Temporal dynamics clustering for analyzing cell behavior in mobile networks

Temporal Dynamics Clustering for Analyzing Cell Behavior in Mobile Network

Shuyang Li[a,*], Gianluca Francini[b], Enrico Magli[a]

[a] *Politecnico di Torino, Italy*
[b] *Telecom Italia, Italy*

Abstract

In recent years, management and resource allocation of mobile networks have become crucial because of a dramatic growth of mobile traffic. For Internet service providers, resource allocation is a challenging problem due to the inhomogeneous distributed nature of user demand, which increases the difficulty of configuring base stations in complicated urban environments. To manage mobile networks in an efficient way, understanding the typical usage behavior is very valuable for network operators. This paper aims at providing tools to analyze the behavior of mobile network cells in an urban environment. To achieve this goal, we propose a new time series clustering algorithm denoted as temporal dynamics clustering. Unlike conventional clustering approaches, this algorithm generates a new representation of time series by summarizing the distribution of the sequence of differences; in this way, temporal dynamics clustering creates clusters based on the variability of time series. We apply the proposed algorithm on a real-world mobile network dataset, showing its superior performance and much faster running time with respect to conventional methods such as hierarchical clustering and K-medoids. Using this approach, we have analyzed the cell behavior with two weeks observation collected in Turin, and obtained interpretable results, in that the behavior of each cell is strongly related to the characteristics of the area and the related human activity. We also study the cell behavior in different time slots by considering time-dependent characteristic of human activity. Through experiments, we show the effectiveness of the proposed algorithm at providing a deeper understanding of traffic usage patterns in intricate urban environments.

Keywords: Mobile networks, Time series clustering, Mobile traffic analysis

*Corresponding author
Email addresses: shuyang.li@polito.it (Shuyang Li), gianluca.francini@telecomitalia.it (Gianluca Francini), enrico.magli@polito.it (Enrico Magli)

1. Introduction

Over the past ten years, we have seen a dramatic growth of mobile traffic due to the rise of video streaming services over mobile devices. According to Cisco white paper, the total number of mobile subscribers will grow to 5.7 billion (71 percent of population) by 2023, and nearly 300 million mobile applications will be downloaded [1]; this massive growth of mobile demand makes the management of mobile networks more critical and challenging. The usage pattern of mobile networks is strongly related to human activities, which results in inhomogeneous distribution of user demand. Previous research [2] shows that the usage pattern is very different when the functionality of areas differ a lot. For example, the cellular towers located at office areas usually suffer heavier mobile demand than others during working time. Even when two cellular towers are located in the same area, it does not mean they are equally important, and it is quite naive to allocate to them a similar amount of network resources. For Internet Service Providers (ISPs), in order to manage mobile networks in an efficient way it is important to understand the behavior of traffic patterns. For gaining a detailed understanding of mobile demand, the attention is focused on cells of LTE mobile network, which can be seen as the most fundamental component of the access network layer. Access network layer is composed of elements called evolved Node B (eNodeB) or E-UTRAN Node B; eNodeB is responsible for connecting mobile devices within its coverage to the mobile network [3, 4]. Each eNodeB consists of multiple cells, and these cells act as connected targets for mobile devices. ISPs can update the configuration of cells constantly to characterize a mobile network; under this scenario, exploring the traffic pattern of cells provides valuable information for maintaining the network. If we want to obtain insights into cell behavior, it is essential to analyze the time series of data generated by cell. Based on what we mentioned previously, mobile demand distributes in an inhomogeneous way among cells, which means some cells serve more mobile users than others. In this case, the more heavily loaded cells have higher priority in the management of LTE network, and they should also be allocated more network resources. However, cell priority is not known in advance, hence the need of cell behavior analysis techniques, which typically employ unsupervised learning methods.

