2015 to 2016 [1]. That growth represents a usage of 7 exabytes per month. That value is expected to grow to 49 exabytes per month by 2021. These statistics show a high internet traffic demand for the future [1]. Fourth-generation (4G) data traffic reached 69% of mobile traffic in 2016 [2]. Based on these numbers, one can see the growing demand for capacity, capability, and quality-of-service (QoS) to accommodate mobile data traffic.

1.2 Motivation

mmWave small cells have been adopted to fill gaps in the macro cell network where demand is growing for capacity, capability, and QoS. mmWave Small cells can be used in existing cellular network infrastructure at a lower deployment costs than adding large cells to an existing cellular network which cannot solve QoS of the network. Therefore, they are a cost-effective solution for mobile operators and are fast becoming the next generation of mobile network.

Although small cell is an elegant way to increase capacity, capability, and QoS, it poses some challenges. Some of those challenges are interference, backhaul issues, and energy efficiency. [3]. One of the main challenges for the service provider is backhaul issues. This challenge consists of several challenges. One of them is demand for a way to plan how many and where to deploy small cells.

The length of coverage for each small cell is between 10 m-100 m. Therefore, for an urban area such as Milan, Italy, with an area of 112,000 sq. m, a clustering technique would be beneficial to determine the number and location of the small cells to deploy. Optimally, a clustering methodology must be able to estimate the number of clusters to deploy to accommodate internet traffic of a specific area of Milan, Italy, that has a high data traffic demand.

There are two other challenges to consider when trying to improve a data network to accommodate the data demand. One challenge is sufficient human resources to install and maintain the small cells to be added to the network. The other challenge is the associated cost required to add the human resources. The cost of installation and maintenance of this type of network is less than if more large cells were added to the existing networks. However, if enough small cells were added to the existing network to accommodate the data traffic for the entire city of Milan, the cost would be higher that if a sufficient number of large cells were added to the existing network to accommodate the entire city of Milan. area of 112 sq. km that required around 1,120 to handle the cluster demand.

Another challenge for small cell backhaul is spatio temporal traffic. According to research conducted by AT&T, users are split into 7 groups with each group having a specific data usage pattern per day [4]. For instance, students have low SMS activity from 7:30 a.m. to 11:30 a.m.; however, from 3 p.m. to 10 p.m., they have heavy SMS activity. Therefore, each area of a city has specific patterns of traffic during various times of day, and the technique for clustering must consider this parameter.

In this research, several activities were performed on a real dataset to solve the problem of finding the minimum number of small cells (clusters) needed to accommodate data traffic while taking spatio temporal traffic into consideration. From this research, an application was created to execute this technique on all call detail record (CDR), or dataset, that contains traffic load information for specific periods during the day for various areas of the city.

Based on the actual findings of the analysis of the Milan data set, it became clear that network improvement could be restricted to just those areas of the city that had the heaviest data traffic but the lowest QoS. Therefore, this research focused on determining the areas of the city having the heaviest data traffic and creating a new network configuration that includes smalls. Therefore, a tool for developing the clustering plan had to be created, which in this case was a multi-step algorithm.

1.3 **Contributions**

This thesis investigates how to implementing small cell planning for mmWave cellular networks. This study is divided in two parts. The first part shows the algorithm for generating a new dataset that is adjusted for small cells. The second part shows different clustering methods and compares them to determine the best method for clustering and using an optimization algorithm to find the best number of clusters to add to the network.

Generating Dataset for Small-Cells Deployment

A CDR dataset is primarily generated to issue a billing record for a user; however, it also is used for tracking call activity for different purposes. [4], [6], and [9] are some research using CDRs for different purposes. In this study, we made an algorithm by

obtaining a CDR dataset and extracting CDR information from it, making a dataset for small cell research.

Chapter 2: Generating Appropriate Data for Small Cells shows the methods that were used for making the algorithm to generate a dataset for small cell. As mentioned earlier, small cells provide the benefit of a coverage length between 10m-100m. Therefore, we made an algorithm that generates a dataset from CDR information with accuracy of 5 m length. This algorithm prepares the dataset for implementing the clustering methodology.

Chapter 2 describes how data that is used in clustering, includes the preprocessing process. Preprocessing is one of the important parts of this study because the information taken from the CDRs can be used for other studies. For instance, we provided some results that not only can be used for other cellular network operators but also can be used for other industries and businesses. We provided results that an industry or business reading the data can understand patterns of travelling and mobile activity of users for a location at different times of a day. For instance, digital marketing agencies can find the best location and time of day that has a high density of users and would be a good location for installing billboards or presenting something. Or, a small business such as a coffee shop can find the best locations to setup its business because it is important to detect the life style of an area such as nightlife in residential and commercial areas. In parts three of chapter 2, different schema of the algorithm is described. The last section of the chapter presents and explanation of how the dataset was generated from CDRs.

In chapter 3 different methods for planning small cell clustering is explained as well as reasons to select some of those methods. Next a comparison of those methods for finding the best clustering is addressed. The dataset referred to in chapter 3 is a subset of the dataset generated in the activities described in chapter 2. There are three important criteria for clustering methods:

Firstly, geographical size of each cluster. As mentioned earlier, for clustering methods, this is one of the important criteria that must be considered.

Secondly, traffic load for each cluster. Select the best clustering method that equalizes the traffic size of each cluster as much as possible.

Thirdly, optimal number and optimal members of clusters. Clusters must have high QoS while having low energy consumption. Therefore, optimization brings the best clustering method for this achievement.

Fourthly, convex small cell clusters. A convex shape is a simple area (not self-intersecting) in which no line between two points on the boundary ever go outside the polygon when they are connecting to each other.

In summary, chapter 3 briefly explains the dataset and then we describe the preprocessing steps devised for better understanding of the dataset. Then different methods that have been used and also techniques for optimization is explained. Then on next chapter, cluster validation and results are provided.

Chapter 2: Generating Appropriate Data for Small Cells

2.1 Milan Data

Telecom Italia and SpazioDati published the first open source telecommunication data through the first Big Data Challenge in 2014 [5]. The challenge was designed to stimulate ideas in the big data field. Due to the high demand for the big data produced by those companies, after the competition they released the datasets under the name of Open Big Data [5]. Open Big Data has provided datasets for Milan and Trento in Italy. These datasets are the Milan and Trento Grid, Telecommunications – SMS, Call, Internet – MI, Telecommunications – MI to Provinces, etc. In this study, the datasets for the city of Milan were used as they covered a larger area than the Trento datasets. Moreover, it encompassed urban and rural areas.

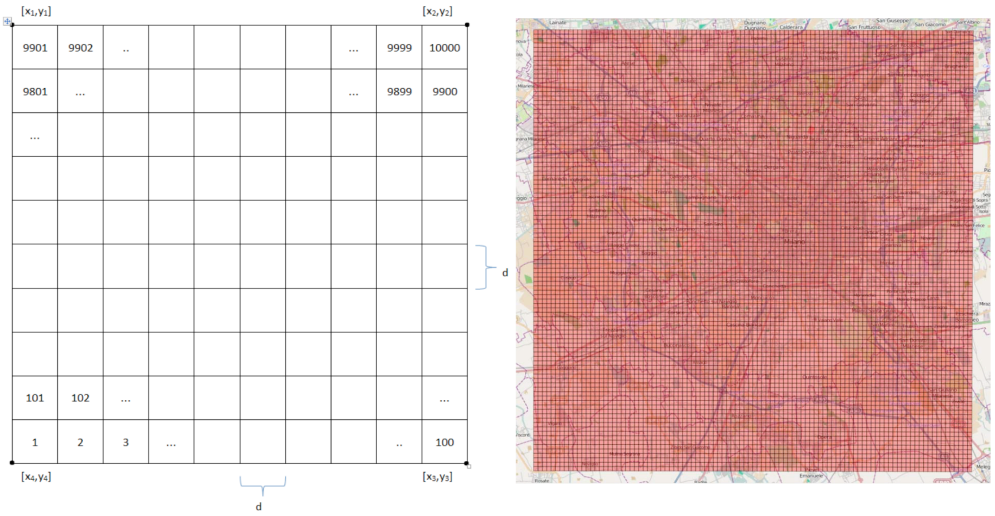
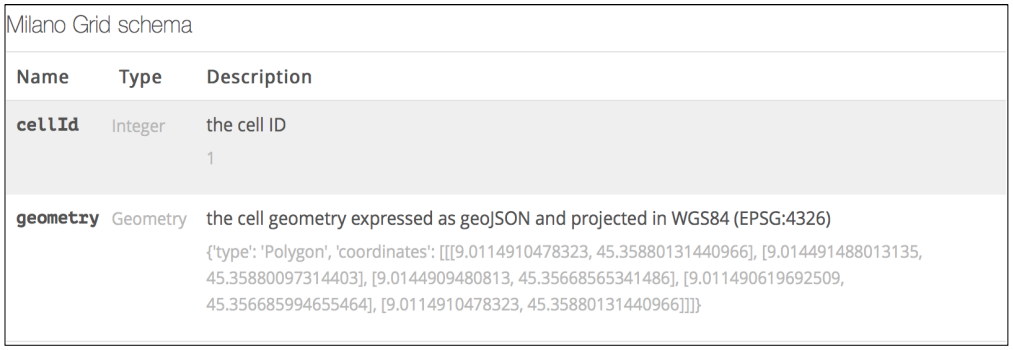


Figure 1. Milan Gridding [5]

Some of the datasets have gridding, which is based on the Milan grid. Figure 1 shows the grids for Milan, an area of 550 km^2. Each square grid has an ID. There are 10,000 square grids, and each one of them is 235m × 235m. Figure 2 shows a sample of the labeling of each grid square and longitude and latitude value of each grid square.



Milano Grid schema

Name	Type	Description
cellId	Integer	the cell ID 1
geometry	Geometry	the cell geometry expressed as geoJSON and projected in WGS84 (EPSG:4326) {'type': 'Polygon', 'coordinates': [[[9.0114910478323, 45.35880131440966], [9.014491488013135, 45.35880097314403], [9.0144909480813, 45.35668565341486], [9.011490619692509, 45.356685994655464], [9.0114910478323, 45.35880131440966]]]}

Figure 2. Milan Square Geometry Location [5]

Telecommunications – SMS, Call, Internet – MI is one of the important datasets for Milan. This dataset provided information about the telecommunication activity of the square grids during the months of November and December, 2013. We used this dataset for implementing mmWave small cells. The size of the dataset is an estimated 40Gb.

According to Open Big Data, this dataset was generated from the CDRs. CDRs contain various attributes of a call that show the user activity. CDRs are used for several purposes such as billing and network management [6]. The following attributes are the fields of CDRs from Open Big Data that present activities of users:

- Received SMS: Shows the number of SMSs a square grid received
- Sent SMS: Shows the number of SMSs a square grid sent
- Incoming Calls: Shows number of calls a square grid received
- Outgoing Calls: Shows number of calls a square grid issued

- Internet: Shows either a square grid starts or ends using internet connection when a CDR is generate

The activity attributes of the current dataset used in this project were formed by combining several aforementioned elements from Open Big Data because of data privacy policies. Therefore, each activity attribute shows just the level of activity of the square grids but does not show any scale. In addition, higher value means higher activity for a specific activity attribute.

Table 1 presents schema of the Milan city dataset. The first column is the grid square ID, which is numeric. IDs of squares range from 1 to 1000. The **time** column is shows time intervals, which are numerical and expressed
in Unix format.

Table 1. SMS, Call, Internet Data of Milan

id	time	Country	SMS_in	SMS_out	Call_in	Call_out	Internet
1	1.383260e+12	0	0.08136262	*NA*	*NA*	*NA*	*NA*
1	1.383260e+12	39	0.14186425	0.156787005	0.16093794	0.05227485	11.02836638
1	1.383261e+12	0	0.13658782	*NA*	*NA*	0.02730046	*NA*
1	1.383261e+12	33	*NA*	*NA*	*NA*	*NA*	0.02613742
1	1.383261e+12	39	0.27845208	0.119925720	0.18877717	0.13363747	11.10096345
1	1.383262e+12	0	0.05343789	*NA*	*NA*	*NA*	*NA*
1	1.383262e+12	39	0.33064143	0.170952030	0.13417624	0.05460093	10.89277060
1	1.383262e+12	0	0.02613742	*NA*	*NA*	*NA*	*NA*
1	1.383262e+12	39	0.68143408	0.220815299	0.02730046	0.05343789	8.62242459

The beginning of Unix time is January 1st, 1970, at UTC, and millisecond is the time interval for this timing format [7]. The time period of recoding activities for this dataset is 10 minutes. The Country column shows a code in numeric format and is the country code of the phones used by callers for each square ID. For instance, if the country code of a row equals 0, this means all SMS and call records of that row belong to Milan citizens. SMS-in and SMS-out activities are shown in two columns, and both are numerical, and present levels of activity in terms of sending and receiving SMS inside the grid square with the specified ID, during a 10-minutes interval. As with the SMS-in and SMS-out activities, the last two more columns show activity information for call-in and call-out. The last column is presenting internet traffic activity. Values in the last column, internet, are numerical and show internet traffic inside the grid square with a specific ID. Format of all files are csv. "If no activity was recorded for a field specified in the schema above then the corresponding value is missing from the file" [5].

