Eötvös Loránd University
Faculty of Informatics

Data analysis of Telekom dataset: change point detection and planning

Dr. Tomáš Horváth
Professor at ELTE

Damiano Fossa
Computer Science

Dr. Maurizio Marchese
Professor at University of Trento

Dr. Máder Miklós Péter
Business Developer at Magyar Telekom

Budapest, 2017.

Acknowledgement

I would first like to thank my thesis supervisor Dr. Tomáš Horváth of the ELTE University. The door to Prof. Tomáš Horváth office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge my Entry Supervisor, Prof. Maurizio Marchese of the University of Trento as the second reader of this thesis, and I am gratefully indebted to his for his valuable comments on this work.

Finally, I must express my very profound gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Contents

Chapter 1

Introduction

Big Data, in the last few years, has became highly important to companies and the society. "*Telecom companies are sitting on a gold mine*", .as they have plenty of data. But what they require is a proper digging and analysis of the data to get deeper insights into customer behavior, their service usage patterns, preferences, and interests real-time. Most operators conduct, already, some analysis that enables them to use their internal data to boost the efficiency of their networks, segment customers, and drive profitability with some success. But the potential of Big Data poses a different challenge: how to combine much larger amounts of information to increase revenues and profits across the entire telecommunication value chain, from network operations to product development to marketing, sales, and customer service – and even to monetize the data itself. Big data offers several benefits: real-time call data record analysis to identify fraud immediately, proactive behavior-based and plan changes, total customer usage performance modeling, and measurement. Moreover, Big Data can improve marketing and sales: with some analysis, it can create event-based marketing campaigns, or offer the right service to the customer basing on his/her behavior and data usage. In conclusion, Big data offers telecommunication operators a real opportunity to gain a much more complete picture of their operations and their customers and to further their innovation efforts.

Given the premises above, telecommunication companies often do not know, or they do not have the right knowledge, to fully utilize their data. However, there is already some advances in this field. Several researchers are working on mobile datasets, to find some useful information. For example, from these datasets, it is possible to estimate the density of population in different regions which are covered by this dataset. Furthermore, some researchers discovered that there is a correlation between the amount of communication and poverty level. Also, using the mobile phone dataset, it is possible to detect emergen-

cies, or even predict them. These projects showed encouraging results for humanitarian organizations [8].

Also, mobile data allows to observe and quantify human behavior. This data opens many possible applications, with intrinsic economic value: for instance, as already mentioned, geo-localized advertisement. More specifically, recent research has shown that mobile phone data could detect where people are [15] and where people travel [13] including the purpose of the trip [29]. Many applications of modeling mobility aim towards transport planning and monitoring traffic with evident applications in accident management and traffic jam prevention.

The aim of our study is to work on the human mobility matter just mentioned. More precisely, we will exploit the dataset provided by Magyar Telekom and already refined by ELTE University researchers. Because of privacy reasons, the data we are going to use is anonymized; specifically, Magyar Telekom has performed a process through which the dataset is anonymized every day. Namely, the same user has a different ID on different days. This will make our research more challenging, but this will prove that we can get valid results, in accordance with privacy laws in force in the European Union. Given these assumptions, we will follow the following approach.
First, we will perform a static analysis. Namely, we will study the dataset, and we will try to remove unnecessary information to find a better representation of the data. And, finally, the last step, we will create a method, which will detect inconsistency on user's behavior. More specifically, the last step is divided in two parts:

- detection of unusual behavior (e. g. detecting the difference in the usual/average number of present phones in the area) most likely meaning/concerned with a traffic jam or car accidents in Hungarian highways;
- detection of the most probable way of a cell phone, (i.e. how to determine that someone is traveling on a highway);

1.1 Outline

The thesis is organized as follows.

- **Chapter 2** provides an analysis of the state of the art regarding Anomalies Detection in Time Series and Human Mobility.
- **Chapter 3** is a description of Anomaly Detection in Time Series, with a brief explanation of Time Series, and few types of transformation of the data.

- **Chapter 4** provides an examination of the kind of Anomaly Detection: Change Point Detection. Furthermore, there is an analysis of the state of the art, and a brief description of the Cumulative Sum method.
- **Chapter 5** is an analysis of the Planning part. More specifically, there is a description of the Dijkstra's algorithm and how it can be utilized to the Call Detail Records dataset.
- **Chapter 6** gives a description of the dataset used; data cleaning and preparation involved, and, lastly, the tools used.
- **Chapter 7** is an explanation of the experiments made and discussion.
- **Chapter 8** is the conclusion of the work and provides the future direction of research.

Chapter 2

State of the art

In the first paragraph of this chapter, an analysis of the state of the art concerning event detection will be presented. Event Detection, or, more generally Anomaly Detection, is the identification of events which do not conform to the expected pattern in a dataset. In a general view, anomalies are points which do not belong to the normal behavior. Extensive work has already been done in this field within diverse research areas and application domain. In the second paragraph, we give an overview regarding the human mobility study, in relation to research on telecommunication datasets.

2.1 Anomaly detection in Time Series

Anomalies detection in time series data has been studied since several years, by various researchers. Breunig et al. [9], proposed the Local Outliers Factor (LOF) algorithm. This method is based on the local density concept of each element. In fact, the elements which have a lower density than their close points are considered outliers. The local density is computed, by calculating the distances of each element with the neighbors. They could demonstrate that the LOF Algorithm can be a reliable algorithm to find anomalies. Oehmcke et al. [33] used the same algorithm to spot anomalies from the time-series long-term marine data from the stationary platform Spiekeroog, an island on the North Germany. From their research, about 83.34% of the detected events were confirmed to be true-positive. To improve the results, they applied dimensionality reduction pre-processing with ISOMAP, which led to higher average precision scores. Toshniwal et al. [38] tried to detect local outliers in addition to global outliers, using the HOT SAX algorithm, developed by Keogh, Lin, and Fu. This method has been extended successfully, where they were able to detect anomalies with a satisfying accuracy. Xi et al. [48] introduced a different method; they used the one-nearest-neighbor algorithm with Dynamic Time Warping distance with

numerosity reduction. In their work, they proved that the combination between the one-nearest-neighbor with DTW distance give exceptional performances. Shoubridge et al. [40], proposed an algorithm to spot change-point in a time-varying graph from which the nodes deviate from the normal behaviour. The algorithm works as follow: time sequence of several network features for all nodes in the graph were extracted. Next, a correlation matrix was built representing the correlation of behavior between all pairs of nodes in the graph on a certain time window. Lastly, they derived a behavior node of all node, and compare it to the recent past behavior. If the current behaviour was found significantly different from the past, it is flagged as anomalous. A similar approach was used by Guralnik et al. [25], specifically, the statistical method of the change-point detection problem in the traffic sensor data. In their research, they tested the *apriori* method to determine beforehand the change-point; after, they examined the incremental approach, in which they decide the function that is used for curve fitting in the interval between the successive change-point. Their experimental results suggested that both algorithms can correctly identify anomalies even if the noise is not too high.

2.2 Human mobility

Mobile phone datasets have been proved to be a reliable source of information prior to an appropriate refining. This kind of data can reveal a great deal regarding the human mobility. In this field of study, there has already been considerable research done. Understanding human mobility is highly important for epidemic control, urban planning, traffic forecasting, and mobile network applications. Xavier et al. [47] analyzed the mobility of the people after large-scale events, such as the final match of soccer championship, music festivals (Rock in Rio), New Year's Eve celebration, political manifestations, or the Olympics. In their work, they mainly focused on 2012 New Year's Eve celebration in three large Brazilian cities: Belo Horizonte, Recife, and Salvador. In their methodology, they defined the Pre-Event and the Post-Event periods as the time intervals that cover the minutes preceding and following the event. From their results, they discovered that the number of calls is increased by a factor between 6.7 and 12.6 during the New Year's Eve celebrations comparing with the number of calls during a day without events. Similar research was made by Bagrow et al. [2], where they pointed out that emergencies generate a spike in outgoing calls and text messages in the physical proximity of the event. This activity decreases considerably right after the emergency. More specifically, bombing activates the largest change in call activity than plane crash, earthquake and blackout (which, they still induce a stronger change than not-emergency events like football matches). These findings

were also confirmed by the work of Gao et al. [22]. Deville et al. [15], shown that the estimation of population density can be produced temporarily and spatially at a national scale.

Various method has been proposed to forecast human trajectories, including Markov Chain models, neural networks, Bayesian networks, and finite automation. Lu et al. [32] utilized the Markov chain model as a technique to measure the uncertainty of movements. The mobility data was provided by the telecommunication company Orange from 500.000 mobile phone subscribers in Ivory Coast. Their research proved that the model above is able to produce a good predictability. Their findings indicate that human mobility is greatly dependent on historical behavior and that they are far from random. Song et al. [42] made a similar research concerning the predictability of human mobility. They measured the entropy of every person's trajectory, and they found out a 93% potential predictability in user mobility. Csáji et al. [13], made another contribution on human mobility issue, using 100000 anonymized and randomly chosen people from Portugal. They could prove that it is possible to make an estimation of the population using mobile phone data. Moreover, they observed that most people spend most of their time in few locations and that the commuting distances can be explained by a gravity model. In their method, they used clustering and principal correlation analysis.

Calabrese et al. [10] proposed a new model to predict the location of a person over time based on individual and collective behaviors. The model is based on the individual's past trajectory and the geographical features of the area where the collectivity moves. Their experimental results show a good level of accuracy regarding prediction error. Calabrese et al. [11], made a similar research. They analyzed the crown mobility during special events. They collected nearly 1 million phone traces in the area of Boston, where they correlated social events people go to and their home location. The results of their research proved that there is a strong correlation between the two variables and that there is a good chance to make a good prediction out of it. Phithakkitnukoon et al. [36] studied an analogous matter, namely the relationship between people mobility and their social networks. They discovered that about 80% of places visited are within 20 km of their nearest social ties' location. The quantity rises to 90% at a radius of 45 km.

Chapter 3

Anomaly detection in Time Series

In this chapter, we investigate the problem of anomaly detection in Time Series data. First, we give a brief introduction of Time Series type of data. Subsequently, we discuss the most common kind of data transformation to find anomalies.

3.1 Time Series

In the last decade, there has been a burst of interest in mining time series data. Time series is a kind of data representation, where a set of observation is associated with a specific time. Time series are used in different fields of sciences such as statistics, earthquake prediction, signal processing, economy, finance, weather forecasting, astronomy, and communications engineering among many other fields [27].

3.1.1. Definition. *(Time series). We call a time series any (finite or infinite) sequence of observations* $(X_t : t \in T)$ *indexed by an ordered set (time). [19]*

There are several types of Time Series. These types can be distinguished based on some factors described above:

- *Continuous* - This type of observations are made continuously over time
- *Discrete* - As opposed to the Continuous Time Series, the discrete type are observation made only at certain time.
- *Stationary* - Data that fluctuate around a constant value
- *Non-stationary* - A series having parameters of the cycle (i.e., length, amplitude or phase) change over time

- *Deterministic* - Observations that can be predicted
- *Stochastic* - Observations are partly determined by past values. Future values have to be described with a probability distribution. This is the example of almost every natural time series.

Moreover, Time Series can present some patterns, which are listed below:

- *Trend* - It refers to a long-term decrease or increase of the data values
- *Seasonal* - It indicates a pattern which is influenced by seasonal factors. Few examples of those are: the quarter of the year, the month, or day of the week, a week, a month, or morning and evening
- *Cyclic* - Ihis pattern implies that the data rises and falls not of a fixed period. The duration of these fluctuations is usually of at least 2 years.[45]

In reality, we see Time Series as a finite number of values, in that case the sequence of observation is a sequence of random variables $(X_1, X_2, ..., X_n)$, is just a an n-dimensional random variable (or random vector). [23]

3.2 Transformation of the data

There exists an important number of challenges related to the handling of data, such as noise and high-dimensionality. Overcoming these challenges is a requirement to be able, for instance, to detect anomalies. Also, the data would be more readable, and therefore easier to be analyzed. Another important reason for the transformation of the data is the computational efficiency. Data with less noise and less dimensionality will be faster to process.
In this section, we discuss different transformations that are commonly utilized.

3.2.1 Aggregation

Aggregation is another technique used to transform time series data. This method groups consecutive values into a single representative value. Aggregation can bring a number of advantages. It can decrease data noise, therefore, makes it smoother. Also, it is able to reduce the data dimensionality. On the other hand, aggregation can present some drawbacks. For instance, by reducing the dimensionality, it can lead to some data loss. Hence it can make it more difficult detecting anomalies. Lin et al. [31] presented a method of aggregation called Piecewise Aggregate Approximation (PAA). The main incentive discussed in

their paper is to create an approximation of the data, which will fit in the main memory, subsequently, perform all the tasks in the main memory, and try to make very few accesses to the memory. The main objective of their approach is to decrease the dimension of a time series from n to M, where $M \leq n$. To achieve that, the data is divided into M equivalent "frames."
More formally:

3.2.1. Definition. *(Time Series Aggregation). Given a Time Series* $\boldsymbol{X}$*, an Aggregated Time Series* $\boldsymbol{AX}$ *is calculated as follows:*

$$\begin{gathered} \textit{given} \\ X = (x_1, ..., x_n),\ \textit{where}\ x_i \in \mathbb{R}\ \textit{and}\ i \in \mathbb{N} \\ \textit{then} \\ AX = (x'_1, ..., x'_M),\ \textit{where}\ M \leq n \\ \textit{and} \end{gathered}$$

$$x'_i = \frac{M}{n} \sum_{j=(\frac{n}{M}(i-1)+1)}^{\frac{n}{M}\times i} x_j \tag{3.1}$$

Intuitively, if M is too small, there will be information loss. On the other hand, if M is somewhat close to n, the time series resulting would be similar to the original one.