Unsupervised learning is the primary choice for extracting hidden patterns from data. Among different approaches, clustering is one of the most widely used methods in exploratory data analysis. Through cluster analysis, objects are split into different groups based on relative similarity, where each cluster groups items that are similar to each other. For identifying the cell behavior, cells have to be split into different groups on the basis of their temporal behavior, and the problem can be cast as time series clustering. In particular, the time series associated to each cell may contain the temporal evolution of several parameters of interest related to the cell, which can be used to determine the importance of the cell itself. Parameters may include the number of connected users, downlink usage, handover and others. In the time series clustering field, many works have been done in various industrial domains, and most of them focus on the

recognition of different waveforms and identification of anomalies within a time series. For example, [3] applies K-means to detect anomalies within mobile traffic time series. [5] uses different clustering methods to study the temporal pattern of energy consumption. Even though the waveforms and anomalies are important features of time series, they still do not provide enough information in our case. If we want to know the importance of cells in order to determine their priority, clusters need to be created by considering the variability of time series. In mobile network scenario, if a cell covers plenty of mobile devices, spikes and strong oscillations can be observed in its time series, which exhibits strong temporal dynamics. From this point of view, a time series clustering algorithm is expected to group series by their temporal dynamics level, since this is a very useful indicator of the importance of the cell. This requires to 1) identify suitable features that can capture the dynamic behaviour, and 2) couple them with a suitable backbone clustering method.

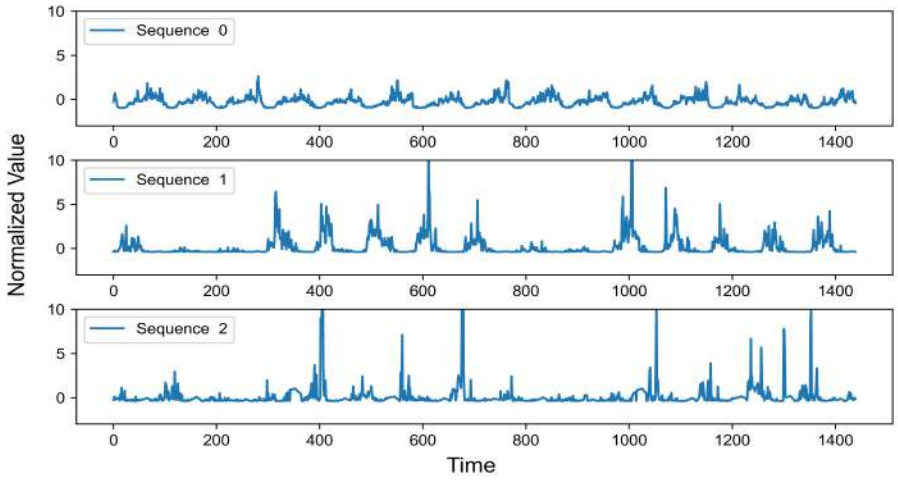


Figure 1: Downlink usage sequences with different oscillation levels.

To group time series by temporal dynamics, it is essential to determine how to measure this characteristic correctly. Since the definition of temporal dynamics is quite vague, it is difficult to find a criterion to evaluate it. In general, a time series which has strong temporal dynamics exhibits spikes and irregularities without a specific pattern. Conventional clustering approaches cannot be used to group time series by temporal dynamics because of the limitation of distance measure and computational cost. In time series clustering, Euclidean distance and Dynamic Time Warping (DTW) have been widely used to measure the similarity between sequences. DTW is first proposed in [6] and it is well-known for its invariance against signal shifting [7]. To measure the similarity between sequences, DTW first computes a pairwise distance matrix between points of two time series, and then a dynamic program following a recursive manner is used to find the minimum-cost alignment between them [8]; some

constraints can be set to guide the search of the optimal path. Compared to Euclidean distance, DTW does a better job at handling sequence misalignment. However, the complexity of DTW is quadratic, which limits the usability of DTW on medium and large datasets. As shown in Figure 1, it is obvious that time series 1 is more similar to time series 2 than to time series 0 in a semantic sense by observing the sequences. But if we calculate the Euclidean distance between these series, the distance between series 1 and series 0 is 56.28 while the distance between series 1 and series 2 is 69.30. If we perform clustering with this similarity measure, series 0 and series 1 should be grouped into the same cluster which is undesired. From this point of view, Euclidean distance is not always a good option for comparing temporal dynamics of time series. If the similarity is measured by DTW, in this case, series 1 is more similar to series 2, and this may lead to correct cluster creation. But the computational cost of DTW is very expensive, as its complexity increases quadratically with respect to the length of series. In mobile network scenario, a time series is usually very long, and that makes DTW impractical in mobile time series clustering. The problem of computational cost becomes more critical when the algorithm requires to calculate pairwise distances among all objects, such as K-medoids and hierarchical clustering. In this case, even though in principle a clustering algorithm can correctly compare temporal dynamics of time series, the complexity can render the method unusable. Based on the above-mentioned issues, a clustering algorithm should address two challenging problems: (1) How to define features that allow to compare temporal dynamics between time series? (2) How to reduce computational cost while measuring similarity correctly?