2.2 Data Preprocessing and Visualization

One of the important steps in our data analysis was data preprocessing because it formatted the dataset for implementing machine learning tools and data analyzing techniques. by applying preprocessing, noise, outliers, missing values, and duplicate data to be detected [8].

According to Table 1, which shows some rows of SMS, call, and internet data for Milan for a day, some values of SMS-in, SMS-out, Call-in, Call-out, and Internet are missing and are shown as *NA*. Also

Table 1, shows that in some cases for the same ID and time, we have more than one row. First of all, we might have duplicate elements in rows in the dataset because of the country attribute. For example, rows 1 and 2 of Table 1 have the same ID and time, but they differ in country code. It shows that for ID = 1, there is more than one telecom network nationality (country code). Since we are looking for the total number of all activity per ID of grid square per type of activity, we can sum up all of the rows that have the same ID and time. In this way, we can reduce the number of rows as well as the number of missing values.

Detecting missed values is another important part of preprocessing. Most of the information in the current data had missing values due to either human mistakes or errors in precision of sensors. Figure 3 shows the proportion of missing values for a 1-day duration.

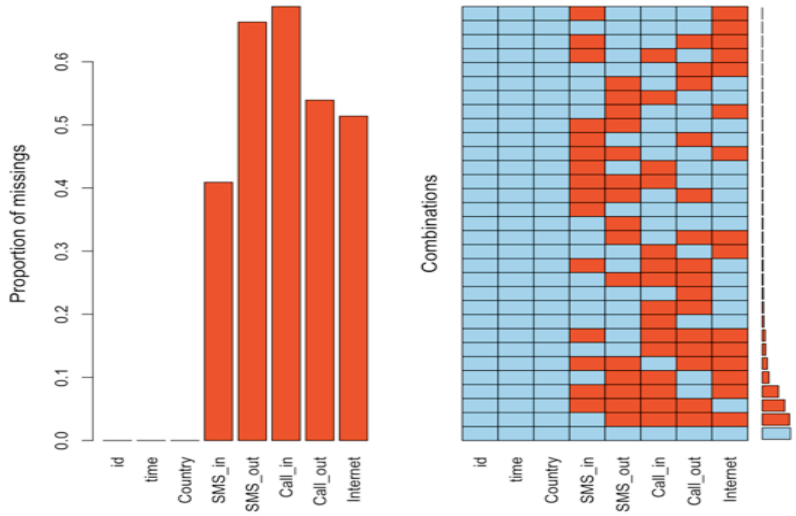


Figure 3. Proportion of Missing Values

There are several options to overcome this problem. The simplest way to deal with missing values in an activity column is to assign a numeric value equal to the average of all activities in the column. Figure 4 shows an example of a way to find missing values.

In Figure 4, on the top left is the original dataset, and by random, some of its rows have been removed. On the top, right the mean matching methods is represented and red circles show missing values have been found by mean matching. Another method is to use K-Nearest Neighbors (KNN). The bottom left of Figure 4 shows the KNN method. In this example, KNN has a worse prediction when compared to the mean matching method. Another famous method for solving the issue of missing values is Regression with Random Error. This method is using regression; however, it adds some noise to the data to predict missing values more accurately. For this dataset, none of the above methods was used. According to what was mentioned earlier about when there was no activity reported for a cell, sensors did not record any activity; Also, it appeared that there is no activity value reported as 0, but reported as *NA;* hence, we can easily change NA values to 0.

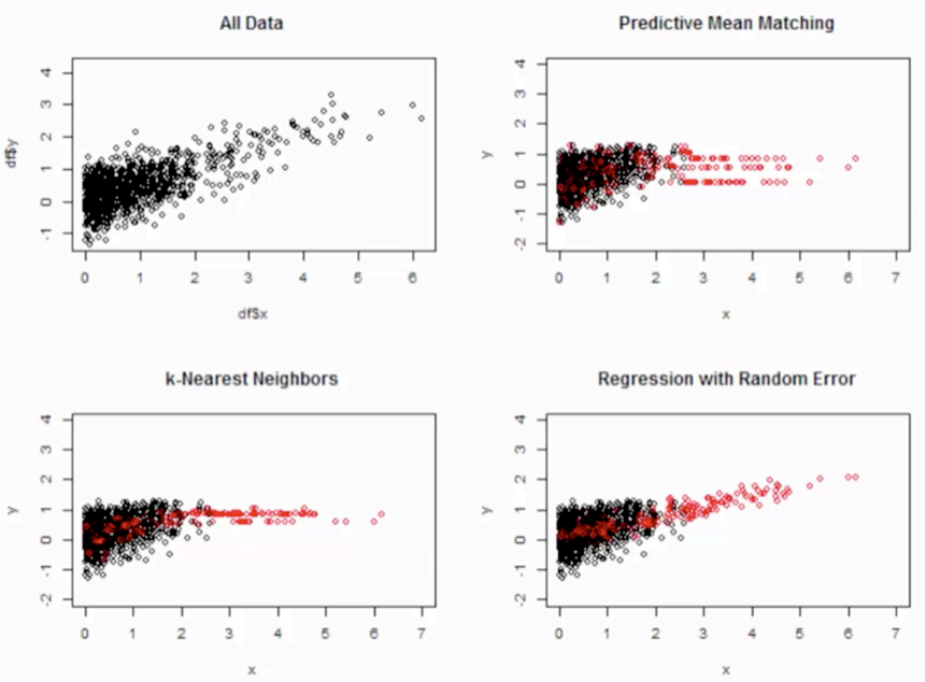


Figure 4. Methodology to Predict Missing Values

Another step-in preprocessing was finding outliers. In some cases, outliers can affect the results of machine learning algorithms to a substantial degree. However sometimes they might not. In our dataset, how we search for outliers matters because of the dataset's time series format. One way to analyze the dataset is to compare all the activity attributes of one day to find the outliers. Another way is to consider all the activity values for one attribute and compare that activity's attribute at a certain day of every week to find outliers. For instance, we can select the activity values captured for the Call-In attribute for every Friday in a month. Figure 5 shows SMS-out for two Fridays. These Fridays have different distributions, and the activity values are different in each hour for each day. However, none of these two methods can be used for detecting outliers due to spatial-temporal dataset.

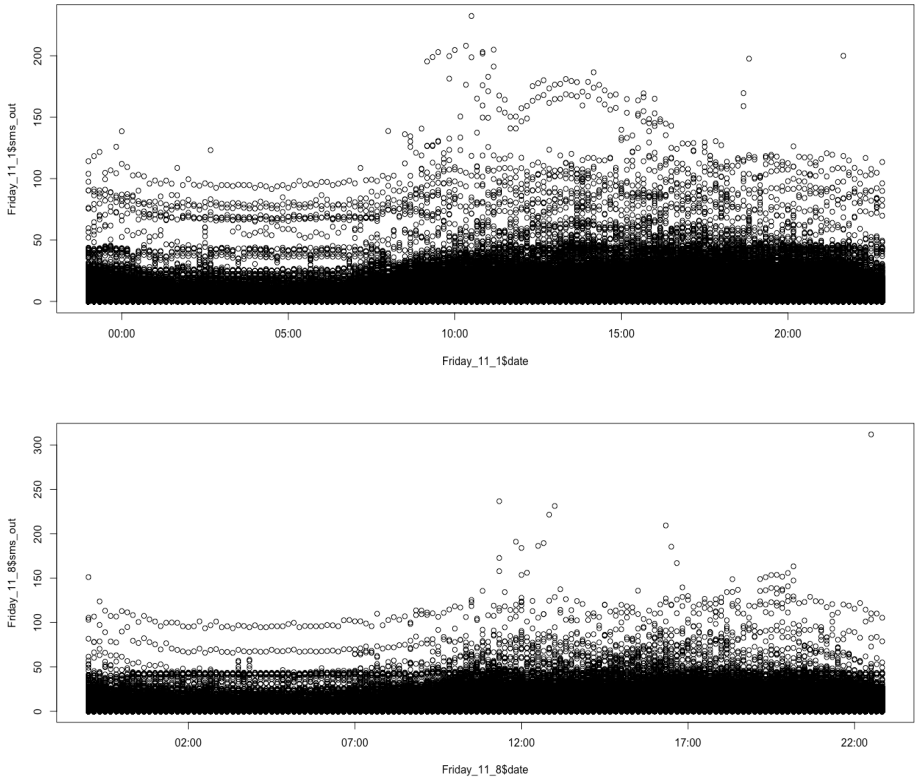


Figure 5. SMS-out for Two Fridays

The best methods to find outliers within activity values is to combine all activity values and find outliers in the combination of activities during 1 month. The outliers can show us abnormal activity during a day of a selected month for a specific location and time. Figure 6 shows all activities for a Friday, and Figure 7 shows the combination of all of the activities for all Fridays in the month. Each activity has a different distribution, but the combination of all activities gives almost the same distribution for the same day of all weeks. By using the discussed method, outliers can be detected in each day.

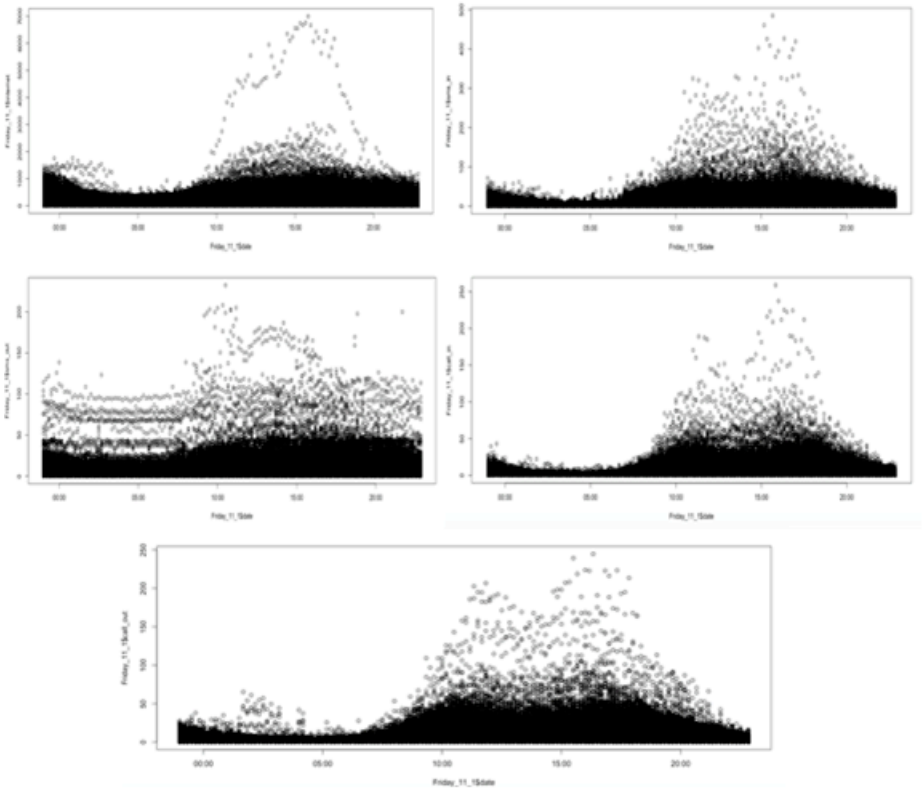


Figure 6. All Activities of a Friday

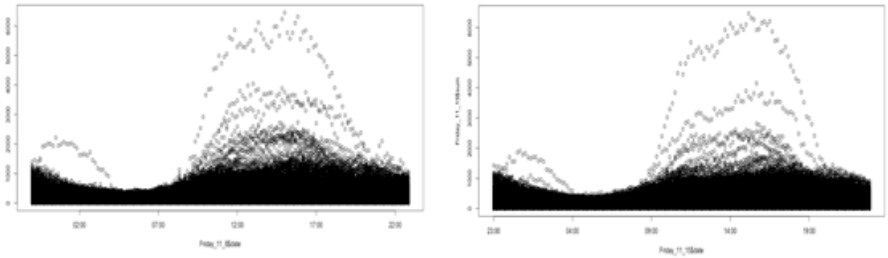


Figure 7. Combination of all Activities for two Fridays

As mentioned before, December was selected for implementing machine learning clustering. Figures 8 and 9 show an overview of all activities during that month. The figures show that the last week of December has several days of abnormal activities, probably due to holidays and its events.