In our work, we are going to aggregate the Time Series by quarter-hours sand half-hours. More specifically, we are going to count every activity in the fifteen and thirty minutes time frame. Therefore, after this type of aggregation, we will have Time Series with fifteen and thirty minutes interval.

3.2.2 Discretization

Let's first give an explanation of discretization introduced by Hartemink [26].
A *discretization* of real-valued vector $\boldsymbol{x}$ of length N is an integer-valued vector $\boldsymbol{d}$ of the same length with the following attributes:

- Each element of $\boldsymbol{d}$ is in the set {0,..., D-1}, where D is a positive integer
- For all i,j, we have $d_i \leq d_j$ if $x_i \leq x_j$

Lin et al. [31] proposed a method to discretize Time Series, called SAX. This approach works as follows: it produces symbols with the same probability, which works well with

normalized Time Series having a Gaussian distribution; after, it identifies some breakpoints, which produces equal-size areas under the Gaussian Curve. Once the breakpoints were identified, it is possible to discretize the data. The authors used the aggregation method explained in the previous paragraph (the Piecewise Aggregate Approximation). Afterward, the PAA coefficients are obtained, and all the coefficients which are below the smallest breakpoint are associated with a letter "**a**". On the other hand the coefficients greater than the smallest breakpoint and smaller as the second smallest breakpoint, are given the letter "**b**", etc. Lin et al. proved their method to be highly competitive comparing to other data representation.

The main reason behind discretization is to decrease the computational complexity, by reducing dimensionality [46]. However, discretization, like aggregation, can lead to information loss.

Chapter 4

Change Point Detection

In this chapter, we are going to discuss the Change-Point technique, used to detect unusual behavior in time series events.

4.1 Introduction

Change Point Detection or Change Detection is a type of statistic analysis which aims to detect a change of the probability distribution of a stochastic process in a sequence of observations. This approach is based on the study of shifts in variance, mean, or correlation. There exist two main separations of Change Detection algorithms. The reason for this distinction is because of a different kind of development, as explained below:

- *Online change detection* - Online algorithms run in parallel with the process they are monitoring, and they process data as it comes available [18]. The goal of this approach is to detect changes from data coming at a particular rate. Therefore, this kind of technique is used to detect changes at a minimum rate possible, while minimizing the false alarms rate. Usually, it is implemented for real-time streaming data.

- *Offline change detection* - Basseville (1993) [5] discussed the method based on the works of Page [35] and Picard [37]. The goal is to discover when there is a change. More specifically, these kinds of algorithms detect variations of the mean, variance, shape and trend of the data.

4.2 State of the art

There are a great variety of Change Point techniques, which fits different applications. Allen B. Downey [18] proposed an algorithm for detecting and localizing change points in

time series data and a method to predict the distribution of the next point in the series. The algorithm proposed works for both "*online*" and "*offline*" problems. The core of method proposed is the Bayesian Statistics, which is based on the posterior probability *P(H|E)*, where *H* is the hypothesis, and *E* is the body of evidence. The result of the algorithm is a posterior distribution of the parameters of the process. Despite the encouraging results, the solution proposed in this work is still at an early stage, since, one of the few disadvantages is that it has a high computational time. Desobry et al. [14], presented a general framework for online abrupt change detection on signals data. Their solution is called Kernel Change Detection, which computes the dissimilarity between two times frames. If the measure obtained exceeds a certain threshold, a change occurred. This was proved to be robust to outliers. Also, simulations showed that the algorithm performs well the standard Generalized Likelihood Ratio (GLR) approach.

Bertrand et al. [7], proposed an offline detection approach for Time Series of independent observations, with an unknown number of change points. Their algorithm is based on the Filtered Derivative with the p-Value method. This method can detect the proper change points and, also, the false ones. Filtered Derivative has been introduced by Basseville & Benveniste in their work [5], however Bertrand et al. [7] believe that this method gives many false alarms. Therefore, they introduce their solution, where they calculate the p-Value to each candidate points, and they eliminate the points which are below a certain threshold. Their results proved the just mentioned approach is performing well, with low time and memory complexity.

Banerjee et al. [3] used a different approach to detecting abrupt changes. This approach is based on Minimax method, where the objective is to minimize the expected detection delay. For this reason, the proposed solution is based on the Cumulative Sum (CuSum) algorithm. More specifically, it is called generalized data-efficient cumulative sum (GDE-CuSum), which is a variation of the above-mentioned CuSum algorithm. In this paper, they extended their previous work [4] to the case when the post-change is composite. The authors have shown that their algorithm is asymptotically optimal. Furthermore, Siris et al. [41] proposed a solution which combined the Adaptive Threshold algorithm and CUSUM algorithm for detecting SYN flooding attacks which are the most common type of Denial of Service (DoS) attacks. They argue that the first one can detect changes, but not the intensity of those; while the latter method can identify the magnitude of the changes. From their work, they could prove that simple algorithms, such as Adaptive Threshold can perform well with high-intensity changes but has a lower performance in detecting small changes. Nevertheless, the CuSum algorithm proved to be a robust approach in a more different kind of event detection.

To conclude this paragraph, in our work, we are going to implement the solution proposed by Taylor 20000a [43], which used a combination of CuSum Algorithm and Bootstrap Analysis.

4.3 The Cumulative Sum Method

In the previous paragraph, the Cumulative Sum Method was mentioned several times. In this section, we give a more detailed explanation of that algorithm.
In statistics, the Cumulative Sum Method is a sequential analysis technique developed by E. S. Page in the University of Cambridge. As cited previously, it is a method belonging to the family of Change Point Detection algorithms. In his work, [34], he mentioned that one of the straightforward methods to detect a change in the mean, θ, of the distribution is a weighted sum of the last k observations. Moreover, he pointed out that if the group of observations considered is small, a big change is detected promptly, and small changes slowly. If the group of observations is larger, little changes are easier to detect. Generally speaking, the idea behind this method is to find a drastic change between the mean and the values.
The CuSum algorithm begins with a problem. Namely, when abrupt change is spotted at time n_c, on a discrete random signals with independent and identically distributed samples $\chi[n]$, there are two hypotheses to be considered: θ_0 and θ_1, both of them with probabilities p_{θ_0} and p_{θ_1}. The first hypothesis corresponds to the *no change* hypothesis, while the second hypothesis corresponds to the *one change* hypothesis [24].
Each sample follows a probability distribution function (PDF) $p(x[n], \theta)$; that is each item in the discrete random signals depend on a deterministic parameter θ such as the mean. When a sudden change happens, θ is modified at the time n_c. Hence, $\theta = \theta_0$ before the change and $\theta = \theta_1$ after the change.
Under the first hypothesis (H_0) the PDF is given by:

$$p_{\chi|\theta_0} = \prod_{n=0}^{k} p(x[n], \theta_0) \tag{4.1}$$

Under the second hypothesis (H_1) the PDF is given by:

$$p_{\chi|\theta_1} = \prod_{n=0}^{n_c-1} p(x[n], \theta_0) \prod_{n=n_c}^{k} p(x[n], \theta_1) \tag{4.2}$$

In the case a change has been detected, the change has to estimated with an estimator $\widehat{n_c}$ [24]. Since the two PDFs just mentioned are difficult to calculate, in our particular case, we decided to validate the hypothesis with real case scenarios.
The Cumulative Sum method, which we are going to use in our work, is a cumulative summation of the difference between the current value and the average of all the values of a Time Series. If during a period of time, the points tend to be above the average, the Cumulative Sum will be positive; therefore it will steadily grow. Likewise, if the values stay below the average, the Cumulative Sum will steadily decrease. A sudden change on the tendency of the CuSum, indicates an abrupt variation on the mean.
The estimator chosen for our particular problem is the maximum absolute value of the CuSum.

A more detailed explanation of our method is given in the next paragraph.

4.4 The algorithm

As already mentioned, there exist several implementations on Change Point Detection. In this work, we are going to explain the version presented by Taylor (2000a) [43].
The procedure uses a combination of cumulative sum (CuSum) and bootstrapping. The algorithm is structured as follows:

1. It first receives a Time Series X, where each date i is associated with a value X_i
2. It, then, calculates the average of all the values previously mentioned. From now on, the average will be called $\overline{X}$
3. At this point, it computes the cumulative sum using the following formula:
   $CUSUM = \sum_{i=1}^{N}(X_i - \overline{X}) \;\; for \;\; i = 1, 2, 3, ..., N$, where N is the total number of values.
4. The first value of the Cumulative Sum is zero:
   $S_0 = 0$
5. It measures the CuSum maximum value, $S_{max} = \max_{1 \leq i \leq N} (X_i - \overline{X})$
6. It measures the CuSum minumum value, $S_{min} = \min_{1 \leq i \leq N} (X_i - \overline{X})$
7. Compute an estimator of the magnitude of the change. This estimator is determined from the difference between the maximum value and the minimum value:
   S_{diff} = S_{max} - S_{min}

8. It performs the bootstrap analysis. This procedure consists on randomly sampling the original values. The method is called *Sampling without replacement*. We are going to create 1000 samples.

9. At each sample generated, the algorithm computes the bootstrap CuSum, using the same process explained on points 2 and 3

10. It calculates the maximum, minimum, and difference of the bootstrap CuSum, as defined on points 5, 6 and 7

11. At each bootstrap, the method determines whether the bootstrap difference is less than the original difference

At this point, we calculate the confidence level that a change occurred. Let N be the number of bootstrap samples (in our case 1000) and X the number of bootstrap where $S_{0diff} < S_{diff}$, the *Confidence Level* is computed as follows:

$$ConfidenceLevel = \frac{X}{N} \tag{4.3}$$

A significant change has been detected if the confidence level is above 0.9.

The change-point analysis is based on the mean-shift model. Let $X_1, X_2, ..., X_n$ represent the data in time order. The mean-shift model can be written as $X_i = \mu_i + \epsilon_i$, where μ_i is the average at time i; ϵ_i is the time error associated with the i-th value.

Once a change has been detected, we need to estimate at what time this change happens. The estimator we are going to use is the *CUSUM* estimator proposed by Taylor (2000a):

$$|S_m| = \max_{I=1,2,3,...,N} |S_i| \tag{4.4}$$

Where N is the amount of points considered. S_m is the point most distant from zero in the $CuSumM$ chart. The point m represents the last point before the change, and $m + 1$ represents the first point after the change.

A more detailed explanation of the algorithm is described below:

Algorithm 1 Change point detection

Receives the a Time Series, where each element is associated with a value

$ValueList \leftarrow TimeSeries.values()$
$Mean \leftarrow ValueList.mean()$

$CUSUM \leftarrow []$
$CUSUM.append(0)$
for Value in ValueList **do** ▷ Calculate the Cumulative Sum
  $CUSUM.append(CUSUM_{i-1} + (Value - Mean))$
end for

▷ Gets the Min, Max and Difference from the Cumulative Sum
$CUSUMmax \leftarrow max(CUSUM)$
$CUSUMmin \leftarrow min(CUSUM)$
$CUSUMmaxINDEX \leftarrow CUSUM.index(CUSUMmax)$
$CUSUMminINDEX \leftarrow CUSUM.index(CUSUMmin)$
$CUSUMdiff \leftarrow CUSUMmax - CUSUMmin$

$Counter \leftarrow 0$
for $i = 0 \rightarrow 1000$ **do** ▷ Perform the Bootstrap analysis
  $NewCUSUM \leftarrow []$
  $NewCUSUM.append(0)$
  $NewValueList \leftarrow random.permutation(ValueList)$
  for Value in NewValueList **do**
    $NewCUSUM.append(NewCUSUM_{i-1} + (Value - Mean))$
  end for
  $NewCUSUMmax \leftarrow max(NewCUSUM)$
  $NewCUSUMmin \leftarrow min(NewCUSUM)$
  $NewCUSUMmaxINDEX \leftarrow NewCUSUM.index(NewCUSUMmax)$
  $NewCUSUMminINDEX \leftarrow NewCUSUM.index(NewCUSUMmin)$
  $NewCUSUMdiff \leftarrow NewCUSUMmin - NewCUSUMmin$
  if $NewCUSUMdiff < CUSUMdiff$ **then**
    $Counter \leftarrow +1$

```
    end if
end for

ConfidenceLevel ← (Counter ÷ 1000) × 100
if ConfidenceLevel ≥ 90 then
    print("The confidence level is over 90%")
end if

CUSUMmaxEstimator ← max(abs(CUSUM))
                                          ▷ Return the Change Point
ChangePoint ← CUSUM.getIndex(CUSUMmaxEstimator)

    return ChangePoint
```

Chapter 5

Planning

In this chapter, we are going to discuss the second part of our research mentioned in the Introduction. This part concerns the planning, namely, taken a tower cell location, which is the shortest path to reach another antenna. We are going to use the Dijkstra algorithm and Google Maps API, which find the shortest path inside a graph.

5.1 Introduction

In 1959 a three-pages long paper named *Note on Two Problems in Connexion with Graphs* [17] was published in the journal *Numerische Mathematik*. In this work, Edsger W. Dijkstra, introduced a solution for the *Shortest-Path* problem, in the graph theory. This algorithm is called the Dijkstra Algorithm. There exist many variations of the algorithm; the original version finds the shortest path between two nodes. However, a more common variant takes a single node of the graph as a source and finds the shortest path from the source to the other nodes.

5.2 The algorithm

The starting point of the algorithm involves the assignation of the **initial node**. Dijkstra's algorithm assigns some initial distance values, and it updates them step by step.

1. It assigns to every node a distance value. It sets it to zero for our initial node and to infinity for all other nodes.