For addressing these issues, we propose a new feature-based clustering approach called Temporal Dynamics Clustering (TDC). This algorithm measures the variability of time series by comparing the distribution of the first-order differences sequences, and computational cost is greatly reduced by an operation called distribution summarization. To evaluate the performance of our algorithm, the proposed method is compared to the baselines which are shape-based approaches. Unlike the feature-based approach using the extracted features of original time series, the shape-based approach uses raw time series directly for clustering, and the features are the sequences of the considered variables [9]. The contributions of this paper are summarized as follows:

- We propose a new time series clustering algorithm to group time series based on their temporal dynamics with low cost, where clusters are created based on the degree of variability of time series. Our algorithm employs suitably defined features coupled with a conventional backbone clustering method; it is evaluated on real-world datasets collected from a mobile network, achieving remarkable performance compared to baselines. The generated clusters are better separated, and our algorithm is 100x faster than other clustering approaches, such as K-medoids and hierarchical clustering.

- Using TDC, we analyze the behavior of cells and obtain interpretable result. Our algorithm splits cells into three clusters, and several areas

are identified as important by observing the geographical distribution of clusters. The identified areas play important role in daily life, such as university, train station and historical district; this fills the gap of cell level analysis of mobile traffic.

This paper is structured as follows. In Section 2, we review previous works related to time series clustering and mobile network analysis. Section 3 presents the proposed algorithm, baselines and metrics for evaluating performance. In Section 4 we introduce two real-world datasets which are used to perform experiments. The proposed algorithm is compared with other baselines, and its behavior is analyzed in detail. We also provide a real usage case of our method which explores cell behavior in Turin, Italy. In Section 5, we draw some conclusions.

2. Related Work

2.1. Time series clustering

Time series clustering is the process of organizing time series into different groups based on a similarity metric. Compared to a simple data clustering problem, time series clustering is more challenging because of several reasons. First, a time series may contain a lot of data including multiple variables, each variable containing multiple records. This makes handling and processing time series difficult for many backbone clustering algorithms, and the speed of clustering process may decrease significantly. K-means is more computationally efficient than hierarchical clustering and K-medoids [10, 11], and it has been widely used in many works [12, 13, 14]. Another problem of time series clustering is the choice of the similarity measure; a time series is often noisy and it may include outliers and temporal shifts, which makes comparisons much more difficult [9]. Time series clustering is very popular for solving real-world problems, such as air quality analysis [15], identification of functionally related genes [16] and emotional behaviour analysis in social network [17]. Most of these works focus on analyzing waveforms or detecting anomalies of time series, but few of them study variability-oriented clustering analysis, which is important in the context of mobile cellular networks.

2.2. Analysis of mobile networks

In recent years, the management of mobile networks has become more challenging due to massive growth of mobile demand, especially when we lack the knowledge of usage pattern in complicated urban environments. For addressing this issue, extensive studies have been done to analyze the behavior of mobile networks. This analysis can be divided into two broad categories: (1) study of mobile user behavior and consumption of different mobile services and (2) study of mobile network behavior on the side of network operators. For user behavior analysis, [18] proposes a framework to detect anomalous telecom customer behavior and analyze users' usage patterns, in order to provide supplemental information for precision marketing of telecom operators and preventing criminal

behaviors. [19, 20] perform national scale analysis of user usage pattern. [20] focuses on the data usage patterns of geographically diverse mobile users meanwhile [19] studies the difference of mobile service consumption by considering various service categories, such as streaming, social networking, cloud service and gaming. For the study on network operator side, [3, 21] use K-means to detect mobile traffic anomalies by using call details record; these works focus on detecting outliers of each time series which aims to study the behavior of a single record instead of the behavior of sequence. Besides the clustering approach, [22] uses neural networks to detect anomalies within traffic sequences, but their dataset is very small as it only consists of data collected from three base stations. [2] is the closest study with our work; they use a huge dataset collected from Beijing to study the behavior of cellular towers in different areas. Through clustering analysis, they find that the traffic usage pattern is very different when the functionality of analyzed areas differs a lot; five functional areas are discussed, including residential area, transport area, office area, entertainment area and comprehensive area. However, this work only studies the difference between different usage patterns, and the importance of cells has not been discussed.