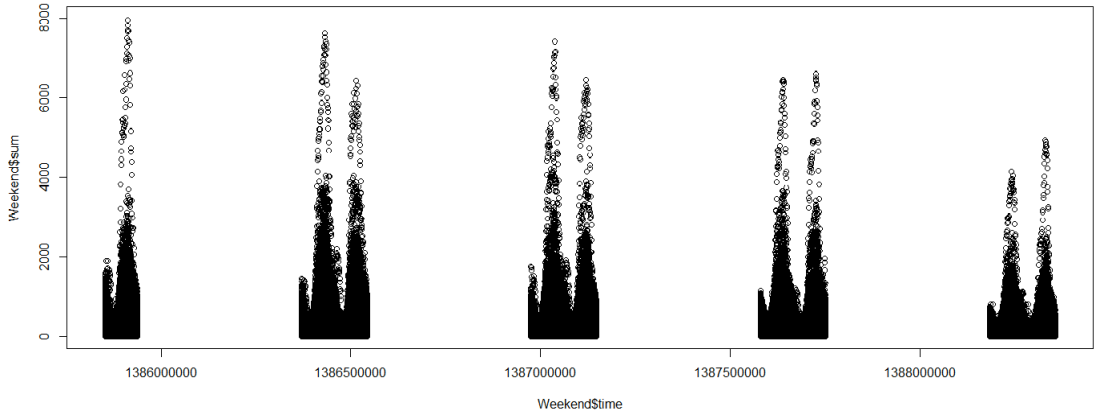


Figure 8. All Activities during Weekends in December

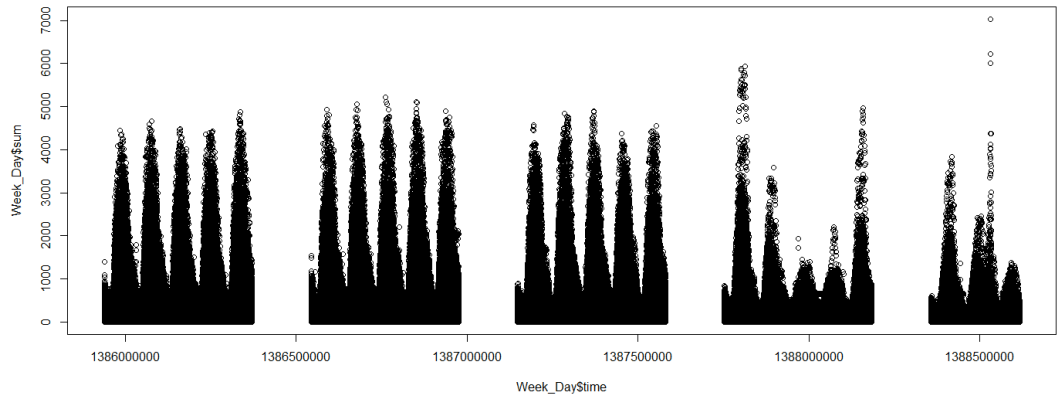


Figure 9. All Activities during Weekdays in December

Although the previous figures and tables show the concentration of events (described as attributes) at a specific time of day in a specific period, i.e., month. Now we wanted to add the element of location to the methodology.

To detect patterns of users' activities based on time and location, two steps must be completed. The first step is to determine the activity threshold for defining high, medium, and low activity. The second step is distributed the activity totals obtained in

the first step by time and location. There are some techniques such as the cumulative distribution to finding thresholds [9]. The problem for those techniques is they cannot give the exact threshold for the dataset because the activities' attributes do not have any scale for calculating thresholds. According to the Fig. 8 and Fig. 9, for our research, the number of activities for weekends are different from those for weekdays. Therefore, two thresholds for weekdays and two thresholds for weekends have been selected to show pattern of activities.

Weekdays

Hence for weekdays, we selected 13.1 as the threshold dividing low and medium activities, meaning fewer than13.1 events defined low activity We defined the threshold for medium activity as between 13.1 and 54.5 activity events. High weekday activity threshold was assigned a value of 54.5.

Weekends

For weekends, if there were more than 73 events of a specific type, the total was defined as high activity; between 25 and 73, medium activity; and fewer than 25 events, low activity.

To find the thresholds for weekdays and weekends for all attributes we changed the threshold values to find the best thresholds for all activity based on time and location. Table 2 and Fig. 11 have clear pattern of activities. (I don't understand why you change thresholds. The reason is not clearly communicated here.)

From these thresholds and by implementing the Markov chain, a pattern of activities emerged for weekdays and weekends. Each state in the Markov chain presented the activity that with the maximum number of square grids. Figure 10 shows

that for a weekday 0.25 of low activity changed to high number of medium activity and 0.75 of low activity staying on same activity. Also, for a weekday, medium activity by probability of one passing to another weekday, and same probability and activity for Friday to Saturday and Saturday to Sunday.

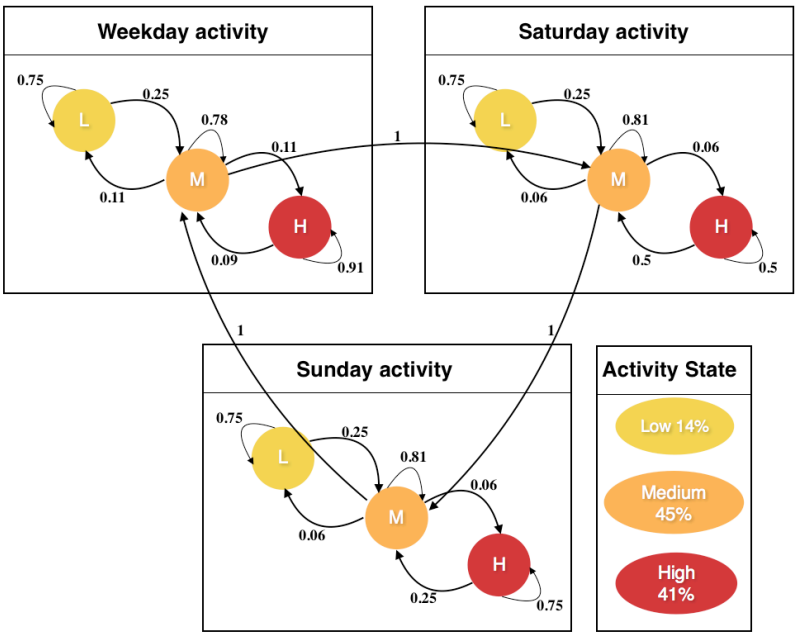


Figure 10. Markov Chain for December

The same pattern of traffic result was found for a 3-hour period on a weekday (see Table 2). For instance, from the table, it is obvious that the highest traffic values occurred from 8 a.m. until 7 p.m. The data also showed that the patterns of activity for Saturday and Sunday were totally different from the weekday activity.

Table 2. Maximum Level of Activity Per Hour

	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
00:00	M	M	M	M	M	M	M
01:00	M	M	M	M	M	M	M
02:00	L	L	M	L	L	L	L
03:00	L	L	L	L	L	L	L
04:00	L	L	L	L	L	L	L
05:00	M	L	L	M	M	M	M
06:00	M	M	L	M	M	M	M
07:00	M	M	M	M	M	M	M
08:00	H	M	M	H	H	H	H
09:00	H	M	M	H	H	H	H
10:00	H	H	M	H	H	H	H
11:00	H	H	M	H	H	H	H
12:00	H	M	M	H	H	H	H
13:00	H	M	M	H	H	H	H
14:00	H	M	M	H	H	H	H
15:00	H	M	H	H	H	H	H
16:00	H	M	H	H	H	H	H
17:00	H	H	H	H	H	H	H
18:00	H	H	H	H	H	H	H
19:00	H	M	M	H	H	H	H
20:00	M	M	M	M	M	M	M
21:00	M	M	M	M	M	M	M
22:00	M	M	M	M	M	M	M
23:00	M	M	M	M	M	M	M

The last result obtained from these thresholds is a heat-map showing telephone activity for the city of Milan.

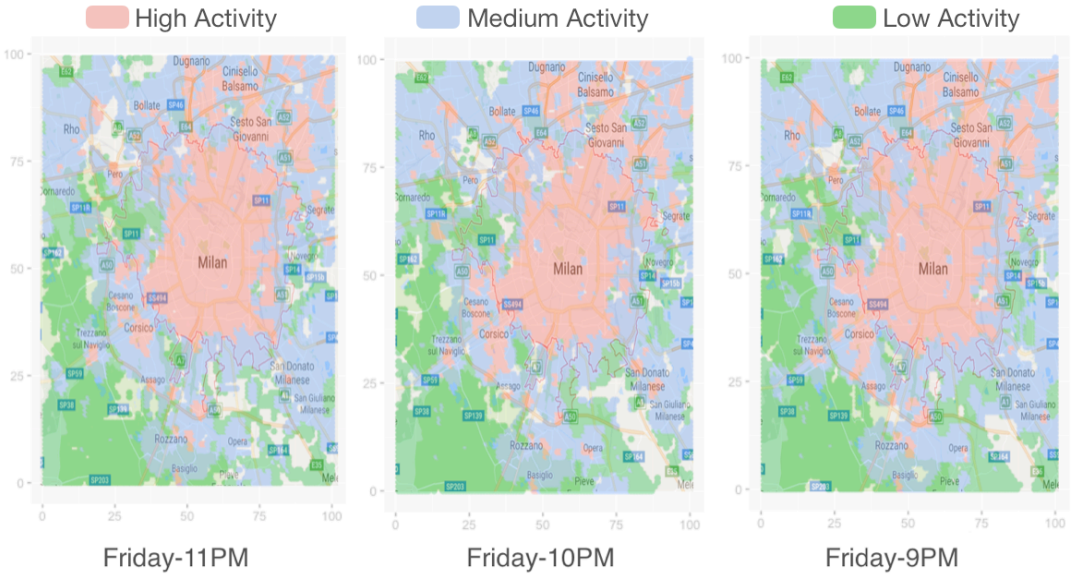


Figure 11. Milan City Heat-Map for Three Hours on a Friday

According to Fig. 11, we have high activity in downtown Milan. The figure also shows that the greater the distance from downtown Milan, the lower the concentration of call during the hours shown. The remaining calls, depicted by the blue color, represent medium level of activity.

According to what was mentioned earlier about the coverage area of mmWave cellular network, we selected the area with high activity density. Since, in this study, we selected an area of 20 by 20 cells in which most of the day, the area had the maximum high activity for a whole month supported by 100 by 100 cells.

This research has done base on parallel processing. Therefore, the methodology can apply to large numbers of grids equally well by adding more number of processers and systems in the cluster.

2.3 Algorithms

2.3.1 Converting Schema

Three steps we have done to convert each grid square to 47 bins from the selected dataset (Figure 12). First of all, detecting points of interest of each grid square. Then detecting streets and roads of each grid square. At last, combining all information from two previous steps with original dataset. In following sections are explained the algorithm to detect points of interest and important criteria about points of interest that must be considered and tools have been used.