2. It sets the initial node as current. Then, it marks all other nodes unvisited. Create a set of all the unvisited nodes called the **unvisited set**.

3. From the current node, the algorithm considers all of its neighbors, and it calculates their tentative distances from the present point. It, thus, compares the newly calculated tentative distance to the currently assigned value and assign the smaller one.

4. When all of the neighbors of the current node are examined, the algorithm marks the current node as visited and removes it from the unvisited set. A visited node will never be checked again.

5. If the destination node has been marked visited (when planning a route between two specific nodes), or if the smallest tentative distance among the nodes in the unvisited set is infinity, then it stops. The algorithm has finished.

6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new current node, and go back to step 3.

The algorithm has computational complexity $O(|V|^2)$, where |V| is the number of nodes. Leyzorek et al. 1957 [30] proposed an alternative version of the algorithm, by using a Priority Queue, implemented by a Fibonacci Heap. This variant could improve the computational complexity of the original version: $O(|E| + |V| \log |V|)$, where |E| is the number of edges [16].

In our work, we are going to adapt the graph algorithm, using the antennas' locations as nodes.

5.3 Applications

TThe Dijkstra Algorithm has several applications in many fields of study. For instance, Stolfi et al. [21], in their paper implemented the algorithm for the Image Foresting Transform (IFT), which is a tool for the design, implementation, and evaluation of image processing. The basic idea of IFT is that it represents a minimum-cost path forest in a graph, where the nodes are the image pixels, and the arcs are the adjacency relations between pixels. Accordingly, the IFT was computed using the Dijkstra Algorithm, which was customized to permit multiple sources. Road networks (like communication networks) are strongly related to the shortest-path matter. Akkanen et al. [1], presented the use of the shortest-path algorithm within a network planning tool (Nokia Networks, 1999). The could prove that the Dijkstra Algorithm can solve several practical problems with minor changes of the same. Specifically, this method has been used in unprotected routing to find a single

route between nodes with the minimum cost. Also, it has been implemented in semi-manual routing, where the user specifies how the traffic is routed. Furthermore, the Dijkstra Algorithm has been used in protected routing.
Eklund et al. [20] discussed a modified version of the algorithm, taking in account of both static and dynamic search heuristic and multiple source node. The could prove that their solution is extremely efficient. Moreover, the authors applied their solution to a 3D Spatial Information System for routing emergency service vehicles.
Inagaki et al. [28] discussed the Dijkstra's algorithm differently, by pointing out that the mentioned algorithm, and similar methods, search only the shortest route, and they cannot determine any other short way. Accordingly, they proposed a modified version of it. More specifically, they suggested a genetic algorithm, which can find the semi-shortest routes, as well as, the shortest route.
As can be seen from these examples, the Dijkstra's method, since its conception, has been analyzed in many studies. Other cases include the work of Zhong [49], where he studied the algorithm in the GSM network; more precisely, on the Mobile Station Roaming Number (MSRN). The MSRN is a temporary phone number assigned to a mobile station which roams into another numbering area (this usually happens when the phone moves to another country). Benaicha et al. [6], instead discussed the algorithm from the hardware perspective, where they suggested a new approach by using FPGA development card (Xilinx), an integrated circuit.
As can be seen, the Dijkstra's algorithm has several useful applications, mostly related telecommunication routing. The thesis' second research question deal with route planning. Therefore, this algorithm will be applied using the Telekom Call Detail Records dataset. The method used is described with more details in the Paragraph 5.4.

5.4 Method

In this paragraph, we present our solution regarding the planning part of our research. Our method uses a combination of the Dijkstra Algorithm and Google Maps API.
More specifically, the solution works as follows:

1. It gets the West points of Hungary.
2. It computes the tower cell location closest to the West point, which will be used as *source node*.
3. It calculates the distance between each pair of nodes.

4. Starting from the source node computed, it builds the graph basing on the distances computed in the previous point.

5. Lastly, using the Dijkstra Algorithm, our method returns the path from the source to any other nodes.

The pseudo code is presented in the Algorithm below.

Algorithm 2 Planning

Receives the list of coordinates of Hungary, $Coordinates$
$West \leftarrow googleMaps.getWest()$
$SourceNode \leftarrow googleMaps.getWest()$
$Pairs \leftarrow getAllPairs(Coordinates)$
$Distances \leftarrow []$
for $PairinPairs$ **do**
    $PointOne \leftarrow Pair[0]$
    $PointTwo \leftarrow Pair[1]$
    $Distance \leftarrow googleMaps.getDistance(PointOne, PointTwo)$
    $Distance.append(distance)$
end for

$Graph \leftarrow BuildGraph()$
$Path \leftarrow Dijkstra(Graph, SourceNode)$
    return $Path$

Chapter 6

Dataset

6.1 Call Details Record

The Call detail record (CDR) is a data record produced by a telephone transaction or other telecommunications equipment, that records the details of a phone call or other telecommunications transaction (e.g., text message) that passes through that facility or device. The location of a cellphone is, usually, estimated from the antenna to which the device is connected. Each time a phone communicates with the antenna, the wireless carrier records the "cell site identifier" [12].
Telecommunication companies record this kind of information for billing purposes. Thus, the user's location is marked with an estimation that is determined by the local tower density. The area of action of a tower varies from a few hundred meters in metropolitan area to a few kilometers in rural regions [42]. In general, a mobile phone tracks our location every time we text, call or web browser; and even passively when it communicates with the cellular network access point [29].
The record contains various attributes of the call, such as time, duration, completion status, source number, and destination number. It is the automated equivalent of the paper toll tickets that were written and timed by operators for long-distance calls in a manual telephone exchange.
A call detail record contains metadata, namely, data about data, containing attributes that describe a particular aspect of a telecommunication transaction but does not include the content of that operation. In a more simple way, a Call Detail Record describing a particular phone call might include the phone numbers of both the calling and receiving parties, the start time, and duration of that call. In the actual modern use, the Call Detail Records are much more detailed, as described below:

- The phone number of the subscriber originating the call
- The phone number receiving the call
- The starting time of the call (date and time)
- The call duration
- The billing phone number that is charged for the call
- The identification of the telephone exchange or equipment writing the record
- A unique sequence number identifying the record
- Additional digits on the called number used to route or charge the call
- The results of the call, indicating, for example, whether or not the call was connected
- The route by which the call entered the exchange
- The route by which the call left the exchange
- Call type (voice, SMS, etc.)
- Any fault condition encountered

6.2 The dataset

Magyar Telekom provided the dataset used for this work. The type of data is Call Detail Records, which is, as explained in the previous chapter, information that telecommunication companies store in the closest antenna from the user location, when the customer makes or receives a call, or sends or receives a message. This type of data contains highly sensitive information about the users. For that reason, we handled anonymized data. More specifically, each customer has been associated an identification number, which changes on a daily basis.

The dataset is structured in five files: CRM, MSC, NGPRS, TAC, and CELLS.
The CRM table file contains general information about the user, such as postal code, age, gender, a unique identifier generated by the account's holder identity, and a unique identifier generated from the phone number. It has 101,865 rows. A more detailed description can be seen in Table 6.1.
The MSC file contains information about the call and SMS activity of each user, such as

Call Event ID, TAC, event type (Initiated Call, Answered Call, Received SMS, Sent SMS), event date, event duration, destination point, an identifier of the hosting tower. This file includes 5,050,008 records. A more detailed description can be seen in Table 6.2.
The NGPRS table contains records regarding calls, such as duration, date time, tower ID, technology (2G, 3G, 4G). This file has 33,068,389 rows. A more detailed description can be seen in Table 6.3.
Lastly, the TAC file incorporates information about the Type Allocation Code, part of a mobile device's IMEI identifying maker and model. A more detailed description can be seen in Table 6.4.

This dataset contains records stored from the 10th of April 2017 to the 17th of April 2017.

In addition, another dataset was used. This second dataset contains far more records that the first dataset. However, it has the same structure as the first one. The CSV table contains around 5.5 millions rows. The MSC file has 868,316,191 records. The NGPRS table includes 5,715,692,694 rows. This dataset contains records stored from the 15th of September 2016 to the 15th of October 2016.

SubscriberID	ClientID	Client Zip	Client City	Client Gender	Client Age	Client Magenta	Client Arpu	Client Switch
58F755BC	58F1E3A3	4287	Vámospércs	0	72	0	5	0
58F755AD	58F15AB5	7150	Bonyhád	1	51	1	1	0
58F755B4	58F2B149	8438	Veszprémvarsány	0	40	0	3	0

Table 6.1: CRM Table

SubscriberID	CallID	Tac	Type	DateTime	Duration	DestpointID	CellID
58F755BB	9594364	86086503	1	12/04/17 17:18	58	1	40141B03F
58F755BD	14308753	35523403	1	12/04/17 15:12	21	4	FF6050FF5
58F755BB	9594364	86086503	1	12/04/17 21:13	29	3	40141B03F

Table 6.2: MSC Table

SubscriberID	CallID	Tac	Technology	Type	DateTime	Duration	CellID
58F755BB	1059300461	86086503	4	85	12/04/17 15:07	1806	CE9422CC5
58F755BB	941873055	86086503	3	18	12/04/17 18:26	71	45C5F6D1B
58F755BB	1059300461	86086503	4	85	12/04/17 15:07	598	5B8B58DAC

Table 6.3: NGPRS Table

Furthermore, the dataset contains a file with the locations of the antennas in Hungary

Tac	Manufacturer	Model	Aka	Os	Year	Lte
1147200	Apple	iPhone A1203	Apple iPhone, Inc.	iOS	2007	0
1110400	Enfora	GSM0308	GSM0308-71	Proprietary	2007	0

Table 6.4: TAC Table

(6.5), which has 39,847 rows. These locations have been plotted into a map, as shown in Figure 6.1.

As can be seen in Figure 6.2, there are several tower cells next to the Hungarian main roads, such as the highways. The aim of our study is to find some unusual behavior happened in those routes.

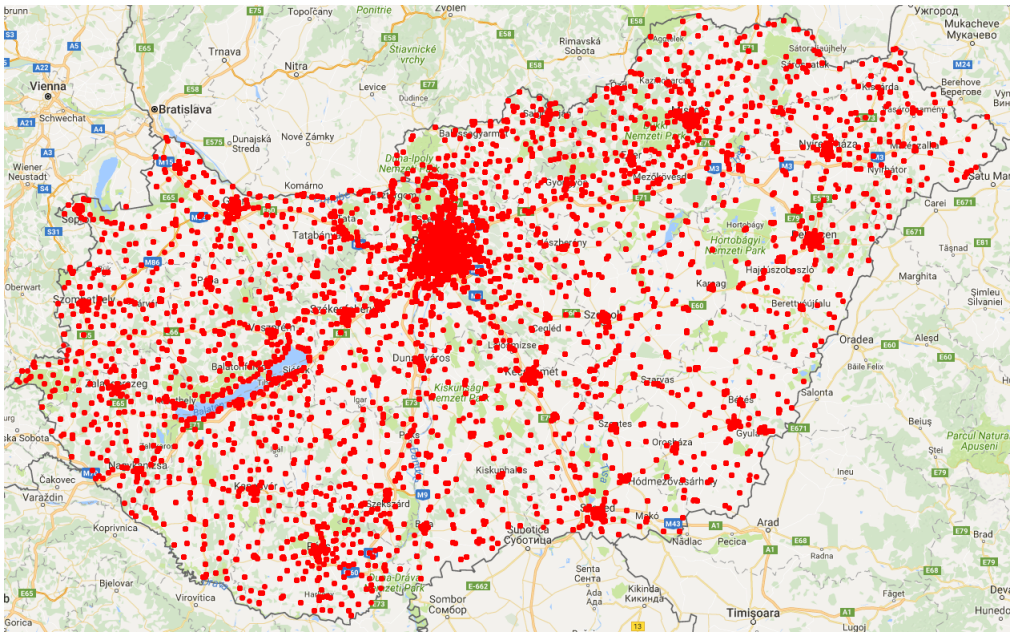

Figure 6.1: Towers' location in Hungary

Technology	CellID1	CellID2	Latitude	Longitude
2G	0112F60300220BBC	B677D701B	47.464148	19.116065
4G	0212F603107304AF	AD3B8C80A	48.320775	21.102736
3G	0112F60300820C2B	29F62254A	47.476661	19.029112

Table 6.5: CELLS Table

Figure 6.2: Towers' location in M1 Highway

6.3 Pre-processing

Before the implementation of the algorithm explained in Chapter 4, we are going to talk about the *pre-processing* phase. This stage is highly important, since, in general, the data contains a significant amount of noise and not useful information. Moreover, in this phase of the procedure, the data is prepared for the next step, to be ready for the algorithm we are going to apply. Therefore, there is the need to clean the dataset considered. The first point of pre-processing is *Data Cleaning*, which is explained in detail in the next paragraph. Afterward, there is a description the second phase, *Data Preparation*.

6.3.1 Data cleaning

Data cleaning is a valuable stage of the process, since it can reduce or remove significantly, the noise and unuseful information contained in the dataset. In our case, we are not going to take into consideration the CRM table, since it provides customers' information, such as address, or place of birth, which are not crucial to our project. Also, we are not going to handle the TAC table, because it has accurate data regarding each phone. However, this information is not critical for our project.
In our case, we are going to exploit the MSC, NGPRS, and CELLS files, where irrelevant columns will be removed. After this step, the MSC table has DateTime, CallID, and CellID column. The NGPRS table incorporates the same columns. The CELL table, instead, has the Cell ID, Latitude, and Longitude columns. The new CellID column contained in the CELL table is created by putting the Cell ID 2 column below the Cell ID 1 columns with the same values in the Latitude and Longitude column. Moreover, the Technology column is deleted, since it is considered irrelevant to our study. The outcome of this process is showed in Tables 6.6, 6.7, 6.8.