3. Methodology

3.1. TDC

As shown in Figure 2, TDC consists of three parts: generation of first-order differences sequence, distribution summarization and K-means clustering algorithm. The first two steps act as the feature extractor which is used to generate features for the subsequent clustering process; the feature extractor of conventional approaches is usually very simple, as these approaches use the samples of variables directly as the features for clustering. Unlike the conventional approaches, in TDC features are designed to summarize the dynamic behavior of the time series, and are then fed into the backbone algorithm (clustering engine) to create clusters.

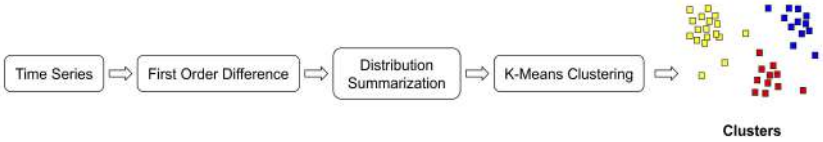
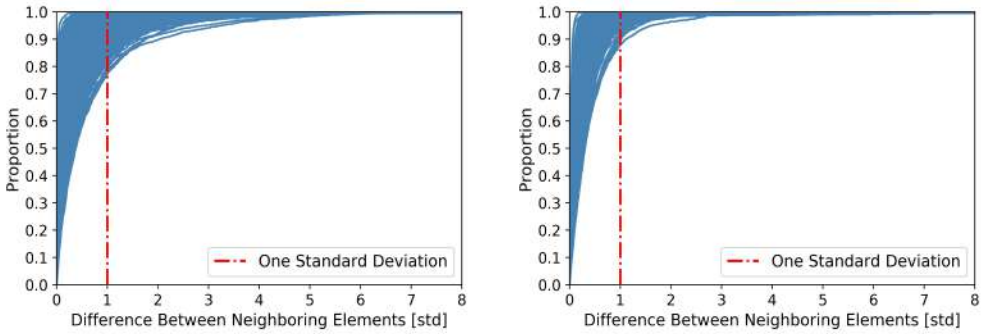


Figure 2: Workflow of TDC.

The aim of the first step is to measure the variation between adjacent elements; instead of using original time series, the first-order difference is calculated for series of each variable. The mathematical expression of the first-order difference is the following:

$$\Delta y_t = y_t - y_{t-1}, \tag{1}$$

where Δy_t is the difference at time t, and y_t and y_{t-1} are the values of the time series at time t and $t-1$. In this way, for each variable we convert the original time series into a sequence of differences, which provides a more natural way to measure local temporal dynamics.



(a) (b)

Figure 3: Cumulative Distribution Function (CDF) of two mobile network KPIs. (a) Downlink usage; (b) Average number of connected users.

The second step is called distribution summarization, which aims at summarizing the characteristics of the time series in a way that enables analysis of dynamics with a small computational cost. For clustering time series by temporal dynamics, distribution summarization should highlight the highly varying parts of differences sequence while retaining the global behavior. Figure 3 illustrates the CDF (Cumulative Distribution Function) of differences sequence of two important mobile network KPIs: the downlink usage and the average number of connected users. Observing the figures, we find the CDFs have similar behavior when the quantile is not so high, and the main difference between CDFs relates to the behavior of the tail. Long tail can be seen as an indicator of strong oscillation considering these CDFs are plotted based on the differences sequences instead of original series, which means the tail behavior encodes the "rare event" of each series. If we want to describe this distribution correctly, it is essential to summarize the tail behavior. However, this requires to define exactly what values belong to the tail. For addressing this issue, the proposed algorithm employs extreme value theory which is mainly used to estimate the tail distribution for analyzing rare events.

Extreme value theory studies the distribution of extreme realizations of a given distribution function, or of a stochastic process [23, 24]. It is widely used to study the behavior of the tail of the distribution, especially in financial and environmental fields. To estimate the tail distribution, it is essential to define the tail of the observed records. One of the most popular approaches is called Peaks-Over-Threshold, which requires a predefined threshold to find the tail part. There are different ways to select the threshold: [25, 26] discussed the

visualization-based approach. The main idea is to use graphical diagnostics tool such as Hill plot and the others. The graphical diagnostics approach requires manual check which can be affected by subjectivity and cannot be done automatically. Optimization-based approaches also exist, e.g. [27] proposed an approach which aims to minimize a cost function to find the best threshold. This method is too complicated as it requires hyperparameter tuning and increases the computational cost. In this paper, we follow a data-dependent rule proposed by [28] which is also used in [29, 30]. According to the rule, the largest u elements of differences sequence are defined as the tail part, and u is calculated based on the following formula:

$$u = \frac{(T-1)^{2/3}}{\log(\log(T-1))}, \tag{2}$$

where T is the length of original time series. Based on tail index u, a quantile index I_q is calculated to split the distribution into the main part and the tail part:

$$I_q = 1 - \frac{u}{T-1}, \tag{3}$$

$$\Delta Y^{main} = \{\Delta y_t \mid \forall t \in [1, T-1] \text{ and } \Delta y_t \leq Quantile(\Delta Y,\ I_q)\}, \tag{4}$$

$$\Delta Y^{tail} = \{\Delta y_t \mid \forall t \in [1, T-1] \text{ and } \Delta y_t > Quantile(\Delta Y,\ I_q)\}, \tag{5}$$

where $Quantile(\Delta Y,\ I_q)$ is the calculated quantile of differences sequence ΔY given quantile index I_q, and ΔY^{main} and ΔY^{tail} are the main part and the tail part of the series. After doing this, the distribution of time series is summarized as a new representation r:

$$r = [\ \mu^{main},\ \sigma^{main},\ \mu^{tail}\ ], \tag{6}$$

where μ^{main} is the mean value of ΔY^{main}, σ^{main} is the standard deviation of ΔY^{main} and μ^{tail} is the mean value of ΔY^{tail}. By summarizing the behavior of differences sequences, the algorithm generates a new representation for original time series which has much fewer elements, leading to a great reduction of computational cost of the subsequent clustering step. If the time series is multivariate, the representation should be computed individually for each variable, and the dimension of generated representation is $r \in R^{3 \cdot V}$ where V is the number of variables.

In the last step of TDC, the generated representation is used as input to clustering. To obtain reasonable results and perform clustering efficiently, the selection of clustering engine is crucial for TDC, and the backbone clustering

Algorithm 1 TDC

Input: Time Series $X = \{x_1, ..., x_N\}, x_N \in R^{T,V}$. The number of clusters k.
Result: Cluster Partition $P = \{C_1, ..., C_k\}$.

$u = \frac{(T-1)^{\frac{2}{3}}}{\log(\log(T-1))}$ ▷ Tail Index
$I_q = 1 - \frac{u}{T-1}$ ▷ Quantile Index
$R = [\ ]$ ▷ Empty Representation

for Object $n = 1$ to N **do**
  $r = [\ ]$
  for Variable $v = 1$ to V **do**
    S = Calculate First-Order Difference(x_n^v)
    Q = Calculate Quantile(S, I_q)
    $S^{main} = \{s_h : s_h \leq Q \text{ for } h = 1, ..., T-1\}$ ▷ Main Part
    $S^{tail} = \{s_h : s_h > Q \text{ for } h = 1, ..., T-1\}$ ▷ Tail Part
    $\mu^{main} = Mean(S^{main})$
    $\sigma^{main} = Std(S^{main})$
    $\mu^{tail} = Mean(S^{tail})$
    append μ^{main}, σ^{main}, μ^{tail} to r
  end for
  append r to R
end for

P = K-means(R, k) ▷ Run K-means with generated representation
return P

algorithm is selected among K-means, K-medoids and hierarchical clustering. Hierarchical clustering suffers from the weakness that it cannot undo what was done previously [11], i.e. the previous merging of sub-clusters cannot be modified even though we find the merging is not a good one later; moreover, its computational inefficiency makes it impractical to use on large datasets [9, 10]. K-means and K-medoids are similar approaches, but the time complexity of K-medoids is much higher than that of K-means [31], and this is undesirable for processing large time series. Based on these issues, K-means is chosen for TDC; the details of backbone algorithm comparison are discussed in Section 4.2. For applying K-means, the similarity between sequences is measured by Euclidean distance on the generated representation. For the entire workflow of TDC, the pseudo code is shown as Algorithm 1.

3.2. Baselines

For assessing the performance and complexity of TDC, we have selected three well-known methods: K-means, K-medoids and hierarchical clustering.