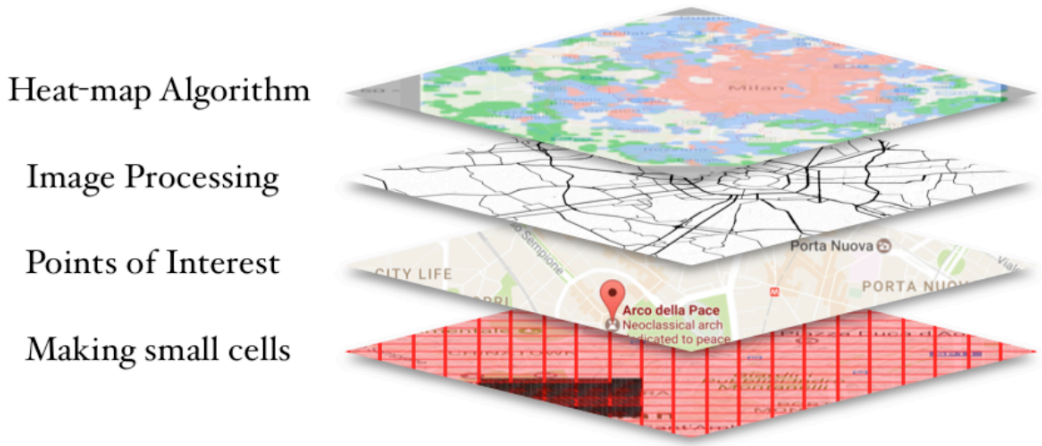


Figure 12. Heat-map and Three Steps for Selecting Points of Interest

2.3.2 Tools Used in Experiment

The details of all the tools used in this experiment including hardware and software for real and two virtual machines are as follows:

- Operating System for Real Machine: Windows 7
- Operating System for Virtual Machine: Linux Ubuntu 16.04
- Language: Python version 2
- Important Python Libraries: Pandas, Multiprocessing, Numpy, json, urllib
- Hardware used in this experiment setup was as below: Processor: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz RAM: 64GB
- Hardware used in this experiment as a virtual machine setup was as below: CPU: 1 Processors and 8 cores per processor, RAM: 16GB

2.3.3 Finding Points of Interest

A point of interest is a specific point that has an attraction for some group of people. For instance, shopping mall, coffee shop, park, parking, and etc. are some examples of points of interest. A point of interest is a location that in overall has more occupancy than a normal place. For implementing points of interest, 3 steps must be considered. First, detecting points of interest; Second, defining occupancy and area of the points; Third, finding working hour of each point.

There are lots of applications to find points of interest. For example, Trip Advisor, Expedia, and etc. are some of the applications to find points of interest but all of them have a problem that are limited to find various types for points of interest. Some of them just show Hotels and other lodging places or some of them have wider performance and they can show park and more locations but still they are limited for

variety. Google API is the best application that can show high number of types for points of interest. It detects 100 types of points, such as gas station, ATM, bank, hotel and etc. We have made an application by using python and Google API to find points of interest. The application gets maximum and minimum of latitude and longitude of the area, then the application gives a csv file that has GPS location of each points of interest with the type ID.

Table 3. Working Hours for Different Types of places

Type of Place	Hours(weekday)	Hours(weekend)	Type value
accounting	9am-7pm	closed	1
airport	-	-	2
amusement_park	8am-11pm	8am-11pm	3
aquarium	9am-5pm	9am-5pm	4
art_gallery	8am-7:30pm	8am-7:30pm	5
atm	8am-5pm	closed	6
bakery	7am-11pm	8am-11pm	7
bank	8am-5pm	closed	8
bar	12pm-1am	9am-7pm	9
beauty_salon	8:30am-9:30pm	10am-6pm	10
bicycle_store	9am-7pm	9am-7pm	11
book_store	10am-7pm	10am-7pm	12
bowling_alley	3pm-1am	2pm-2am	13
bus_station	-	-	14
cafe	7am-7pm	8am-12pm	15
campground	-	-	16
car_dealer	9am-7pm	9am-7pm	17
car_rental	8am-7pm	8am-12:30pm	18
car_repair	8am-6pm	8am-12:30pm	19

2.3.4 Finding Occupancy and area of a point

After finding all points of interest for the area, we need to find area and occupancy for each point. Most of countries has own measurement to calculate standard occupant density and area for different places. We used the information from Europe countries for this calculation. From the information, we figured out number of bins belong to a place and maximum number of occupancy for that specific point of interest. We have developed the application that gets the csv file from last function. It will check type ID of each location from the csv file with its source, which is predefined, then it makes a new csv file that has bins for over the selected places. Each bin has several attributes. One of the attribute shows the ID of bin, two attributes show GPS latitude and longitude, two attributes show ID latitude and longitude due to able match with the original dataset. The last attribute shows type of location that here we have more than 100 types of points of interest because each point of interest has different shadow layers around itself which depends on the type of point of interest it has half value of occupancy or one third value of occupancy. Also in some bins, we don't have any value, it means those bins do not belong to any either points of interest or shadows.

2.3.5 Finding Work Hour of each point

Finding work hour is the complex part of the application. It is important because it has effect on user density but it is complex because different neighborhoods and different type of point, has own work schedule. For better understanding, a branch of McDonald in a neighborhood, works 24 hours however another branch on same neighborhood works just from 8AM to 10PM. Therefore, for each location the application must check working hour. But there is no library and command for

programming language to detect working hour. Therefore, we defined a data frame as a source for the application that has average working hour for each type of points of interest. Figure 16 shows working hour table, the first column gives name of each type, second and third columns give working hour of each type for weekend and weekday and the last column gives type value of each location. In this part, the application just adds two columns to main data-frame to define working hour points of interest for weekend and weekday.

2.3.6 Image Processing

From Google API, just 100 types of places can be detected and these places mostly are detected from commercial and industrial area however still some other types of places have not been detected. Residential area, several commercial areas, private landscapes, and roads/streets are some example of points that the application has not detected.

We have made another function for the application to detect streets and roads and add them as a new type to the main data frame. The application by using JavaScript selects area and degree of details for the area. Degree of details can be used to define type of streets and roads; We want for analyzing. For instance, if we set zoom value as 11, the application will detect alley also but if we define zoom value as 8, it just detects streets bigger than alley street size. The application converts color of the Google map to black and white by using JavaScript (Figure 14) to prepare the map for image processing. Then the application takes a screenshot of the map by using python. It uses python image processing to read value of each pixel. If value of pixel is black it means

that pixel is part of a street or road, and it checks type of that location on main data frame; If the point on main data frame does not have any value for type, it will assign that location as a street otherwise it will leave the location and check another point.

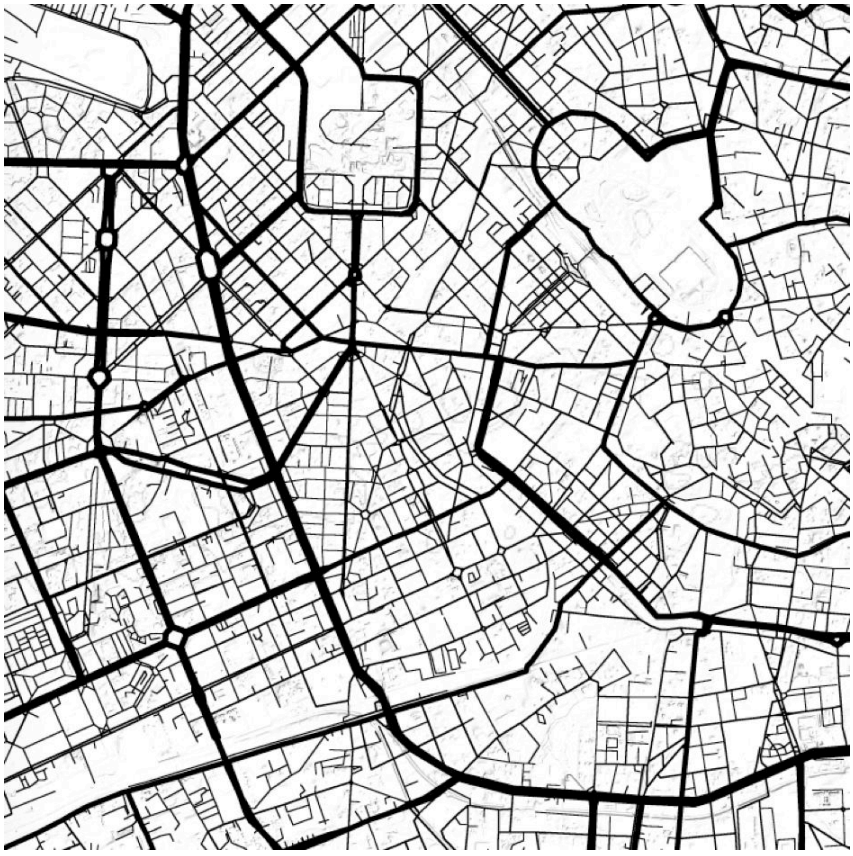


Figure 13. The selected area in black and white mode using JavaScript

2.3.7 Generating a New Dataset

After image processing and finding points of interest, now we have a data frame that has all bins for the selected area with type and hour for all bins and occupancy of each. There is one more function left which gets the original dataset and splits the dataset to different datasets regarding to hours. Then by combine information from a divided dataset and the main data frame, it generates a new dataset for each hour where instead of having grid squares we have bins.

2.4 Results

The application, after all previous processes, generates a new dataset that every bin has a new activity value regarding to combination of time, type of location, and activity from the original dataset. Figure 18 shows the last map after finding points of interest with shadows and streets and roads, light color on the heat-map shows high number of occupancy and dark color shows low number of occupancy.

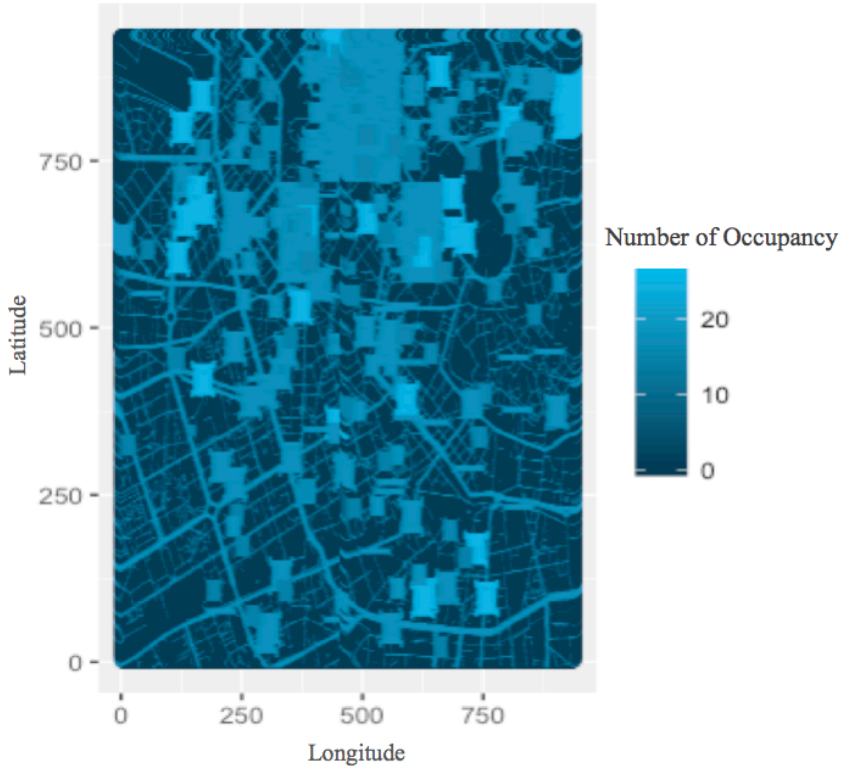


Figure 14. Heat-map of Occupancy, x is latitude and y is longitude

One of the property of this application that is developed by parallel processing for big data. generating just a grid square which is 235m×235m, takes around 2.67 seconds by using multi processors. However, same processes with a single process takes around 9.77 seconds. For generating the last dataset for each date, it takes around 39.47 minutes and same results with just a single processor takes around 130.251 minutes.

Chapter 3: Small-Cells Planning

One of the challenges for small-cells planning is, finding the best clustering algorithm. In this chapter, all conditions that a clustering method must have, is explained. Then regarding to those conditions, we provide our methodologies for planning small cells and in next chapter we discuss all results of these methodologies and select one of them as the best.

3.1 mm-wave small cells

Mobile network operators are looking for the best topology of planning small cells and the best topology cannot find easily because of several conditions of small cells.

In a cellular cell, there is different types of users such as student, employer and etc. [4]. Some of users are considered as users that are using mobiles and data more than other users. Also, some users are using data continuously with low download or uploading packets but some users are using data more than others but in a short time. Hence, mobile network has a complex structure because of different group of users.

Mobile network operators are looking for a technique that makes a small cell that has almost constant traffic load.