DateTime	CallID	CellID
15/04/17 21:01	9594364	40141B03F
12/04/17 23:28	9594364	42861B03F
13/04/17 09:39	9594364	35523403

Table 6.6: New MSC Table

DateTime	CallID	CellID
12/04/17 15:07	1059300461	40141B03F
12/04/17 08:36	941873055	42861B03F
12/04/17 18:26	1059300461	35523403

Table 6.7: New NGPRS Table

6.3.2 Data preparation

After the Data Cleaning, the Data Preparation is the next step. This stage deals with the arrangement of the data, to be ready for the algorithm.

In this phase, we used two different approaches. The first one deals with general data, where we create a table containing the activity of each day considered, without taking into consideration a particular antenna. The second procedure, instead, deals with information related to a specific location (latitude and longitude).

The first method is explained as follows: firstly, the MSC and NGPRS files were processed distinctly. More specifically, these data were sorted by date, and, afterward, they were grouped by date with an interval of fifteen and thirty minutes. Each period is associated with the total number of calls or messages related to that time frame. Secondly, the two files obtained (MSC and NGPRS), were merged to create a single file, which contains all the information concerning MSC and NGPRS. The aforementioned step creates a file called "*grouped.csv*" as showed in the Table 6.9. The file just mentioned, contains data regarding each day considered in the dataset, and each antennas location.

The second approach is explained as follows.
The method works similarly as the method explained above. Namely, the MSC and NG-

CellID	**Latitude**	**Longitude**
0112F60300220BBC	47.464148	19.116065
B677D701B	47.464148	19.116065
0212F603107304AF	48.320775	21.102736
AD3B8C80A	48.320775	21.102736
0112F60300820C2B	47.476661	19.029112
29F62254A	47.476661	19.029112

Table 6.8: New CELLS Table

PRS files were processed in an isolated way, where we removed the CellID column, and, instead, we put the latitude and longitude, associated to that ID contained in the Cells table. Afterward, we selected some incidents happened in the days considered by the dataset. Subsequently, we identify the tower cells' locations close to that episode's location, and, then, we grouped the information by date associated with a number of calls, for each site. The outcome of this process looks identical as the Table 6.9.

After this step, the data is ready to be processed by the algorithm.

DateTime	**Calls**
2017-04-10 00:00:00	48354
2017-04-10 00:30:00	30367
2017-04-10 01:00:00	30900
2017-04-12 04:30:00	36063
2017-04-12 05:00:00	45012
2017-04-12 05:30:00	57429

Table 6.9: *grouped.csv* Table

6.4 Tools

In this section, we give a brief description of the tools used to clean, prepare the data, and to perform the algorithm.

6.4.1 Python

Python is programming language conceived in the 1980s by Guido van Rossum. This language is made for general purposes. However, it is widely used in the scientific field, especially in Data Science. This is due to the fact that Python contains a good amount of libraries, made to handle math, statistics, data processing, and data visualization. Few example of these libraries are "Numpy", implemented to process arrays, lists, and math operations; "Pandas", created to process certain data, and create some statistics out of it; and "Matplotlib", used to display data in graphs and charts.

6.4.1.1 Pandas

Pandas is a software library written for Python programming language. This software is used for data manipulation and analysis. More specifically, it is widely used to handle Time Series and to manipulate CSV tables, such as grouping, merge and join. In our project, this library was used mostly for data cleaning and data preparation.

6.4.1.2 Numpy

Numpy is a software library written for Python programming language, used to handle multi-dimensional arrays and matrixes. Also, it provides functions and operators that operate efficiently on arrays. In this project, Numpy was used to create the Cumulative Sum list, to create random permutations of the array during the bootstrap analysis, to compute the mean from test values and to do some other mathematical operations.

6.4.1.3 Matplotlib

Matplotlib is another software library written for Python programming language. It is used to plot several kinds of graphs, such as line plot, histograms, scatter plot, and 3D plot. In this work, Matplotlib was used to plot the general Time Series and the Time Series related to a specific tower cell.

6.4.2 Bash

Bash is is a Unix shell and command language written by Brian Fox for Unix systems. In this project, Bash was mainly used for its high performance. Specifically, when processing the data with Pandas, was supposed to take an excessively amount of time, we manipulated the dataset with some bash commands. We largely used the *Cat* command to print the content of a file to another or to merge two files. The *Grep* command, to find specific strings in the data inside a file, such as latitude and longitude, in order to get the call and message activity on a specific antenna. Lastly, the *wc* command to print the number of rows of a CSV file.

6.4.3 Test environment

As previously mentioned, we used two datasets: the first concerns the days of the Easter week, and the second deals with the days from the 15th of September 2016 to the 15th of October 2016. The second dataset contains much more data that the first one, therefore, we used a dedicated server to perform the experiments. The tests of the first dataset were executed in Macbook with the following characteristics: 2 GHz Intel Core i7, 8 GB 1600 MHz DDR3 and 256 GB of storage.
An Ubuntu 16.04.1 LTS server used to perform the second dataset tests. It has the following specifics: 15 Intel(R) Xeon(R) CPU, 65 GB of RAM, and five hard drives.

Chapter 7

Experiments and Discussion

All the experiments explained in the following paragraphs were made on two datasets with different time frames. Below, we are going to explain the tests made on the first dataset. Subsequently, we are going to describe the experiments made with the second dataset.

7.1 First Dataset

The first thing we did is the plotting of the general call/messages activity, throughout the Easter week. Specifically, we have plotted the days from the 10th of April to the 17th of April. We have identified some car accidents, and traffic jams happened in the week mentioned in the previous sentence. There were different serious incidents occurred in the Hungarian highways, which leads the police to close the roads or to narrow them. Consequently, it creates various traffic jams. These incidents happened in the Hungarian highways M1, M3, M5, M7. The examples considered are the follows:

- Due to increased traffic, a congestion of 3 kilometers on 13 April 2017 was developed at the M1 motorway leading to Austria, at the border crossing point. According to the road map information, increased traffic is also slowed down by an accident on the M1 motorway, near Hegyeshalom. Here, it is expected to decrease the lane bandwidth and increase congestion.
- A road traffic accident, happened on Thursday a few minutes before 17 o'clock on the M3 motorway leading to Hatvan. Up until now, unclear conditions have hit two cars and one truck. Both traffic lanes can be accessed, but the accident caused a long congestion. [39]

Firstly, we have plotted the activity contained in the MSC file with fifteen minutes and thirty minutes interval respectively. The graphs with both time frames can be seen in the

Figure 7.1 and Figure 7.2.

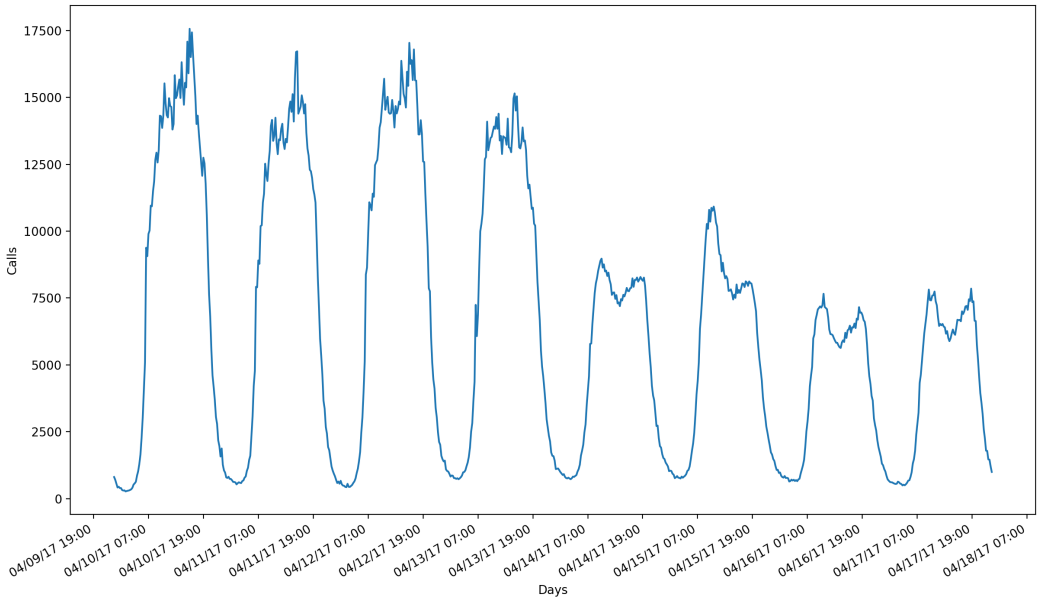

Figure 7.1: General activity during the Easter week, MSC file, 15 minutes time frame

Secondly, we have plotted the activity contained in the NGPRS file with fifteen minutes and thirty minutes interval respectively. The graphs with both time frames can be seen in the Figure 7.3 and Figure 7.4.

Subsequently, we plotted the whole calls and messages activity throughout the days considered before, by combining the MSC and the NGPRS tables, with fifteen minutes and thirty minutes interval accordingly. The graphs with both time frames can be seen in the Figure 7.5 and Figure 7.6.

As can be seen from the MSC and NGPRS graphs, the first seems less stable than the latter. Namely, it changes considerably on the weekend comparing to the rest of the days. MSC contains both calls and messages, therefore, this might be the reason for the just mentioned difference.

Moreover, as can be detected from both MSC and NPGRS graphs, and the general graphs, there was a considerable more activity between the 10th and the 14th of April, on both 15 and 30 minutes time frame. This condition could be explained as follows: the Easter week is a particular week, where most of the people work until Thursday, instead of Friday. Therefore, it explains the fewer activity from Friday to Sunday.

Consequently, we continued the investigation, and we applied the algorithm explained in a

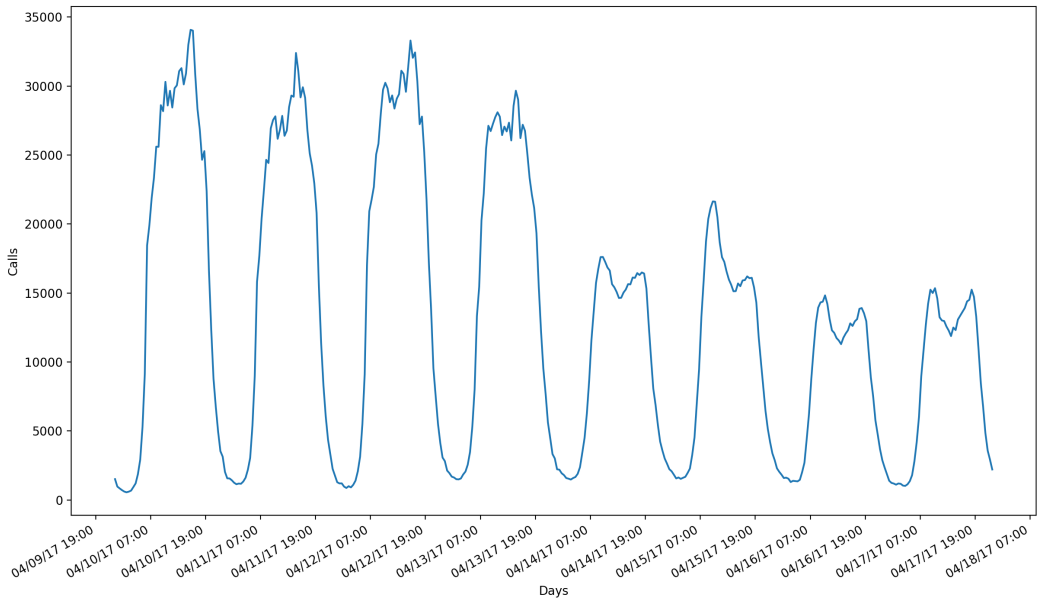

Figure 7.2: General activity during the Easter week, MSC file, 30 minutes time frame

previous chapter. Firstly, we have computed the Cumulative Sum of each time frame. After this step, we have performed the bootstrap analysis, which gives the Confidence Level above 95%. We assumed that something happened in those days. Lastly, we have taken as change point estimator, the furthest Cumulative Sum value from zero, and we have identified 13th of April around 8.30PM as Change Point in each graph presented earlier. The congestion lasts the whole day.