After analyzing different datasets such as CDRs dataset and finding number of clusters and information about each cluster, mathematic formula for physical part of antenna and cell must be implemented to plan small cells. For instance, calculating path-loss model and other elements. Therefore, shape of a cell is another important

parameter. Regular hexagon, irregular polygon, and fractal shape are some of the famous shapes for a mobile cell topology. For this study, irregular polygon is selected for shape of each cell for clustering to keep traffic load constant. One of the important criteria of irregular polygon and another cell network that they must be convex.

As mentioned earlier, the network coverage area for mmWave small cells is between 10m-100m. This limitation is one of the important challenges for planning small cell. It makes algorithm and calculation more complex because we have to consider those two previous conditions. In addition, it is not rational due to cost for installing and maintaining for mmWave to cover a whole city by mmWave network. Therefore, we need a threshold to reduce number of clusters for areas that have low number of activities.

CDRs and network data are time series data that show information about mobile activity for users during time period. According to [4], users in a cell are divided in different groups regarding to their type of activity. These groups in different time has different pattern of mobile activity. Also, these groups in some area has high density of members and in some other area has low density of members. All of these reasons cause that data activity for mobile network has spatio temporal pattern. By using optimization algorithm, we can find the optimum number and size of clusters.

3.2 Dataset and Data Preprocessing

The dataset is used in this part; it is retrieved from last chapter. It has 4 attributes which are latitude ID has range from 0 to 940, longitude ID has range from 0 to 940, time in Unix format, and traffic load has range from 0 to around 10,000. Each latitude and longitude ID present location of each bin that has area of 5m×5m on map. This

dataset shows internet activities of users in selected 22,090km^2 area of Milan city at December 2014. Figure 15 shows heat map of user activity for first hour of December. Light color on the plot shows high density of activities and dark color shows low density of activities.

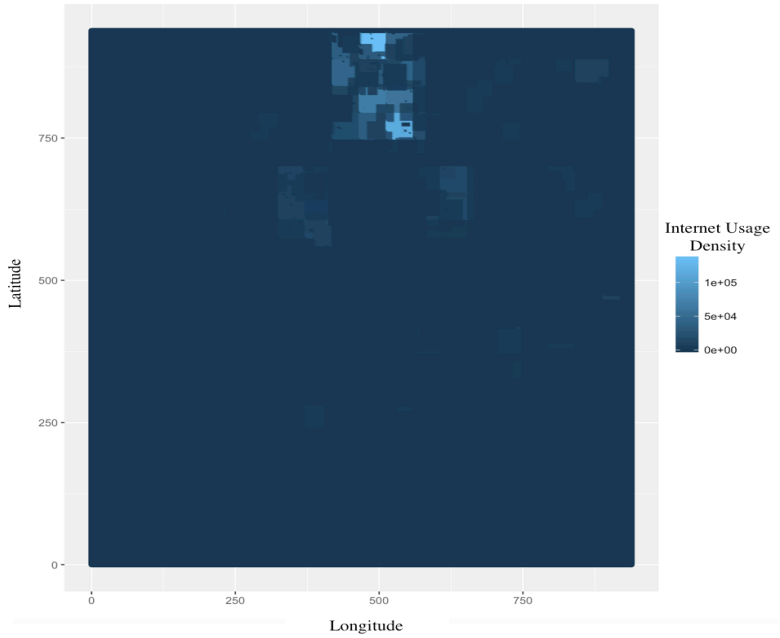


Figure 15. Internet Usage Density Heat-Map

According to figures 8, 9, and heat map (figure 11), we can see except abnormal activity at the end of December, there is no other abnormal activity for this month otherwise we have to eliminate those abnormal activities to get normal pattern of activities for using internet.

To find number of cluster and size of small cell for each we used weekdays to get a topology. According to table 2, All weekdays have same pattern. Therefore, we are looking the best topology for weekdays.

3.3 Related Work

As mentioned before, the purpose of this research is, to find the best method for clustering. When clustering techniques are mentioned Machine learning (ML) and Data Mining are two names that pupping up on mind. Clustering is one of the purpose of implementing Machine Learning and Data Mining. Since, for literature review is tried to find the best related research in this area. According to [6], there are several techniques that are used more than other techniques. Density Based by 53 percent has the highest number for is used in different papers, then receptively from high to low, we have k-means, Grid Based, Hierarchical Based, Bin-Packing Based and other methods. We have selected several algorithms as base methods for this research which are DBSCAN, K-means, and other algorithms that have been tried to develop regarding to the dataset. In following sections are explained the reasons to select these methods and briefly mention advantages of each method over than other methods. Then, in Methods part, we discuss more in details about each algorithm and in results and discussion chapter we bring our reasons to select the best algorithm.

3.3.1 Motivation to Choose DBSCAN Clustering

First of all, according to [6], more than 50 percent of research have used this method for clustering, since this reason it just a reasonable motivation to read and work on this method. Second, although processing time for this technique is more than other techniques, it has an accurate result to compare to other ML clustering techniques. It is

important for implementing the topology we have the highest accurate result due to cost and QoS for users [8]. Third, as can see from name of this algorithm, this algorithm is working base on density of area and we want to do clustering base on user activities which kind of users' density. Therefore, this is exactly same thing that in this study we are looking for, to get clusters regarding to density of geo location.

3.3.2 Motivation to Choose K-means Clustering

The first reason that causes K-means is elected as another method for clustering, same as DBSCAN this method is popular for clustering [6]. Although this method just has 17 percent of popularity, still this method has the second place of popular techniques for clustering. But this is not the main reason. The main reason is because this technique has the simplest algorithm to compare most of techniques. Therefore, it can do clustering faster than DBSCAN method. K-means is one of the popular technique for analyzing and clustering big dataset [14]. It can be used for analyzing real-time dataset.

3.3.3 Motivation to Choose Other Clustering

Regarding to dataset and the purpose for clustering, we used other clustering techniques. Moreover, we have made some other techniques by using idea and algorithm from other methods because for clustering this dataset, there are some criteria that must be included which are discussed more in section **3.4 Algorithms**.

One of the technique that used in this part is Hierarchical Based algorithm. This algorithm has algorithm as simple as K-means. Since, it is one of the motivation to

select this algorithm. Also, the base idea of this algorithm is splitting dataset regarding to similar properties that nodes of each cluster has. Also base on idea of this technique and splitting-merging technique, we have made another clustering which is explained on next part. There is one more method that was implemented base on vacuum cleaner challenge and motivation to select these new methods for this type of dataset to try bring new algorithms that have been never used for mobile network datasets [12].

3.4 Algorithms

In this section, we explain all algorithms are used in our study. We describe the reasons and method to select clusters for mmWave small cells. At last, we explain an optimization method and how we developed it for our dataset.

All of the algorithms in this part have been modified for the CDRs dataset and criteria for mmWave small cells. We select number of clusters for each method base on size of area and traffic load data of cells.

3.4.1 DBSCAN Clustering

Density Based Spatial or DBSCAN clustering is a method for clustering that is based on density of area. DBSCAN regions high density area from low density area. There are two parameters to the algorithm, MinPts and Eps. Eps is maximum radius of the neighborhood and MinPts is minimum number of points in the neighborhood of a point. There is an issue for this method which is selection method to assign value for two parameters. There are some techniques to find the parameters such as K^{th} nearest

neighbor (KNN) or by assigning a value for MinPts which it must be equal or bigger than number of dimensions of the dataset plus 1.

A cluster in DBSCAN has two set of points, core and non-core. Core samples can be found from other cores by calculating distance. A cluster also has a set of non-core that are neighbors of a core sample but they are not themselves core samples. DBSCAN has two advantages to compare to other methods. Firstly, it can find number of clusters without finding it. Secondly, DBSCAN can find clusters from noisy dataset.

From figure 15, two clusters low and high are shown for MinPts = 4. Left figure shows high density where point P is neighbor with three other points and all nodes in the cluster exceed the threshold which is 4. However, for right figure, for point q, the neighborhood is considered as low density because in the cluster with radius of Eps it does not exceed the threshold of MinPts.

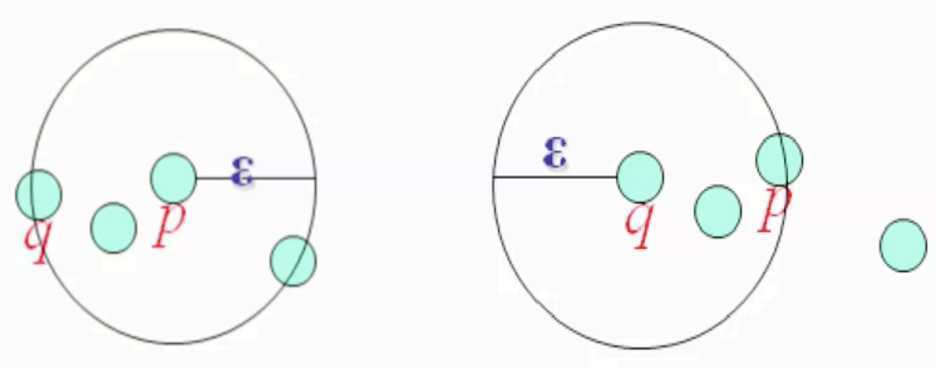


Figure 16. High and Low-Density Neighborhood

Figure 17 shows that how DBSCAN finds clusters for a dataset. The top-left plot shows dataset value. In the top-right plot, by using MinPts and Eps, outliers and border points and core points can be detected where green points are core points, blue points are borders and outliers are shown by red. In the third plot outliers are a cluster, and other clusters are defined by using clustering method and density connectivity. Two

points are density-connected when they they are commonly density-reachable from a point.

In our study, we implemented DBSCAN on an hour of dataset to compare the results with other clustering methods. We select range of radius for different values to check which parameters gives us the best result to compare with other methods.

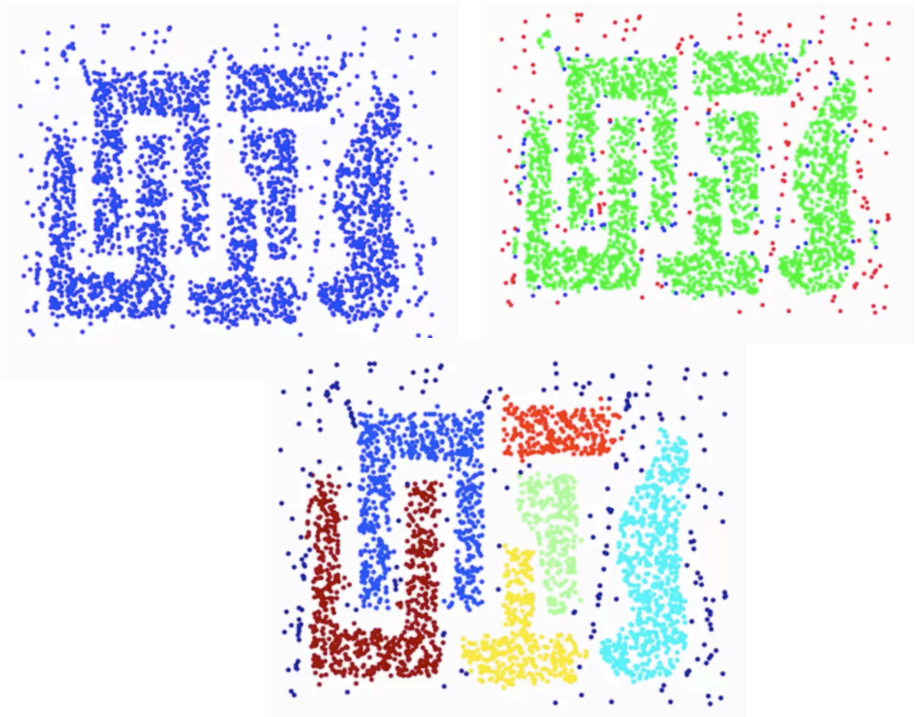


Figure 17. DBSCAN Clustering Steps

We select the best cluster where it has low load variance and low number of clusters. For MinPts, we selected 16 because the minimum number of nodes that each cluster can have is 16 bins.

3.4.2 Hierarchical Based Clustering

In this method, algorithm starts with n clusters where n is number of objects in data. In other word, each node in a data is a single cluster. Then it is reducing the number of clusters by joining those two clusters that are most similar to each other. This step is repeating until there is only one cluster left which is all objects of data.

There are different techniques for implementing Hierarchical Based Clustering: Centroid, it is clustering base on distance between the centroids of the two clusters. Average Linkage, it is averaging dissimilarity between all pairs of two points. Single Linkage, it is clustering by finding dissimilarity between the two most similar data. Complete Linkage, it is based on dissimilarity between the two most dissimilar data.

For our study, we used Hierarchical Based distance matrix clustering. We did not find number of cluster base on a tree diagram. As mentioned earlier a cluster has several conditions. Therefore, we implement our algorithm to limit size, load, and shape of cluster base on our parameters and then check conditions for each number of cluster till it exceeds the conditions.

3.4.3 K-means Clustering

K-means is one of the fastest method for clustering. Because of this ability, it is being used for large number of dataset. One of the weakness of k-means is to find

number of clusters k. However, we modified this clustering methods due to conditional small cells we are looking for. K-means algorithm is divided in two sections:

Firstly, find member of each cluster.

Secondly, find number of k clusters.

To find member of each cluster, if the purpose to find two clusters. First of all, two points as a center of each cluster, called centroid, must be selected by random value (Figure 18.b). Then each node of dataset is compared with these two centroids and the closest centroid takes as the center of the cluster that the node belongs to (Figure 18.c). After these steps, center of each cluster must be found and then again compare distance of each node with selected centers (Figure 18.d). Last two steps are running until those two centers are not moving anymore. K-means is described by algorithm in figure 19.

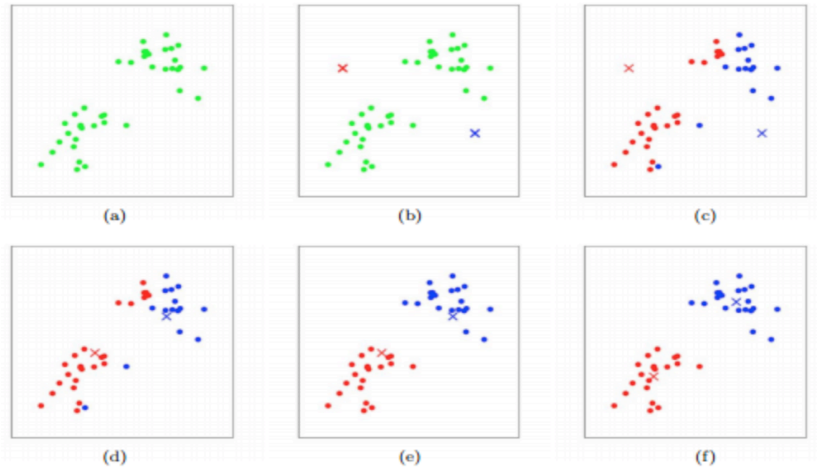


Figure 18. K-means clustering process [11]

Algorithm K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

Figure 19. K-means Algorithm [8]

One of the weakness of k-means is to select start point for centroid. The method to reduce the effect of this weakness is to run this algorithm with different value of centroid. In our algorithm, we ran k-means for 100 times to find the best members of cluster.

As mentioned earlier for other clustering algorithm, we modified each clustering techniques to find number of clusters. Although the technique to find k number of clusters has not been used to find k for k-means clustering, this technique has been selected to select areas with high density. In k-means clustering, elbow method is used

$$\text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \qquad \textbf{Equation 1}$$

to find k number of clusters. K-means is running for k = 1 till k = number of nodes or half of number of nodes. For each k, Sum Squared Error is calculated (SSE). Equation 1 shows how to calculate SSE. In the equation k is number of cluster, C_i is center of cluster i^{th}, and x is member of cluster i^{th}.

Figure 20 shows SSE value for k from 1 to 6. By using Elbow method, we select a point that has minimum number of cluster and minimum SSE value where k = 2 is the best value for this example because from k = 2 to k = 6, SSE does not have drastic change. There is a problem for Elbow method that in some cases it is not easy to detect a point for k.

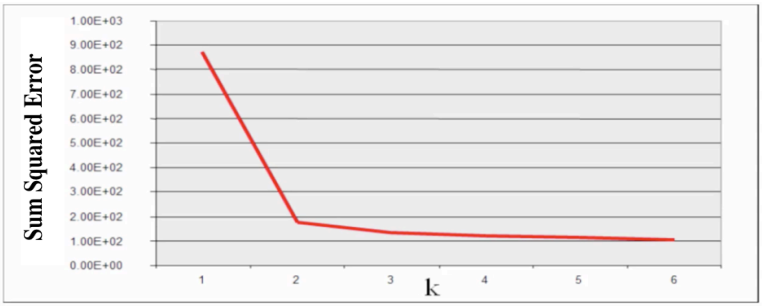


Figure 20. Elbow Method to Find the Best SSE

3.4.4 Other Clustering Methods

Nowadays, mostly Machine Learning (ML) techniques are using for clustering. In this section, we explain some new techniques for clustering that we have developed regarding to dataset and conditions we have for implementing mmWave small cells. There are two clustering methods that we have developed. However, the base idea for these two techniques are coming from ML but they have not been used for telecommunication.

First technique is Split-Merge technique. In this technique, we used idea from post-processing for ML. There is a technique for k-means to reduce SSE value. In k-means after finding number of clusters and members of clusters, to reduce SSE value, first we must find a cluster that has the highest SSE value. Then by splitting that cluster in two clusters we reduce the SSE value. After splitting there are two techniques for complete post-processing. By removing the centroid of the split clusters and reassigning the points to other clusters, we can reduce number of clusters and SSE. Another technique, by finding closest centroid to the split clusters merge two clusters with other clusters. We have made a new technique for clustering by merging split-merge technique and Hierarchical Based Clustering.

In our technique, we consider all data as a cluster. Then by splitting them into two other clusters we reduce value of load of the cluster. Splitting is either vertical or horizontal to have convex clusters. We repeat this process for cluster that has the highest value. This technique is opposite of Hierarchical Based clustering. Splitting threshold is defined where two cluster has the closet load. By this technique, at the end we will have clusters that they have same load activity.

There is another method that we have made it from vacuum cleaner problem. Vacuum Cleaner problem is one of the famous challenge for Artificial Intelligence (AI). AI is looking for the most optimize technique that a cleaner clean area that has dust in the fastest time and shortest pattern. There are several famous techniques to clean area that has dust. Figure 21, shows some of the techniques but one of the famous techniques is come from ML techniques which is Reinforcement technique.

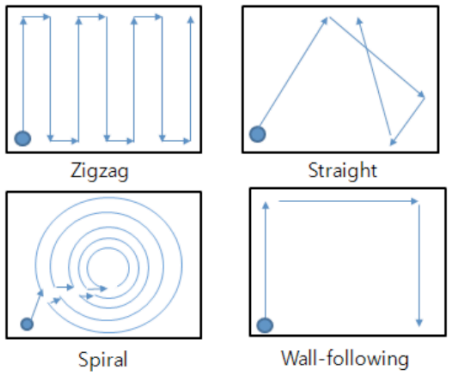


Figure 21. Famous Cleaning Pattern for Vacuum cleaner robot [12]

One of the Reinforcement example is a robot starts from start point and it check all neighbor cells of cell that robot is in. It selects a neighbor cell that either has highest value or same value. It continues until reaches the final point which has the highest value. Then from that point it is valuing other cells. This problem is for vacuum cleaner robot when it does not have a map for dust density. However, in our problem, we have all value of cell. In our technique, the algorithm selects a corner point for start where it is defined as a bin that has lowest ID and it compares all neighbor bins for selected ID. The algorithm adds neighbors from the highest traffic load *data* till lowest traffic load *data*. The algorithm is continuing this step until the selected cluster exceeds either traffic load *data*. Then the algorithm continues clustering for another cluster. These

steps are continuing until we clustering all bins in the dataset. The algorithm is described by algorithm in figure 22.

As mentioned earlier number of bins are in this dataset is 883,600. Moreover, each cluster has maximum 400 bins. Therefore, minimum number of cluster than we can have is 2,209. However, in some area again we do not have high activity from figure 15. Therefore, by using Elbow method (figure 20), we try to find minimum number of cluster where we have low variance for traffic load data of clusters. In addition, we will have figure 20 but instead of SSE we will have variance for traffic load data of all clusters.

Algorithm Vacuum Cleaner.

```
1: repeat
2:    Select the lowest unclustered ID and add as a new cluster.
3:    repeat
4:       Find neighbor bins of the cluster.
5:       Add neighbor bins from the biggest load till the lowest load to the cluster.
6:    until Size of the cluster or traffic load exceeds
7: until No more unclustered ID is left.
```

Figure 22. Vacuum Cleaner Algorithm

3.5 Optimization

Our goal of clustering from previous to find number of clusters and area the clusters must be located for mmWave small cells. But the final result that we are looking for, is optimal number of clusters and optimal area of the clusters for any hour of a week. Therefore, in this section we discuss about optimization to solve the problem. There are different algorithms for implementing optimization but one of the famous algorithm is Ant Colony optimization.

Ant colony optimization algorithm is a probabilistic technique for solving mathematical calculation problems related to time series which can be reduced to finding good paths.

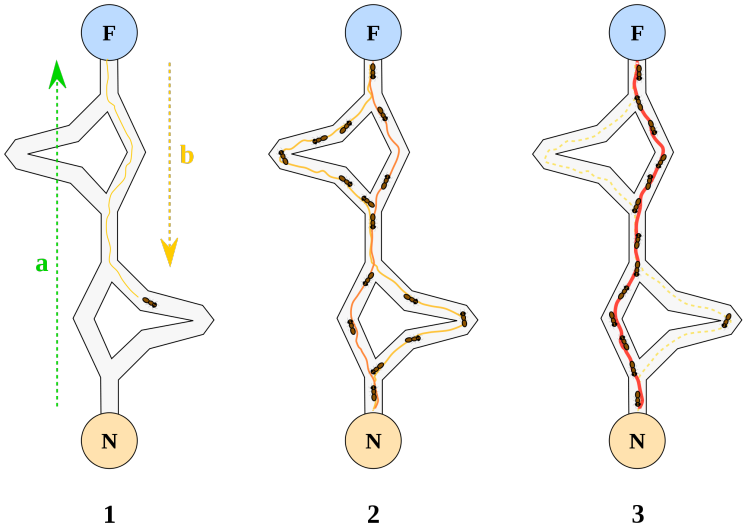


Figure 23. Ant Colony Algorithms [13]

According to figure 23.1, There are some ways between F and N where some ants try to go from point F to N and come back. We try to come up that how ants find the fastest way. In this example, the ants are going immediately after each other. If the first ant selects to go from yellow pattern, it takes T_1 to get N. Other ants try to go from new way until there is no newer way. In this example, delay time from moving time for the first ant and others are very small because of that we do not consider that time. When the first ant is arrived if it is the first ant that arrives to point N it means this way is the fastest but as can see in the figure the path for the first ant is not the fastest. Therefore, the first ant will come back from the path of the ant which arrived earliest and it well select path of ant that has smallest value of T. After some time, all ants will figure out the fastest way from ants that they went same time and arrived earlier or from

the previous ants. On Figure 23.3, all ants select the fastest way and every other time some ants randomly select different time to check the path again.

For this study, we use combination of Ant Colony and Reinforcement which discussed in section 3.4.4 for optimization. Ant Colony algorithm is used for checking the clustering result for each hour and Reinforcement algorithm is used to modify the clustering result for any hour. We assume that figure 24.a shows a result for the best clustering method. The algorithm uses the same clusters for different hour which shows in figure 24.b. The traffic load data for figure 24.b is different from 24.a and if the variance load data for figure 24.b is less than 24.a it means the clustering result is good, otherwise the clustering must be changed.