Subsequently, we deepen the analysis, by taking into consideration the accidents selected in the Easter week.
As mentioned earlier, we selected two cases happened during Easter week. Firstly, the big traffic jam near the Austrian-Hungarian border and the car accident occurred on the M3 motorway leading to Hatvan. Hence, we identified the coordinates of the tower cells near the area of the congestion.
The latitude and longitude determined in that area are: 47.9126656221,17.1678863103

We followed the same approach used to investigate the general activity. We, first of all, plotted the activity associated with that antenna using the MSC table and the NGPRS table with fifteen and thirty minutes time frame (7.7 and Figure 7.8). These four graphs present a similar trend. Afterward, we have plotted the activity of the MSC and

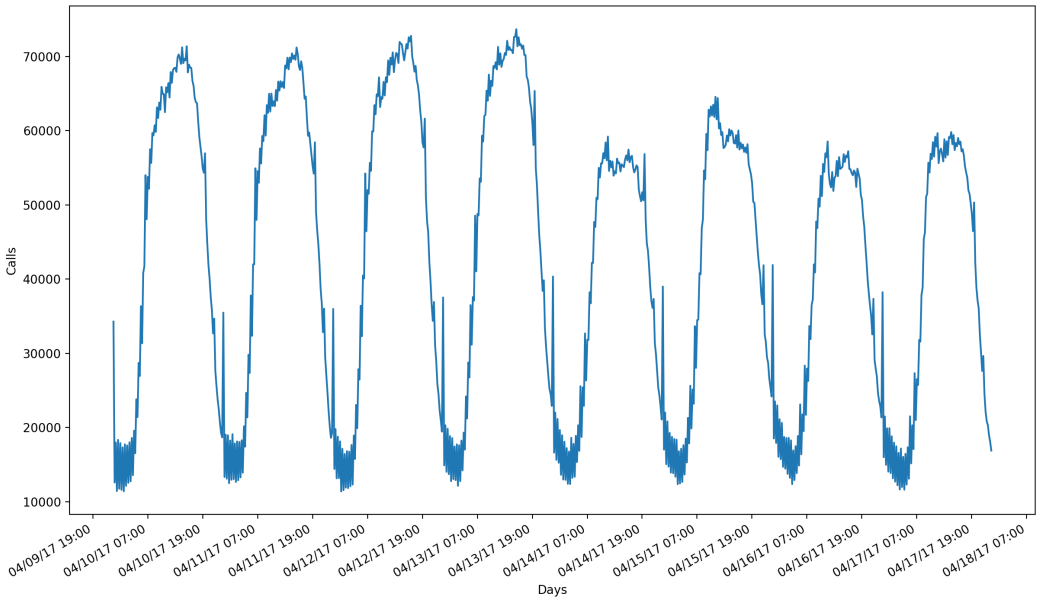

Figure 7.3: General activity during the Easter week, NGPRS file, 15 minutes time frame

NGPRS combined, associated to that antennas, as can be seen in Figure 7.7 and Figure 7.8.

Here, call activity seems more sparse, with some peaks on the 10th, 13th, 15th and the 17th of April. Consequently, we continued the investigation, and we applied the algorithm explained in a previous chapter. Firstly, we computed the Cumulative Sum of each time frame. After this step, we performed the bootstrap analysis, which gives the Confidence Level above 95%. We assumed that there was at least a Change Point. Lastly, we took as change point estimator, the furthest Cumulative Sum Value from zero, and we have identified 13th of April around 8 PM as Change Point, where the accident happened at around 5 PM.

Afterward, we considered the second incident occurred in the Easter week, namely, the traffic accident which took place on Thursday 13th of April a few minutes before 5 PM on the M3 motorway leading to Hatvan. The incident created a high congestion where only the third lane road could be used. The latitude and longitude of the closest determined in that area are: 47.6414444197,19.5261483162

Likewise, the same procedure of the previous incidents was followed.
Therefore, we, plotted the activity associated to that antenna using the MSC table and

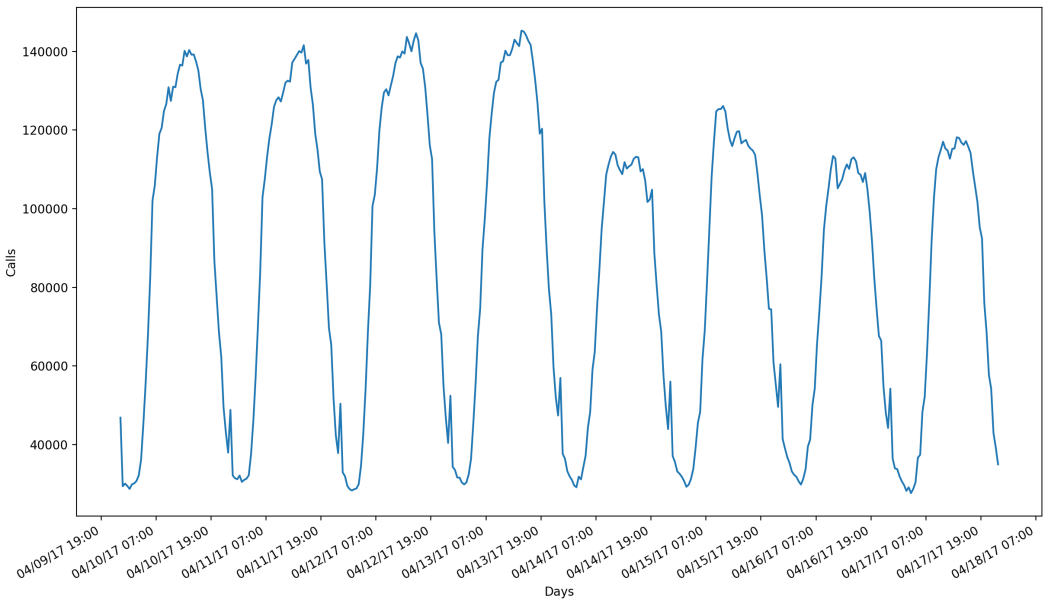

Figure 7.4: General activity during the Easter week, NGPRS file, 30 minutes time frame

the NGPRS table with fifteen and thirty minutes time frame. Afterwards, like the traffic jam considered, we have plotted the activity, with fifteen and thirty minutes time frame, associated to that antennas, as can be seen in Figure 7.9, 7.10, 7.11, 7.12, 7.13 and 7.14.

In this case, there is a bit difference comparing to the tower cell considered earlier. More specifically, with slightly more activity on the first four days, that the rest of the week. Also, the MSC and NGPRS graphs showed different trends. Thus, we decided to perform the algorithm, to see if the outcome is similar. Firstly, we have computed the Cumulative Sum of each time frame. After this step, we have executed the bootstrap analysis, which gives the Confidence Level above 95%, which gives the assumption that there was at least one significant Change Point. Lastly, we have taken as Change Point estimator, the furthest Cumulative Sum Value from zero, and we have identified 13th of April around 9 PM as Change Point. The result obtained showed the same Change Point: the 13th of April, at around 8PM.

We continued the investigation, by running the algorithm using the file with MSC and NGPRS combined. The results obtained showed the same Change Point detected in the MSC and NGPRS taken individually.

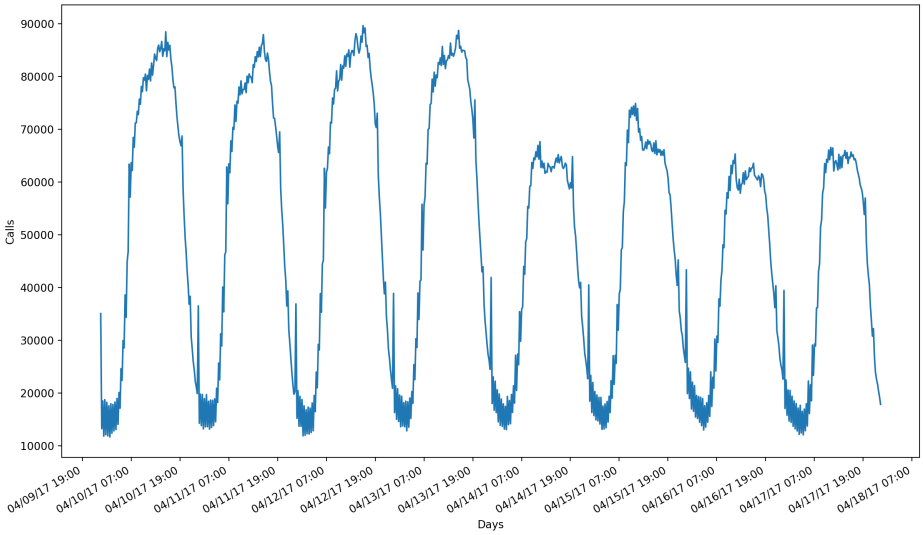

Figure 7.5: General activity during the Easter week, 15 minutes time frame

7.2 Second Dataset

The second dataset considered covers the days between the 15th of September 2016 to the 15th of October 2016. In this case, we tried a different approach on preparing the data, plotting and performing the algorithm. Contrary to the first dataset, in which we analyzed each day contained in it, here we took few days singularly. We, firstly, identify a car accident happened on the 27th of September 2016:

- On Tuesday afternoon, early evening there was a serious traffic accident on Highway 1 and the M1 motorway. A man died on Tuesday afternoon in Nagyigmánd's accident. According to the police, for a while, due to location and rescue, the traffic of the highway was stretched to Route 1 on a single stretch. A rescue helicopter was also called to the scene; the passenger car driver was transported to the hospital in Győr with life-threatening injuries. [44]

The estimated incident location is 47.6819212317,18.0099900000. The closest tower cell identified has the following coordinates: 47.6763226127,18.0341952315.
Therefore, since the accident happened on Tuesday, we decided to consider the other three Tuesdays (the 20th of September, the 4th of October and the 11th of October). This decision was made because this data covers more days. Thus, it contains much more information than the first dataset. Furthermore, we implied that same kind of days (Mondays, Tuesdays, etc..) should have highly similar trends. So, if a change happened, it should be

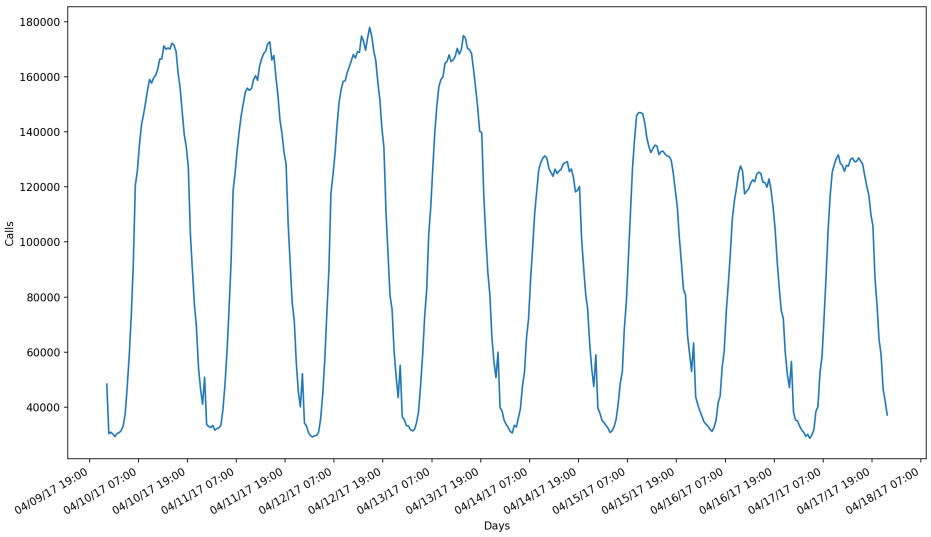

Figure 7.6: General activity during the Easter week, 30 minutes time frame

readily visible.

In this incident, we used the NGPRS table only, since the location of the antenna was not found in the MSC table.
First of all, as in the previous paragraph, we plotted the four days singularly and together using thirty minutes time interval (the fifteen minutes interval looks very similar), as can be seen in the Figures 7.15.
Secondly, we plotted the four days individually: Figure 7.16, 7.17, 7.18 and 7.19.

As can be seen from the general plotting, there is considerable more activity on the day of the accident. Moreover, on the 27th of September, there are more than 600 calls between 5 PM and 8 PM, which is approximately 100 more than the remaining days. Therefore, we expected to find a Change Point on that day, the hours after the disaster.
Following this supposition, we further our investigation by applying the algorithm on both time interval.

The outcome obtained seems very encouraging since the Change Point detected by our method is on 27th of September, at around 8 PM.

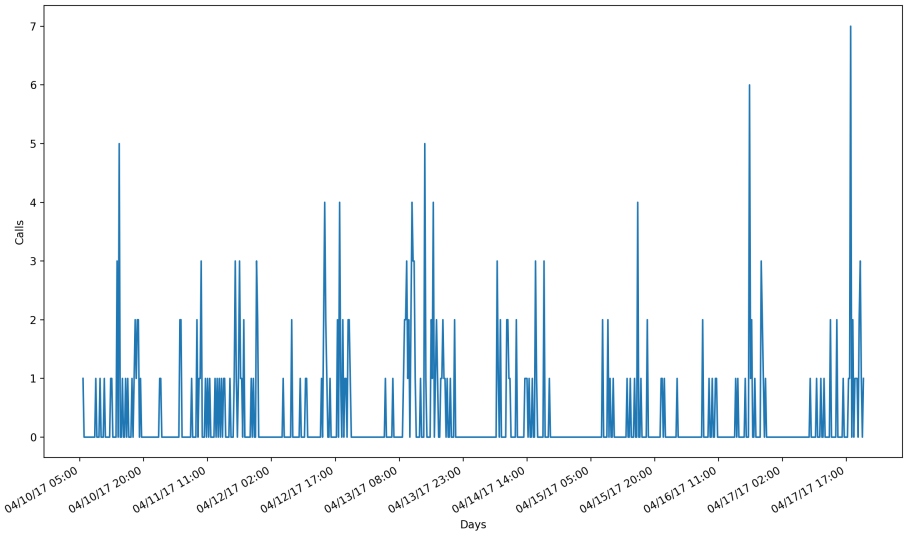

Figure 7.7: Activity on the 47.9126656221,17.1678863103 antenna during the Easter week, 15 minutes time frame

7.3 Discussion

The plotting showed that our Time Series are seasonal. Namely, they changed based on the time of the day. More specifically, the peak of calls and messages is reached during the afternoon, and the lowest point is attained between late night and early morning. Our belief is that, if something happens is most likely spotted by observing the change of the mean. The experiments made with the two datasets reveal very encouraging results. The traffic jam happened on the 13th of April 2017, created a severe change in the message and call activity, which it was spotted by the algorithm. Also, the general activity of the whole week reveals a Change Point, which was detected by our method. Likewise, the car accident happened on the same day (13th of April) created a spike in the activity. However, our method might be not always precise, and might not work impeccably with a one-week data. That is the reason why we decided to try a different approach with the second dataset, by taking the same day each week (Tuesday). Another important reason is, since the second dataset covers one month, it has with much more information. The accident we identified seemed to be a tough disaster, which generates a peak on the activity in the tower cell nearby. As expected, this drastic change was detected by our method, proving that it works well these types of transformation of the trend. Lastly, serious accidents and big traffic jams create a spike in the activity in Time Series data, since, on these occasions, vehicles have to stay in the same place, or else, they move extremely slowly. Consequently, much more