Now, the algorithm selects the lowest traffic load data cluster and check all neighbor clusters. It selects a neighbor cluster which has the biggest traffic and/or it is bigger than the limitation traffic plus error. Then it tries to remove bins from the biggest cluster and add those bins to selected cluster. The algorithm repeats this step until the selected cluster area exceeds the limitation or passes the limitation for traffic or there is another neighbor that has the biggest traffic load. If the selected cluster achieves the

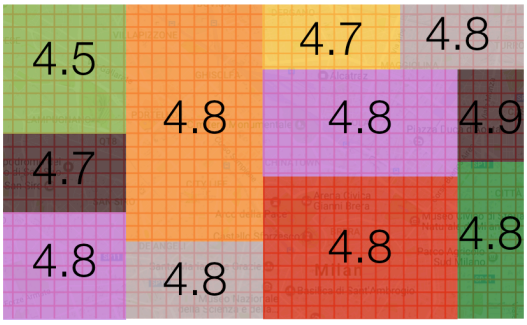


Figure 24.a Result for the best clustering algorithm for hour_0

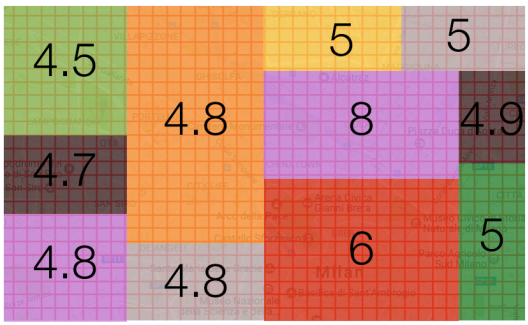


Figure 25.b Same clustering result from 24.a for hour_4

minimum traffic limitation or limitation for the area first the algorithm check shape of the cluster and if it's not convex it tries by add or remove bins make the shape convex. If there are gaps between bins in a cluster means the shape is not convex therefore by adding bins from the highest cluster or adding bins to lowest cluster, makes the cluster convex. Then the algorithm will select another cluster that has the smallest value. Else, the algorithm will select another neighbor that has the biggest traffic load. The algorithm continues these steps until it exceeds the limitation for minimum and maximum traffic load or the changes for variance it is less than beta.

Algorithm Optimization algorithm.

```
1: for all cluster
2:    Select the lowest traffic load that is unchecked.
3:    repeat
4:       if the max traffic load neighbor cluster is more than high threshold or selected cluster is less than low threshold then
5:          Remove bins from the neighbor and add them to the cluster
6:          Implement Convex function
7:       else
8:         End repetition.
9:    until Size of the cluster exceeds the limitation.
10: end for
```

Figure 26. Optimization Algorithm [8]

Chapter 4: Results and Discussion

4.1 Results

In this chapter, we describe the results for each method. Section 4.2 discusses and compared all methods and states reasons for choosing an algorithm as the best method.

Table 4. Normal DBSCAN Results

Number of cluster	Max traffic load	Max length	Max bins	eps	MinPts
610	159074178.86	939	528781	20	16
682	159074178.86	939	486516	15	16
813	159074178.86	939	268986	10	16
967	159074178.86	967	132888	5	16

Table 4 shows results for DBSCAN algorithm for different values of eps and MinPts. We selected different eps to find the best distance value less than 40 bins, which is equal to 200 meters. Distance shows the number of bins between either maximum and minimum latitude or maximum and minimum longitude. We selected 40 bins because they cover an area with a radius of 100m. MinPts value of 16 was selected because the minimum length of a cell is 10 meters since, cells cannot have number bins less than 16. Figure 27 shows DBSCAN clusters when eps=20 and MinPts=16. We found maximum threshold of 2173515 from the smallest cell that had the highest density of users.

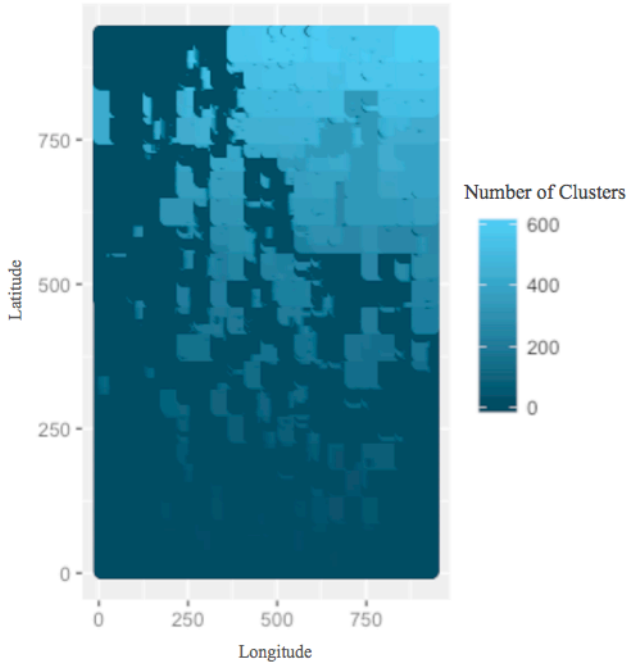


Figure 27. DBSCAN with eps=20, MinPts=16

Table 5. Hierarchical Based Algorithm for 10000 bins

Number of cluster	Max load traffic	Max length
10	371326	100
50	36243	79
100	22055	52
200	14179	35

Hierarchical-Based clustering is the second algorithm that we have checked. This method was not able to manage large sized datasets. However, we have tried it on a sample of the dataset. The sample dataset had 10,000 bins, and a maximum traffic load limitation for this dataset was 12558 events. The maximum length this algorithm got for 200 clusters was less than the maximum length of a cluster. The maximum cluster traffic load was close to the threshold (Table 5). Figure 28 shows a pattern of 200 clusters.

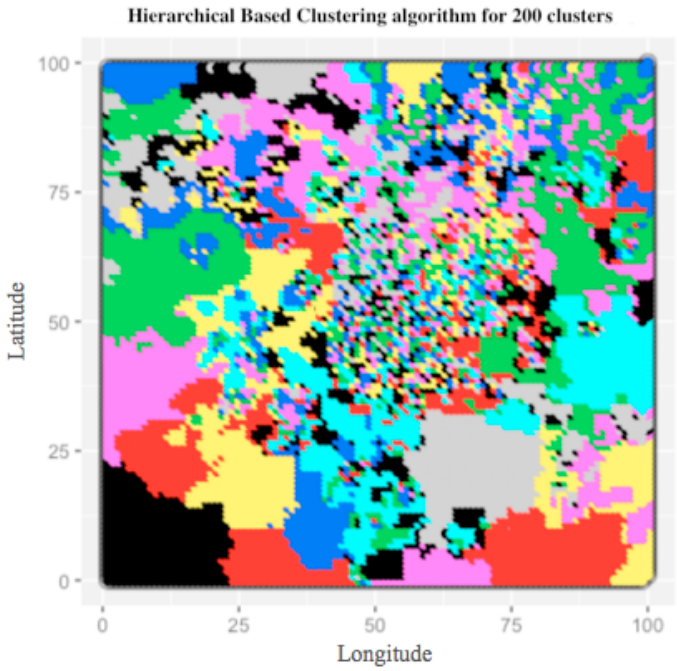


Figure 28. Hierarchical algorithm for 200 clusters

As mentioned earlier in chapter 3, K-means is able to manage large amounts of datasets. Table 6 shows the distance and traffic load for 10, 100, 1000 clusters in K-means. After running the K-means algorithm for k = 1000 clusters, the maximum traffic load and distance was bigger than the conditions (Table 6).

Table 6. K-means algorithm results

Number of cluster	Max traffic load	Max length
10	325430773	938
100	300867193	808
1000	159074188	477

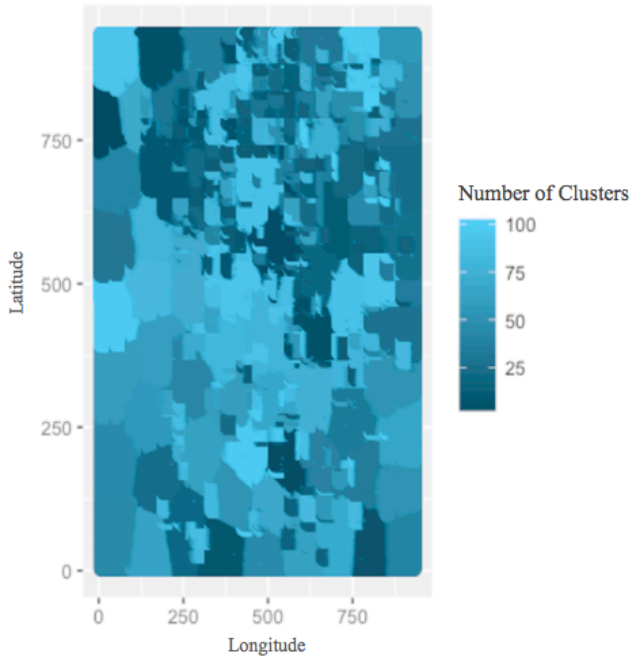


Figure 29. K-means algorithm for 100 clusters

Figure 29 shows how K-means did clustering for the original dataset. We used a statistic technique to change properties of a parameter of the dataset to get other results for the K-means algorithm. We scaled the traffic load parameter to 0–100. Therefore, the traffic load threshold/limitation also changed. Table 7 shows the results from the scaled K-means after implementing the technique on the dataset.

Table 7. Scaled k-means algorithm results

Number of cluster	Max traffic load	Max length
10	653853.3	389
100	274417.3	125
1000	117100	50
2000	73500	45
3000	31400	45
3500	18538.72	30

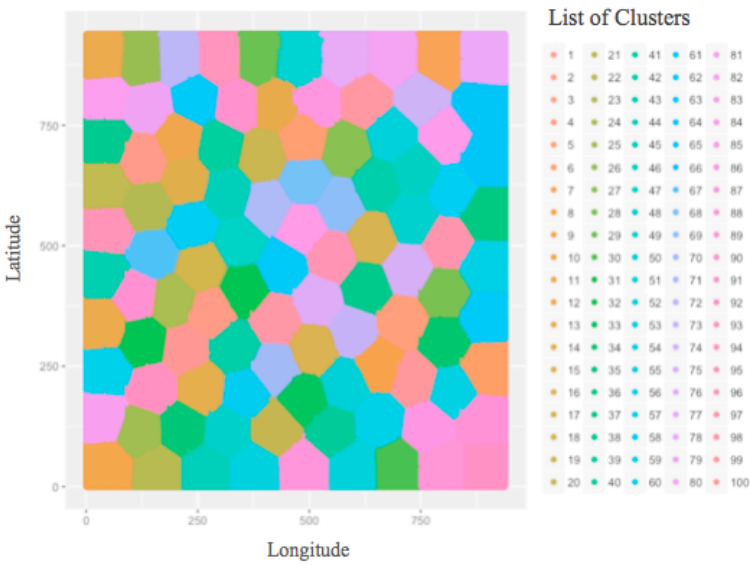


Figure 30. Scaled K-means clustering for k = 100

There are two more methods left that we implemented for mmWave small cells planning. As mentioned earlier, one of them is the split-merge technique. We ran this technique on a sample of the dataset to check the effect of it on the sample then deployed it on the main dataset. Figure 31 shows the result before implementing on whole dataset. Split-merge clustered mostly base on longitude. Therefore, the most of clusters have coverage length longer than the limitation.

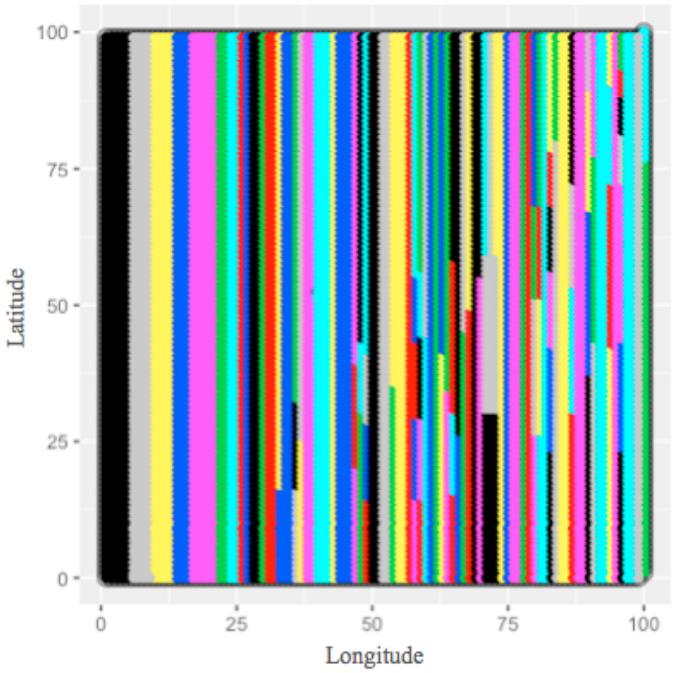


Figure 31. Split-Merge Clustering

The last method is vacuum cleaner method. We implemented this technique on a sample of the dataset. Figure 32 shows the vacuum cleaner method when id = 1 is initial Id for clustering and Id = 10,000 is the last id of clustering. The vacuum cleaner method found 284 number of clusters which following all conditions for mmWave small cells except convexity.

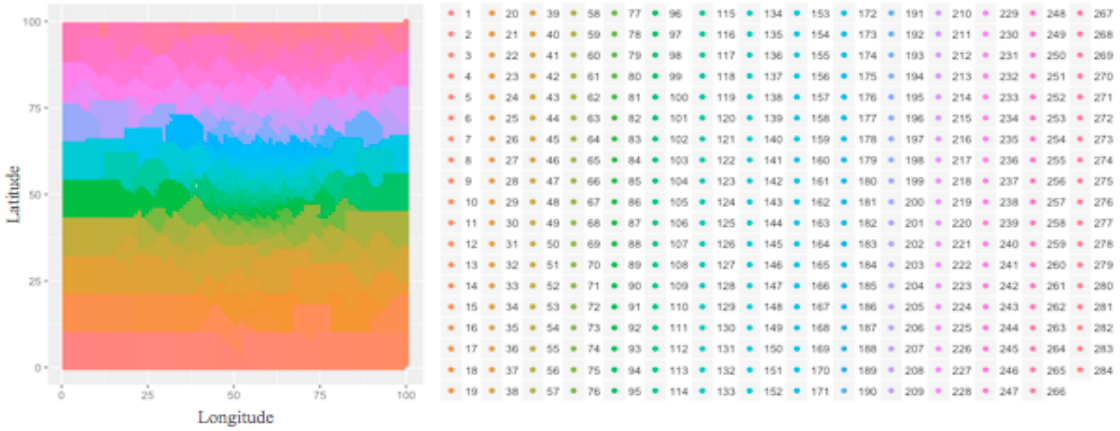


Figure 32. Vacuum cleaner algorithm clusters

We developed vacuum cleaner method by solving the convexity problem. In second version of vacuum cleaner we changed initial Id from Id = 1 to Id that had highest activity value. We add a function to check convexity and make clusters convex. Figure 33 shows clusters for this method where each color shows area of each cluster.

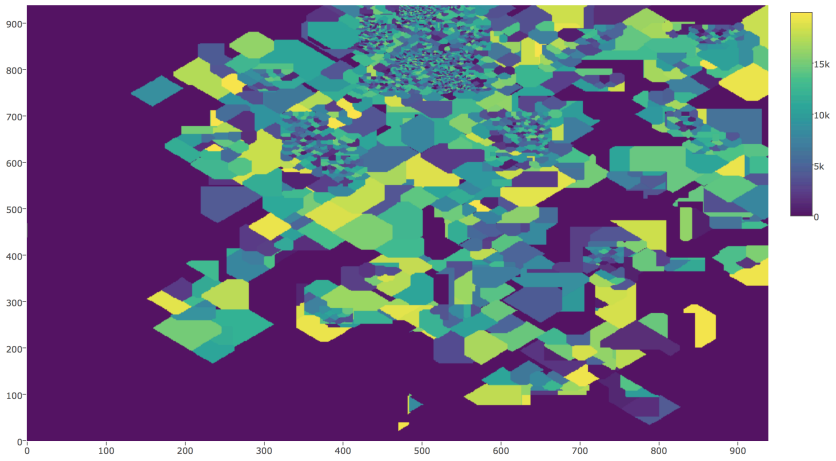


Figure 33. Vacuum cleaner algorithm clusters V.2

4.2 Discussion

We elected to use Vacuum cleaner algorithm for clustering. DBSCAN has two main problems. First of all, according to table 4, the algorithm detected high number of bins as noises and considered all of them as one cluster. Therefore, results from DBSCAN show Max constant traffic load and Max length for different eps. Secondly, we are looking for max traffic load around 2173515 and length of a cell 40 but Max traffic load for DBSCAN was 75 times more than desire traffic load likewise maximum length of cells was 23 times more than desire length.

Although, Hierarchical Based Clustering gave us max clusters length less than threshold and traffic load closed to the load threshold, it had two main problems. Firstly, the algorithm cannot manage large amount of dataset. Secondly, although maximum load

traffic and length were less than limitation, there are lots of cells that they have bins less than 16, so those clusters and nodes need another algorithm to manage all of them (Figure 28).

However, K-means manages large amount of data but this algorithm has a hidden problem regarding to max length. By looking to figure 29 it is not clear that there are cells with length higher than 477. This problem is happened because k-means algorithm is based on distance. In generated dataset, traffic load has high effect on our result because range of it is between around 0 to 135844. However, cell ID is between 1 to 940. Therefore, k-means is clustering the dataset based on load traffic. We scaled traffic load parameter to solve the problem. Then we repeated k-means algorithm with scaled dataset. According to table 7, for 3500 number of clusters the maximum length is less than 40 bins and traffic load is 18538 which is close to our threshold.

Split-Merge clustering method, gives almost equal traffic load for each cluster but this structure is not practical due to length of each cell. Length of each cell is as same as range for either latitude or longitude because this algorithm tries to split a cluster to two clusters based on which two clusters have more similar traffic load.

The last algorithm is for Vacuum cleaner algorithm. This method gives almost the best number clusters, the lowest variance traffic load for all cells. Vacuum cleaner algorithm has a main issue that is not time efficiency. We used vacuum cleaner algorithm because we are looking for accurate result for deploying small cells.

The result from vacuum cleaner algorithm and ant colony optimization show for selected area, 2097 mmWave cells, 424 small cells and 25 macro cells are required. According to Figure 34, yellow part is covered with macro cells, dark green is covered

with small cells, and light green is covered with mmWave. From whole area that is covered with 25 macro cells (22,090 sq. km), small cells cover 11,276.3 sq. km, and mmWave cells cover 1,853.28 sq. km.

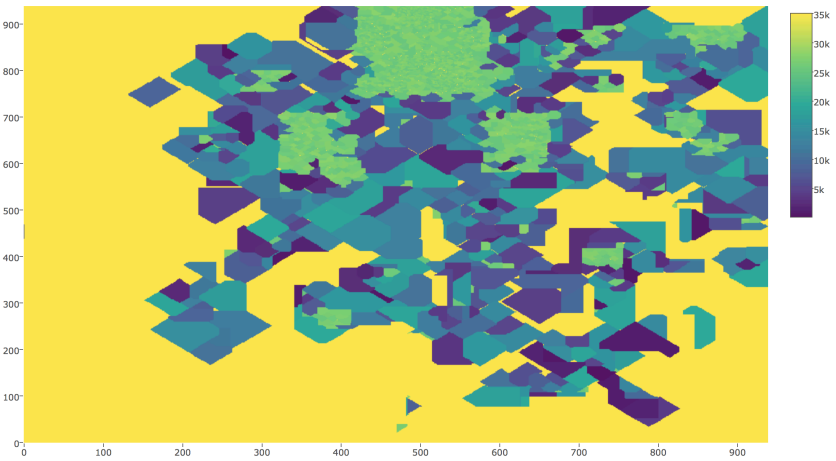


Figure 34. A Detail Map of Small and Macro Cells

Figure 35 shows the histogram of traffic load after optimization for initial time hour. Some cells have traffic load less than the defined boundary because in different hours we have different pattern of activity and some cells have traffic load more than boundary and by adding more macro cells we can manage the traffic load.

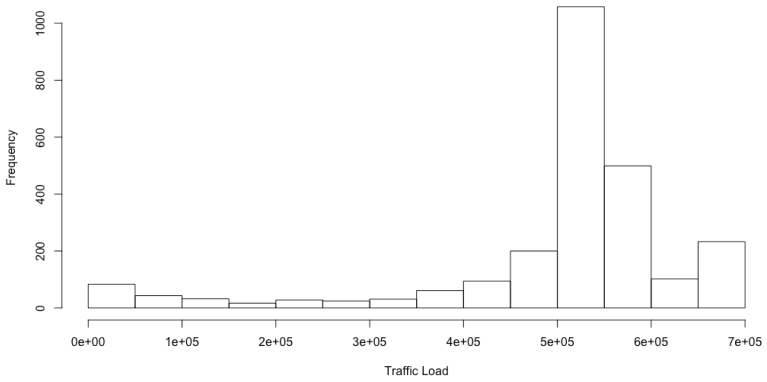


Figure 35. Histogram of Traffic Load

Chapter 5: Conclusion and Future Work

The primary purpose of this study was to mine mobile network data to improve QoS, capacity, and coverage of networks by implementing mmWave small cells for existing macro-cell networks. We focused on call detail records, real data that was generated every time a user used the mobile network for different types of activity such as accessing the internet and for sending or receiving short message services (SMSs) to detect meaningful user patterns of activity.

In the beginning, machine learning (ML) and data mining techniques helped to approach the desired goal. The first achievement of this study was to detect patterns of mobile activities for the entire city of Milan, Italy. By implementing a Markov chain and some data preprocessing techniques of data mining, a basic pattern of activities was detected. Not only can mobile network operators benefit from knowing these patterns, but other businesses and industries also can gain a better understanding of their customers' behavior regarding their product or service. We created a heat-map and table hours of users' activities during a day. The heat-map and table hours show classifications of user activities. From this information, we selected high activity areas, i.e., those having 20-by-20 number of cells for two reasons. First, the purpose of small cell networks is to compensate for weak capacity of macro cell networks during periods of high user activity. Second, a large dataset demands a cluster of servers for processing and generating results.

There is imperfection limitation when implementing mmWave small cells in some CDR datasets. The CDR dataset is a collection of users' information gathered during their use of the macro cell. However, the collection area of a cell in macro cell

networks is different from that of a cell in a small cell network. Therefore, there is a gap, scale of dataset, for achieving the desired goal. To overcome this limitation, we developed an algorithm to scale the CDR dataset. The algorithm generated a new dataset by using the, maximum and minimum of latitude and longitude of the CDR dataset. In this study, the algorithm converted a big cell with an area 235m × 235m to 2209 small bins with and area of 5m×5m.

Finally, to accomplish the desired goal, deploying small cells at mmWave frequencies we compared five ML techniques to determine which was the best. We compared different clustering methods such as K-means, DBSCAN. scaled K-means, and the vacuum cleaner technique to identify the best algorithm.

Vacuum cleaner algorithm had the accurate result to compare to other methods. However, the time of processing for this method was more than other methods. After implementing vacuum cleaner clustering and ant colony optimization, 25 macro cells were found to cover the whole selected area, 424 small cells were found to cover 11,276.3 sq. km, and 2097 mmWave cells were found to cover 1,853.28 sq. km. From this number of small cells more than 617 cells are temporary cells which different hours depends on traffic must be on or off.

There are two main objectives that we want to complete in the future. First, in this study, we performed clustering for normal day activity. However, during a year, there are many abnormal activities. For instance, in November, there were three important football games that contributed to a spike in cell activity in our test location. In addition, the second important object has two parts. One, is to improve our algorithm

for running on a cluster. Currently it runs on parallel processors, but it needs to run on a cluster of processors due to dataset's large size.

References

1. Wearable technology and the internet of things. https://www.ericsson.com/networked-society/trends-and-insights/consumerlab/consumer-insights/reports/wearable-technology-and-the-internet-of-things, 2016.

2. Cisco visual networking index: global mobile data traffic forecast update, 2016-2021. 2017.

3. R Farrell, Small cells and their technological challenges.

4. Becker, Richard A., et al. "Clustering anonymized mobile call detail records to find usage groups." PURBA, San Francisco, CA, USA (2011).

5. https://dandelion.eu/datamine/open-big-data/

6. Daware, Sachin. Big Data analytics solution for small cells deployment using machine learning techniques. Diss. University of Oklahoma, 2016

7. https://www.epochconverter.com

8. Tan, Pang-Ning. Introduction to data mining. Pearson Education India, 2006.

9. Luna, Ana, and Romain Gauthier. "DAZIO: Detecting Activity Zones based on Input/Output call and SMS activity."

10. Cici, Blerim. "Mobile data analysis for smart city applications." 2016.

11. http://stanford.edu/~cpiech/cs221/img/kmeansViz.png

12. Lee, Hyunsoo, and Amarnath Banerjee. "Intelligent scheduling and motion control for household vacuum cleaning robot system using simulation based optimization." *Winter Simulation Conference (WSC), 2015*. IEEE, 2015.

13. https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Aco_branches.svg/2000px-Aco_branches.svg.png

14. Hashmi, Umair Sajid, Arsalan Darbandi, and Ali Imran. "Enabling proactive self-healing by data mining network failure logs." *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 2017.