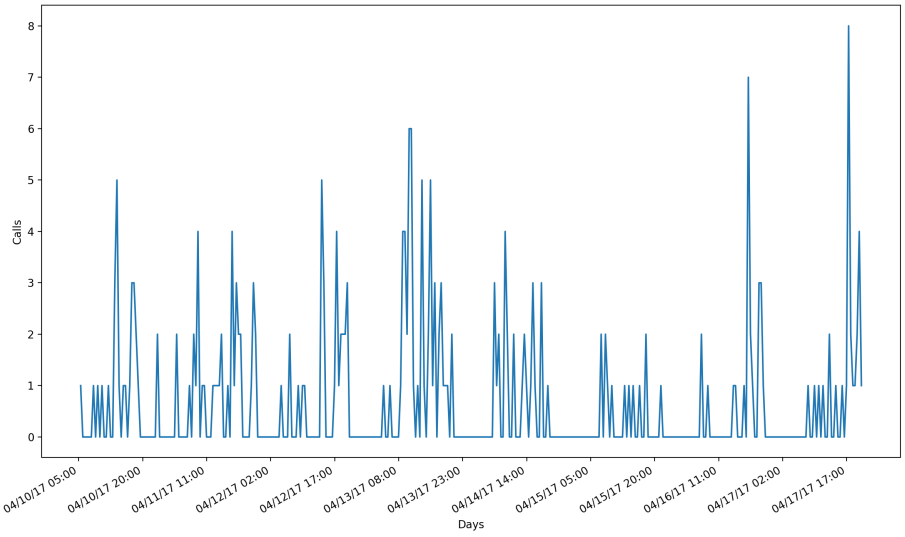

Figure 7.8: Activity on the 47.9126656221,17.1678863103 antenna during the Easter week, 30 minutes time frame

phone users are connected to the same tower cells, causing an increased workflow in it.

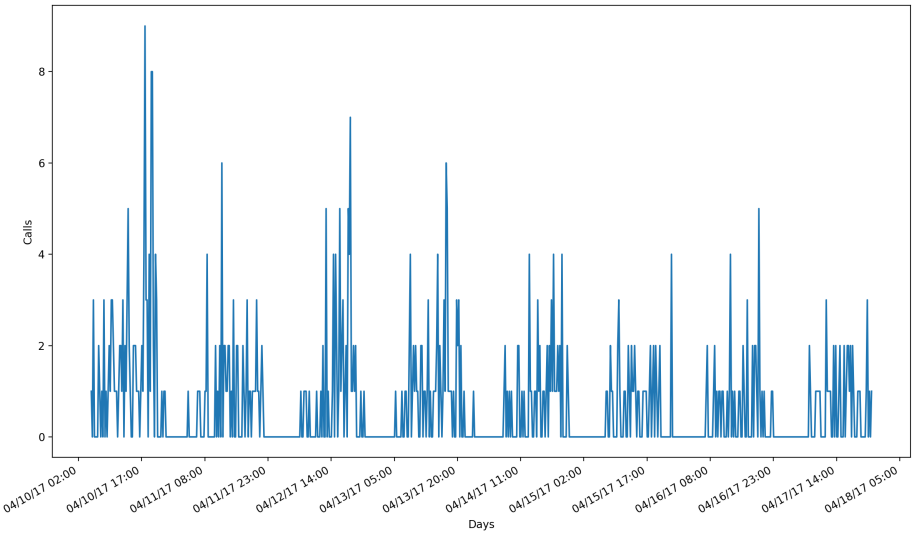

Figure 7.9: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, MSC table, 15 minutes time frame

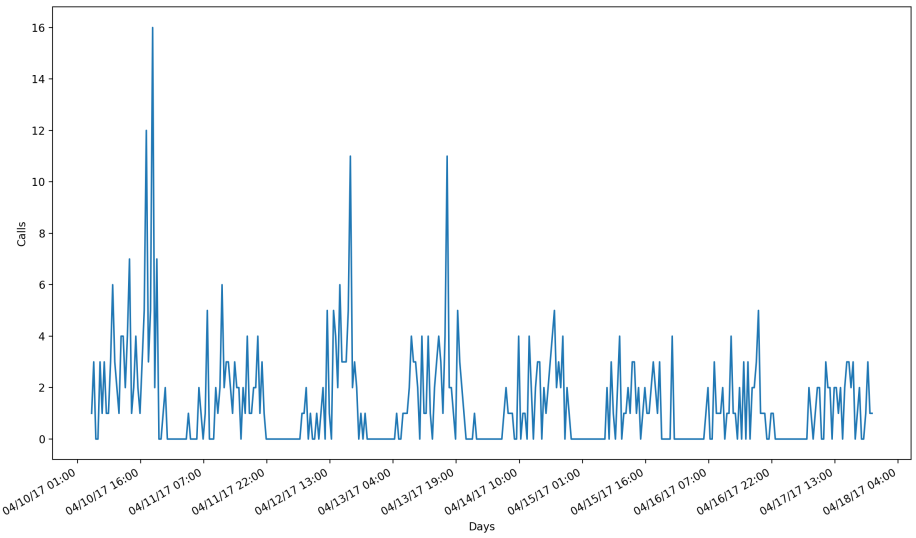

Figure 7.10: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, MSC table, 30 minutes time frame

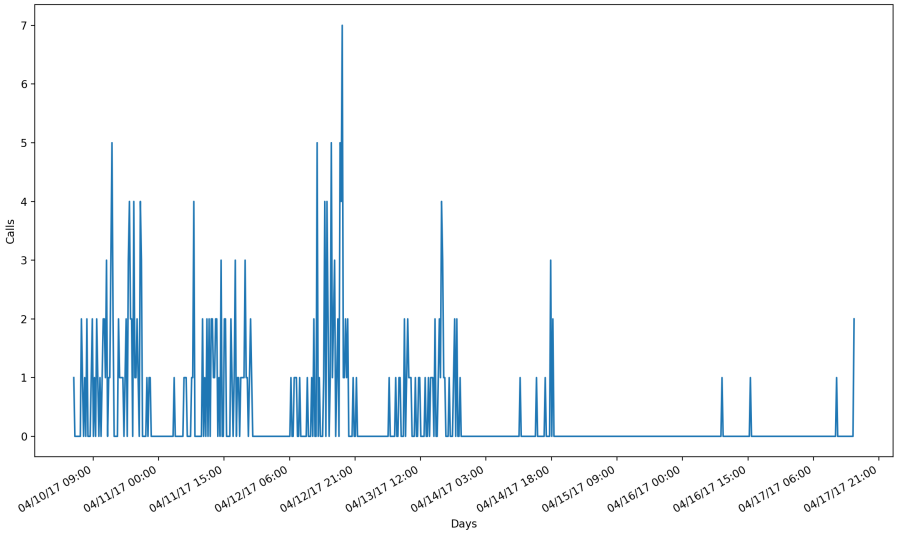

Figure 7.11: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, NGPRS table, 15 minutes time frame

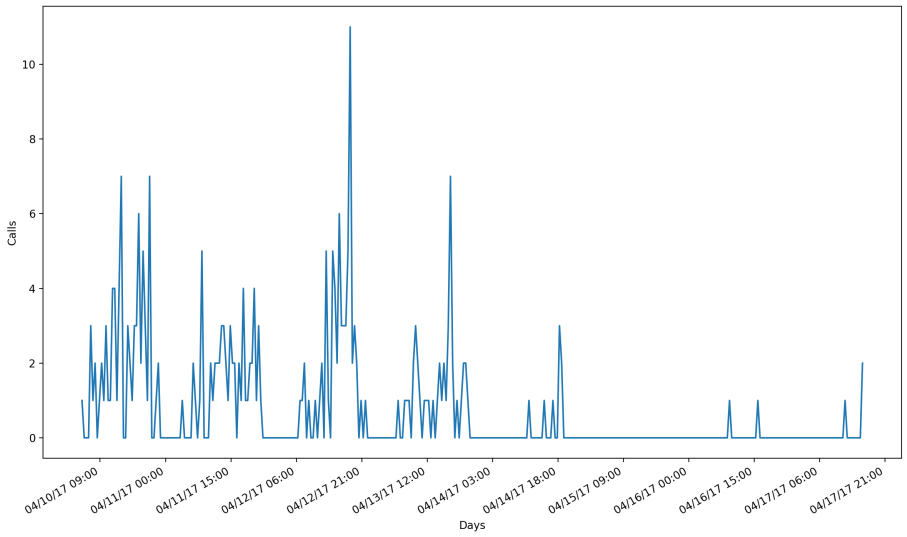

Figure 7.12: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, NGPRS table, 30 minutes time frame

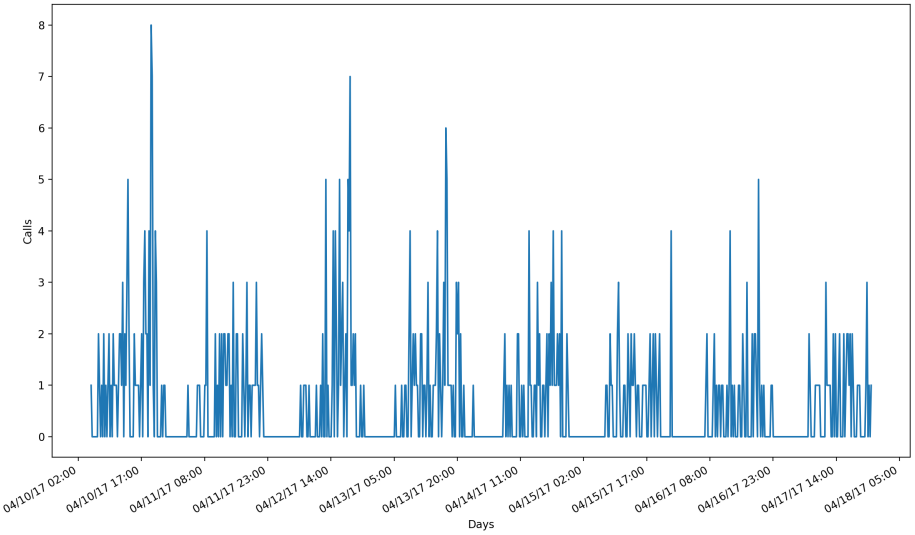

Figure 7.13: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, 15 minutes time frame

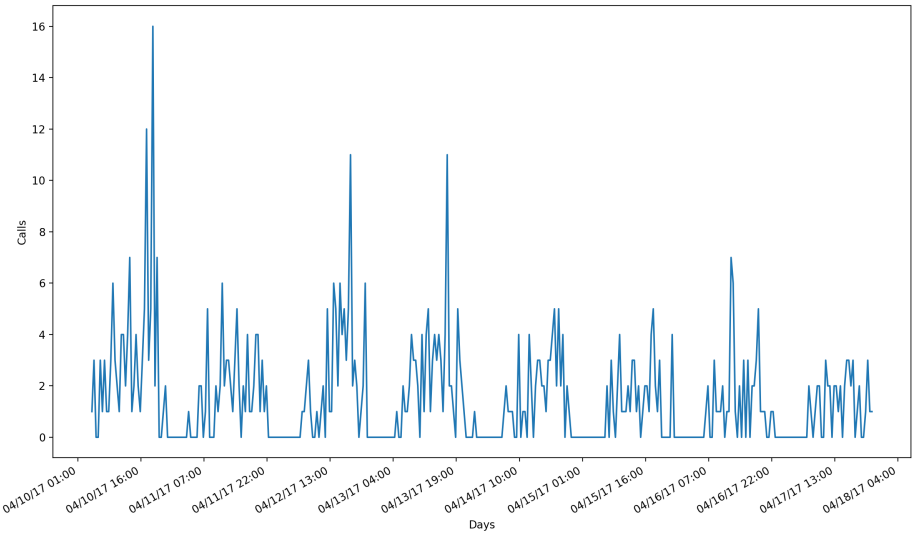

Figure 7.14: Activity on the 47.6414444197,19.5261483162 antenna during the Easter week, 30 minutes time frame

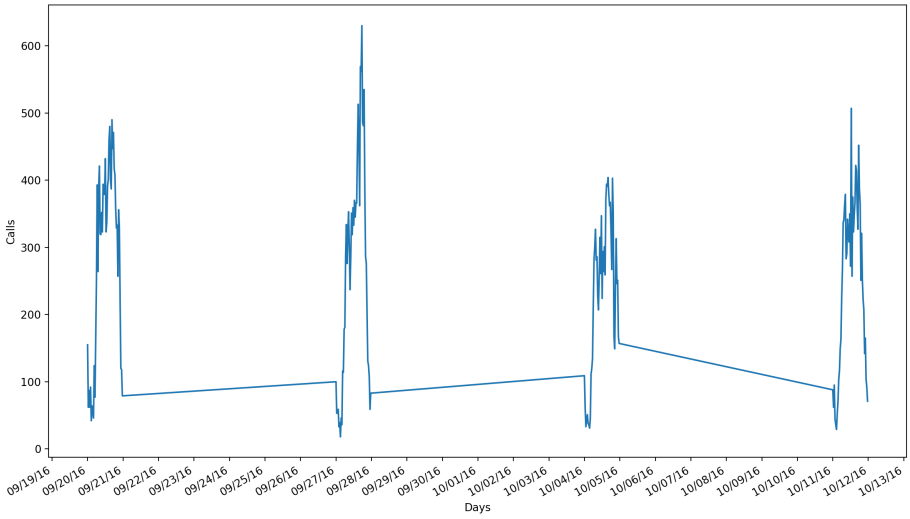

Figure 7.15: Activity on the 47.6819212317,18.0099900000 antenna during in four days, 30 minutes time interval

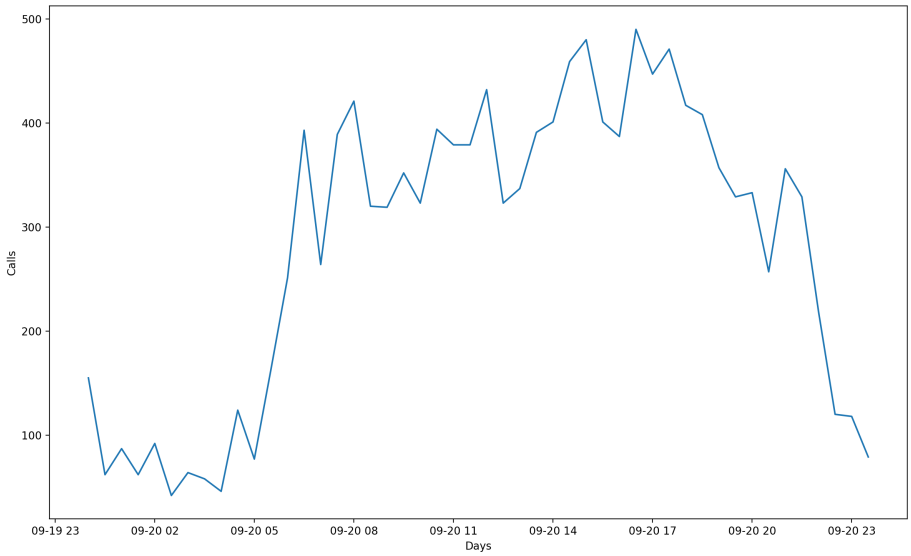

Figure 7.16: 20th of September 2016, 30 minutes time interval

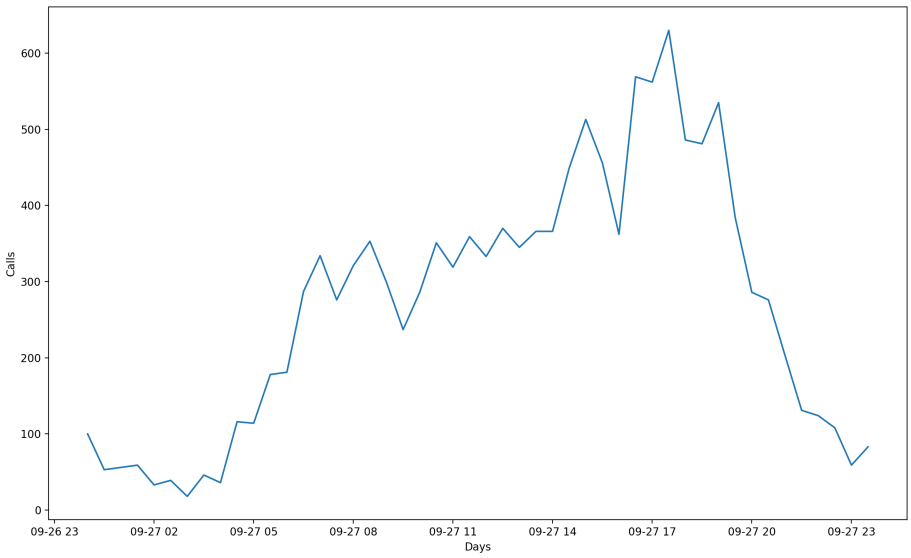

Figure 7.17: 27th of September 2016, 30 minutes time interval

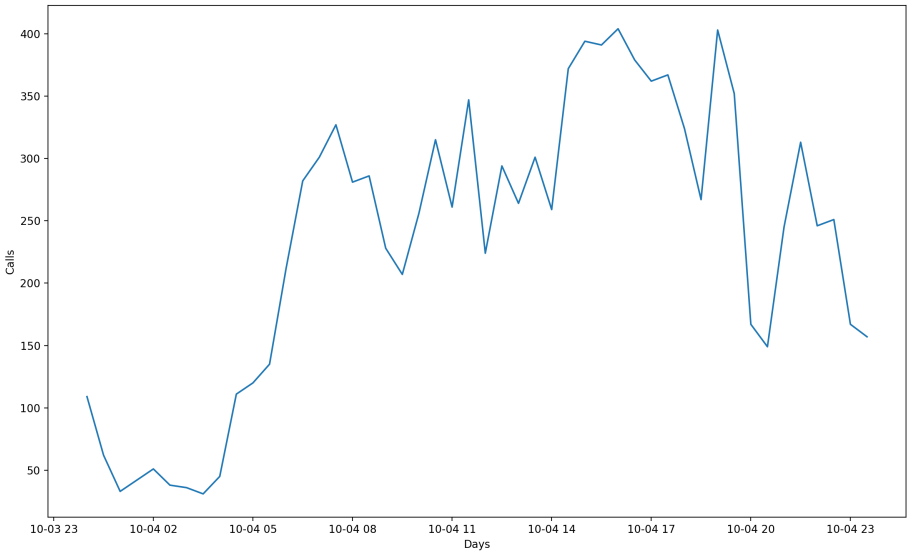

Figure 7.18: 04th of October 2016, 30 minutes time interval

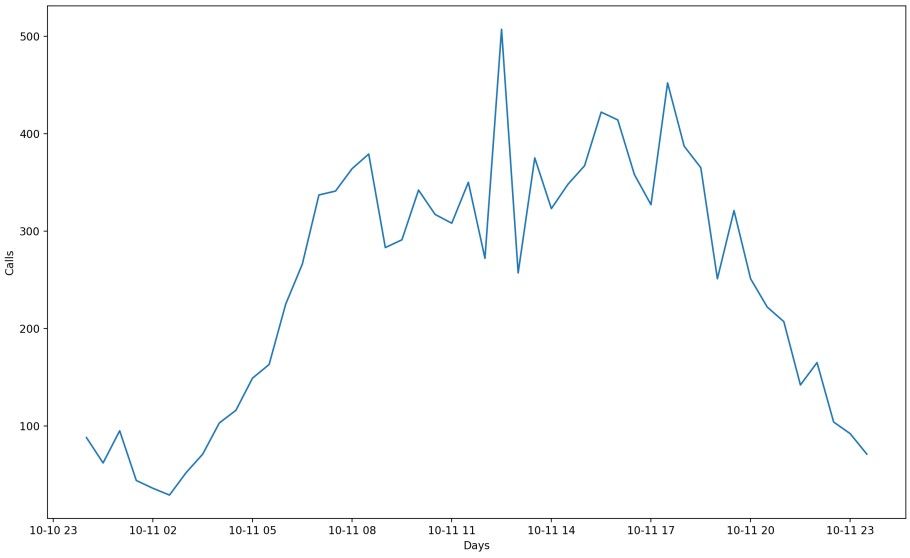

Figure 7.19: 11th of October 2016, 30 minutes time interval

Chapter 8

Conclusions

Anomaly detection is a useful matter which has been researched for many years by now. Moreover, unusual behavior in Time Series data is considered important at a similar degree. In fact, there exist several applications related to Anomaly Detection in Time Series, such as detecting anomalous heart beat pulses using ECG data.

As explained in the State of the Art chapter, telecommunications companies, such as Telekom or Vodafone, can provide highly important data, valuable for business purposes and scientific research. This information is called Call Detail Records (CDR), which is essential, log data of calls and messages made by customers, recorded in the antennas where the users are connected. Moreover, the research done using CDR is beneficial, since, it provides useful information concerning some unusual behavior and human mobility. For instance, it can determine the population census from a country; it can understand how people behave in a critical situation, such as traffic accident, or any other calamity, like earthquakes.

In our work, we processed the CDR data provided by *Magyar Telekom*. The dataset is portrayed with Time Series representation. Therefore, our focus was on Anomaly Detection on Time Series data. After giving an explanation about Time Series, we identified a method to detect these anomalies using the Change Point technique. This approach is based on the idea that if the mean changed considerably, hence something most likely happened. We used the algorithm introduced by Taylor (2000a) [43], where the Cumulative Sum, Bootstrap Analysis were used. We worked with two datasets, which covers two different periods: the first contains information happened during the Easter week in 2017, while the second includes data recorded during the second half of September 2016 and the first half of October 2016. Moreover, we identified some car accidents, and congestion occurred in those days.

After the data cleaning, the data preparation necessary, we were ready to test the method

selected, which, apparently, it was not yet tested with the CDR dataset. The results obtained are encouraging, since the algorithm could detect a substantial change in the days of the accidents, and, more importantly after the hour in which they occurred. We, therefore, imply that the method analyzed is performing fairly well. Consequently, we wish to continue our research in this direction. One weakness we have found is the accuracy. In fact, we believe that the method utilized can be improved, by adding, for instance, thresholds, such as deviation standard, or control limits. Also, another kind of estimators to detect the change can be used.

8.1 Future work

Anomaly detection is a vast research field, where several algorithms have been tested. Also, the study concerning the Call Detail Records data is at a good stage, where a lot of discoveries has been made. Nevertheless, there is a considerable more effort to be done. The future direction we are going to propose is the following.
Our work can be extended, by using a different type of Time Series Aggregation and discretization, such as the one proposed by Lin et al., in order to see if the results differ from the outcome presented in this thesis. Moreover, there are many approaches to find some sort anomalies on Time Series data. One method which can be used is the window-based procedure, where data from the same time frame (days, week, or month) are compared using the Dynamic Time Warping (DTW) algorithm, proposed by Professor Eamonn Keogh, which returns the degree of closeness between two Time Series.
Furthermore, the second point of the research was not implement it in practice, because of time constraint. However, we would like to continue the research, since we believe that the Dijkstra's algorithm in combination with other technologies, such as Google Maps API, can have several more applications than the ones presented on the Paragraph 5.3.

Bibliography

[1] Jyrki Akkanen and Jukka K Nurminen. "Case study of the evolution of routing algorithms in a network planning tool". In: *Journal of Systems and Software* 58.3 (2001), pp. 181–198.

[2] James P. Bagrow, Dashun Wang, and Albert-László Barabási. "Collective Response of Human Populations to Large-Scale Emergencies". In: *PLoS ONE* 6.3 (Mar. 2011). Ed. by Yamir Moreno, e17680. DOI: 10.1371/journal.pone.0017680. URL: https://doi.org/10.1371/journal.pone.0017680.

[3] Taposh Banerjee and Venugopal V Veeravalli. "Data-efficient minimax quickest change detection with composite post-change distribution". In: *IEEE Transactions on Information Theory* 61.9 (2015), pp. 5172–5184.

[4] Taposh Banerjee and Venugopal V Veeravalli. "Data-efficient quickest change detection in minimax settings". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6917–6931.

[5] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*. Vol. 104. Prentice Hall Englewood Cliffs, 1993.

[6] Ramzi BENAICHA and Mahmoud TAIBI. "DIJKSTRA ALGORITHM IMPLEMENTATION ON FPGA CARD FOR TELECOM CALCULATIONS". In: ().

[7] Pierre Raphael Bertrand, Mehdi Fhima, and Arnaud Guillin. "Off-line detection of multiple change points by the filtered derivative with p-value method". In: *Sequential Analysis* 30.2 (2011), pp. 172–207.

[8] Vincent D Blondel, Adeline Decuyper, and Gautier Krings. "A survey of results on mobile phone datasets analysis". In: *EPJ Data Science* 4.1 (2015), p. 10.

[9] Markus M. Breunig et al. "LOF". In: *ACM SIGMOD Record* 29.2 (June 2000), pp. 93–104. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388.

[10] Francesco Calabrese, Giusy Di Lorenzo, and Carlo Ratti. "Human mobility prediction based on individual and collective geographical preferences". In: *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE. 2010, pp. 312–317.

[11] Francesco Calabrese et al. "The geography of taste: analyzing cell-phone mobility and social events". In: *International Conference on Pervasive Computing*. Springer. 2010, pp. 22–37.

[12] *Cell Phone Tracking: Trends in Cell Site Precision*. 2013. URL: `https://www.cdt.org/files/file/cell-location-precision.pdf`.

[13] Balázs Cs Csáji et al. "Exploring the mobility of mobile phone users". In: *Physica A: statistical mechanics and its applications* 392.6 (2013), pp. 1459–1473.

[14] Frédéric Desobry, Manuel Davy, and Christian Doncarli. "An online kernel change detection algorithm". In: *IEEE Transactions on Signal Processing* 53.8 (2005), pp. 2961–2974.

[15] Pierre Deville et al. "Dynamic population mapping using mobile phone data". In: *Proceedings of the National Academy of Sciences* 111.45 (2014), pp. 15888–15893.

[16] Robert Dial et al. "Shortest path forest with topological ordering: An algorithm description in SDL". In: *Transportation Research Part B: Methodological* 14.4 (1980), pp. 343–347.

[17] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[18] Allen B Downey. "A novel changepoint detection algorithm". In: *arXiv preprint arXiv:0812.1237* (2008).

[19] Jean-Marie Dufour. *Introduction to time series analysis*. Jan. 2014. URL: `https://www2.cirano.qc.ca/~dufourj/Web_Site/ResE/Dufour_2008_C_TS_IntroductionTS.pdf`.

[20] Peter W Eklund, Steve Kirkby, and Simon Pollitt. "A dynamic multi-source Dijkstra's algorithm for vehicle routing". In: *Intelligent Information Systems, 1996., Australian and New Zealand Conference on*. IEEE. 1996, pp. 329–333.

[21] Alexandre X Falcão, Jorge Stolfi, and Roberto de Alencar Lotufo. "The image foresting transform: Theory, algorithms, and applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.1 (2004), pp. 19–29.

[22] Liang Gao et al. "Quantifying information flow during emergencies". In: *arXiv preprint arXiv:1401.1274* (2014).

[23] Jan Grandell. *Time Series Analysis*. Jan. 2014. URL: https://www.math.kth.se/matstat/gru/sf2943/ts.pdf.

[24] Pierre Granjon. "The cusum algorithm a small review". In: *Gipsa-Lab, Grenoble, France, Team SAIGA* (2012).

[25] Valery Guralnik and Jaideep Srivastava. "Event detection from time series data". In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*. ACM Press, 1999. DOI: 10.1145/312129.312190. URL: https://doi.org/10.1145/312129.312190.

[26] Alexander John Hartemink. "Principled computational methods for the validation and discovery of genetic regulatory networks". PhD thesis. Massachusetts Institute of Technology, 2001.

[27] Muhammad Imdadullah. *Time Series Analysis and Forecasting*. Jan. 2014. URL: http://itfeature.com/time-series-analysis-and-forecasting/time-series-analysis-forecasting.

[28] Jun Inagaki, Miki Haseyama, and Hideo Kitajima. "A genetic algorithm for determining multiple routes and its applications". In: *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*. Vol. 6. IEEE. 1999, pp. 137–140.

[29] Shan Jiang et al. "A review of urban computing for mobile phone traces: current methods, challenges and opportunities". In: *Proceedings of the 2nd ACM SIGKDD international workshop on Urban Computing*. ACM. 2013, p. 2.

[30] M Leyzorek et al. "Investigation of Model Techniques–First Annual Report–6 June 1956–1 July 1957–A Study of Model Techniques for Communication Systems". In: *Case Institute of Technology, Cleveland, Ohio* (1957).

[31] Jessica Lin et al. "A symbolic representation of time series, with implications for streaming algorithms". In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM. 2003, pp. 2–11.

[32] Xin Lu et al. "Approaching the limit of predictability in human mobility". In: *Scientific reports* 3 (2013).

[33] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. "Event Detection in Marine Time Series Data". In: *KI 2015: Advances in Artificial Intelligence*. Springer International Publishing, 2015, pp. 279–286. DOI: 10.1007/978-3-319-24489-1_24. URL: https://doi.org/10.1007/978-3-319-24489-1_24.

[34] ES Page. "Continuous inspection schemes". In: *Biometrika* 41.1/2 (1954), pp. 100–115.

[35] ES Page. "On problems in which a change in a parameter occurs at an unknown point". In: *Biometrika* 44.1-2 (1957), pp. 248–252.

[36] Santi Phithakkitnukoon, Zbigniew Smoreda, and Patrick Olivier. "Socio-geography of human mobility: A study using longitudinal mobile phone data". In: *PloS one* 7.6 (2012), e39253.

[37] Dominique Picard. "Testing and estimating change-points in time series". In: *Advances in applied probability* 17.04 (1985), pp. 841–867.

[38] Shiblee Sadik and Le Gruenwald. "Online outlier detection for data streams". In: *Proceedings of the 15th Symposium on International Database Engineering & Applications - IDEAS '11*. ACM Press, 2011. DOI: 10.1145/2076623.2076635. URL: https://doi.org/10.1145/2076623.2076635.

[39] *Second accident*. 2017. URL: http://www.police.hu/hirek-es-informaciok/legfrissebb-hireink/kozlekedesrendeszet/baleset-az-m3-as-autopalyan-29.

[40] PETER SHOUBRIDGE et al. "DETECTION OF ABNORMAL CHANGE IN A TIME SERIES OF GRAPHS". In: *Journal of Interconnection Networks* 03.01n02 (Mar. 2002), pp. 85–101. DOI: 10.1142/s0219265902000562. URL: https://doi.org/10.1142/s0219265902000562.

[41] Vasilios A Siris and Fotini Papagalou. "Application of anomaly detection algorithms for detecting SYN flooding attacks". In: *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*. Vol. 4. IEEE. 2004, pp. 2050–2054.

[42] Chaoming Song et al. "Modelling the scaling properties of human mobility". In: *Nature Physics* 6.10 (2010), pp. 818–823.

[43] Wayne A Taylor. "Change-point analysis: a powerful new tool for detecting changes". In: *preprint, available as http://www. variation. com/cpa/tech/changepoint. html* (2000).

[44] *Third accident*. 2016. URL: http://www.kisalfold.hu/komarom/ket_sulyos_baleset_-_meghalt_egy_utas_mentohelikopter_vitt_egy_sofort_a_gyori_korhazba/2491023/.

[45] *Time series components*. 2013. URL: https://www.otexts.org/fpp/6/1.

[46] Li Wei et al. "Assumption-Free Anomaly Detection in Time Series." In: *SSDBM*. Vol. 5. 2005, pp. 237–242.

[47] Faber Henrique Z Xavier et al. "Understanding human mobility due to large-scale events". In: *Third international conference on the analysis of mobile phone datasets (NetMob)*. 2013.

[48] Xiaopeng Xi et al. "Fast time series classification using numerosity reduction". In: *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, 2006. DOI: 10.1145/1143844.1143974. URL: https://doi.org/10.1145/1143844.1143974.

[49] Pengxiang Zhong. *The algorithm application for telecommunication routing*. URL: https://courses.cs.washington.edu/courses/csep521/07wi/prj/zhong.pdf.

Appendices

```python
#!/usr/bin/python3
import csv
import pandas as pd
import numpy as np

files = ["hackathon_msc.out.20170410", "hackathon_msc.out.20170411",
"hackathon_msc.out.20170412", "hackathon_msc.out.20170413",
    "hackathon_msc.out.20170414", "hackathon_msc.out.20170415",
    "hackathon_msc.out.20170416", "hackathon_msc.out.20170417"]

#Output file
end_file = "hackathon.msc.out.csv"
out_file = open(end_file, "w")

#File fieldnames
fieldnames = ['DateTime', 'CallID', 'CellID']
writer = csv.DictWriter(out_file, fieldnames=fieldnames)
writer.writeheader()

for onefile in files:
    with open(onefile, "rt") as file:
        reader = csv.reader(file)
        for row in reader:
            for r in row:
                split = r.split(";")
                writer.writerow({'DateTime': split[5], 'CallID': split[0], 'CellID': split[8]})

print("MSC parsed")

print("Cleaned")

df = pd.read_csv(end_file)

#sort csv file by timestamp
sorted_obj = df.sort_values(['DateTime'], ascending=[True])
#remove rows with null values
sorted_obj.dropna(subset=['CellID'], inplace=True)
print("Sorted")

df_index = sorted_obj.set_index(pd.DatetimeIndex(sorted_obj['DateTime']))

#group datetime hours and count the number of calls
df_grouped = df_index.groupby(pd.TimeGrouper(freq='30Min')).CallID.count()
print("Grouped")

grouped_file = "grouped_msc.csv"
grouped_out_file = open(grouped_file, "w")

fieldnames = ['DateTime', 'Calls']
groupedWriter = csv.DictWriter(grouped_out_file, fieldnames=fieldnames)
groupedWriter.writeheader()

for i, row in df_grouped.iteritems():
    groupedWriter.writerow({'DateTime': i, 'Calls': row})

print("grouped_msc.csv created")
```

Figure 1: Data cleaning and aggregation on the MSC table

```python
#!/usr/bin/python3
import csv
import pandas as pd

lat = "47.6415373874"
long = "17.4154518223"

coord_file = lat + "_" + long + "_ngprs.csv"
out_coord_file = open(coord_file, "rt")

output_file = lat + "_" + long + "_ngprs_cleaned.csv"
cleaned_out_file = open(output_file, "w")

fieldnames = ['DateTime', 'CallID']
writer = csv.DictWriter(cleaned_out_file, fieldnames=fieldnames)
writer.writeheader()

out_coord_file.readline()
for row in out_coord_file:
    split = row.split(",")
    dateTime = split[0]
    callID = int(float(split[1]))
    writer.writerow({'DateTime': dateTime, 'CallID': callID})

df = pd.read_csv(coord_file)

df_index = df.set_index(pd.DatetimeIndex(df['DateTime']))

#group datetime hours and count the number of calls
df_grouped = df_index.groupby(pd.TimeGrouper(freq='15Min')).CallID.count()
print("Grouped")

grouped_file = lat + "_" + long + "_ngprs_15_grouped.csv"
grouped_out_file = open(grouped_file, "w")

fieldnames = ['DateTime', 'Calls']
groupedWriter = csv.DictWriter(grouped_out_file, fieldnames=fieldnames)
groupedWriter.writeheader()

for i, row in df_grouped.iteritems():
    groupedWriter.writerow({'DateTime': i, 'Calls': row})

print("candidate_grouped.py done!!")
```

Figure 2: Data aggregation

```python
#!/usr/bin/python3
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.dates as mdates

days = mdates.DayLocator()
dateparser = lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')
groupedDf = pd.read_csv("grouped.csv", parse_dates=[0], date_parser=dateparser, index_col=[0])

x = groupedDf.index

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
plt.plot(x, groupedDf['Calls'])

ax.xaxis.set_major_locator(mdates.DayLocator(interval=1))
ax.xaxis.set_major_locator(mdates.MinuteLocator(interval=720))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%D %H:00'))
plt.xlabel('Days')
plt.ylabel('Calls')

fig.autofmt_xdate()

plt.show()
```

Figure 3: Code snippet for plotting the tower cell selected

```python
#!/usr/bin/python3
import pandas as pd
import numpy as np
import math as ma
import matplotlib.pyplot as plt
import pylab as pl

dateparser = lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S')
groupedDf = pd.read_csv("47.6819212317_18.0099900000_30_grouped.csv", parse_dates=[0], date_parser=dateparser)

# compute the mean
mean = groupedDf['Calls'].mean()

# compute the cumsum
cumsum = []
cumsum.append(0)
i = 1
for index, row in groupedDf.iterrows():
    x = float(row['Calls'])
    cumsum.append(cumsum[i-1] + (x-mean))
    i += 1

assert(np.isclose(cumsum[-1], 0.0))

# compute cumsum max and min
cumsum_max = max(cumsum)
cumsum_min = min(cumsum)
cumsum_max_i = cumsum.index(cumsum_max)

# compute cumsum diff
cumsum_diff = cumsum_max - cumsum_min

# convert list to array
calls_arr_or = np.array(groupedDf['Calls'])
calls_arr = np.array(groupedDf['Calls'])

# perform bootstrap analysis
count = 0
for index in range(1000):
    new_cumsum = []
    new_cumsum.append(0)
    i = 1

    new_calls_arr = np.random.choice(calls_arr, groupedDf['Calls'].size, replace=False)
    for item in new_calls_arr:
        new_cumsum.append(new_cumsum[i-1] + (item-mean))
        i += 1

    # compute cumsum max and min
    new_cumsum_max = max(new_cumsum)
    new_cumsum_min = min(new_cumsum)
    new_cumsum_max_i = new_cumsum.index(new_cumsum_max)

    # cumpute cumsum diff
    new_cumsum_diff = new_cumsum_max - new_cumsum_min
    if(new_cumsum_diff<cumsum_diff):
        count += 1
```

Figure 4: Change point detection algorithm part 1

```python
# compute the confidence level
confidence_level = (count/1000)*100

# getting the index from the max cumsum index
# computing the first estimator
cumsum_max_est = max(abs(i) for i in cumsum)

ind = 0
max = 0
cumsum_max_i_est = 0
for item in cumsum:
    if(abs(item) > max):
        max = abs(item)
        cumsum_max_i_est = ind
    ind += 1

#print(cumsum_max_i_est)

for index,row in groupedDf.iterrows():
    if(cumsum_max_i_est == index+1):
        print(row['DateTime'])
```

Figure 5: Change point detection algorithm part 2

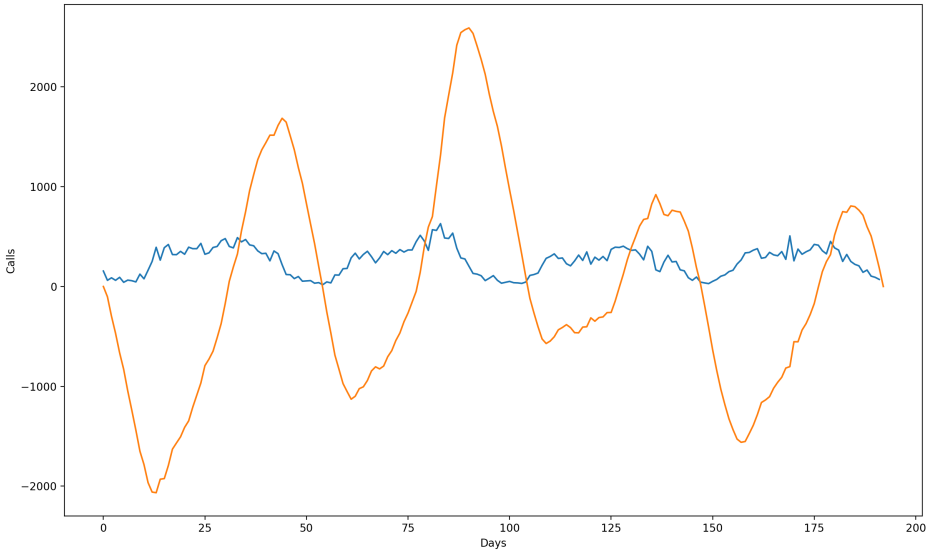

Figure 6: CUSUM Chart from the second dataset incident