K-means has been discussed in previous section, K-medoids is similar to K-means, its target is to minimize the distance between intra-cluster object and the corresponding cluster centroid [32, 33]. Comparing to K-means, K-medoids does not calculate the centroid with intra-cluster objects, instead selecting an existing object as the centroid. In this way, K-medoids is more robust to outliers. Hierarchical clustering is an algorithm which builds a tree of clusters, the clusters at each level are joined to create the cluster at the next level based on the similarity measure. There are different linkage criteria to join the clusters; in this paper, we applied hierarchical clustering with average linkage which is widely used in applications. These three methods have been used employing the original time series as input; the objective is to assess the performance and complexity of these baseline approaches with respect to TDC.

3.3. Metrics

To evaluate the performance of clustering algorithms, we use four different criteria. The first one is the running time of each algorithm, which is a straightforward way to compare the execution time and computational cost. The other criteria are silhouette value [34], Davies–Bouldin Index [35] and Calinski-Harabasz Index [36]. All of them are used to measure the quality of the separation of clusters. Silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation), and it ranges from -1 to 1. The higher the silhouette value, the better the separation. Negative value means the clusters are separated in a wrong way. Davies–Bouldin Index is defined as the average similarity measure of each cluster with its most similar cluster, and the index ranges from 0 to ∞. A lower Davies–Bouldin Index indicates a better clustering result. Calinski-Harabasz Index measures the ratio of the sum of between-clusters dispersion and of intra-cluster dispersion for all clusters, and its value ranges from 0 to ∞; the higher the index, the better the clustering is.

4. Result

In this section, experiments are carried out and the related findings are discussed. We perform clustering on two datasets: we first use the larger dataset to determine the backbone algorithm of TDC and evaluate the performance of TDC and baselines with the aforementioned metrics, and then compare the behavior of TDC and K-medoids in detail. After that, a small dataset is used to visualize the clustering result of TDC and analyze usage pattern of cell in the city of Turin (Italy). To implement the baselines, we use the following libraries: K-means (Tslearn [37]), K-medoids (PyClustering [38]) and hierarchical clustering (Scikit-learn [39]). All experiments are conduct on a computer with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz and 64 GB memory.

4.1. Dataset

In our experiments, we use two real-world datasets, both of them collected and processed by Telecom Italia S.p.A which is the largest italian telecommunications services provider; the datasets have been preprocessed to deal with missing values and standardize data. The first dataset consists of 16300 multivariate time series, and the records are collected from 100 cells in LTE network. The length of each sequence equals 1440 samples collected in 15 days. The second dataset monitors the activity of cells over 2 weeks from 09 November 2020 to 22 November 2020. 956 multivariate time series are collected from different cells in Turin, and the length of each time series is 1344 samples. In this study, we are interested in two important network parameters: the downlink usage and the average number of connected users. Clustering algorithms are performed based on these two variables.

4.2. Comparison of Backbone Algorithms

As mentioned in Section 3.1, we evaluate the performance of various approaches to select the backbone clustering algorithm for TDC. The results are reported in Table 1.

Backbone Algorithm	Running Time	Silhouette	DB Index	CH Index
K-means Clustering	**4.65 mins**	0.435	0.785	**18417.76**
K-medoids Clustering	5.94 mins	0.423	0.789	18403.98
Hierarchical Clustering: Average Linkage	4.73 mins	**0.591**	**0.543**	4541.66
Hierarchical Clustering: Ward Linkage	4.74 mins	0.363	0.882	14519.89

Table 1: Comparison of backbone clustering algorithms.

We perform TDC with four backbone approaches: K-means, K-medoids, hierarchical clustering with average linkage and hierarchical clustering with ward linkage. From Table 1, hierarchical clustering with ward linkage is the worst one among these approaches, while the average linkage version has the highest silhouette score and DB index, but its CH index is much lower than the others. Comparing to K-means, K-medoids has similar performance of clustering metrics; however, the running time of K-medoids is much longer than K-means which is undesirable. According to these metrics, K-means and hierarchical clustering are good candidates as backbone method. To compare these algorithms in detail, Figure 4 illustrates the behavior of clusters generated by different algorithms. From the figures, K-means, K-medoids and hierarchical clustering with ward linkage generate similar clusters; these approaches split time series into different groups based on their temporal dynamics. If we perform hierarchical clustering with average linkage, we cannot identify temporal dynamics level from Figure 4(c); it is difficult to compare these clusters when their behavior differs a lot, especially one of them has sinusoidal shape. Based on metrics and cluster behavior, K-means is the best one and it is selected for TDC.

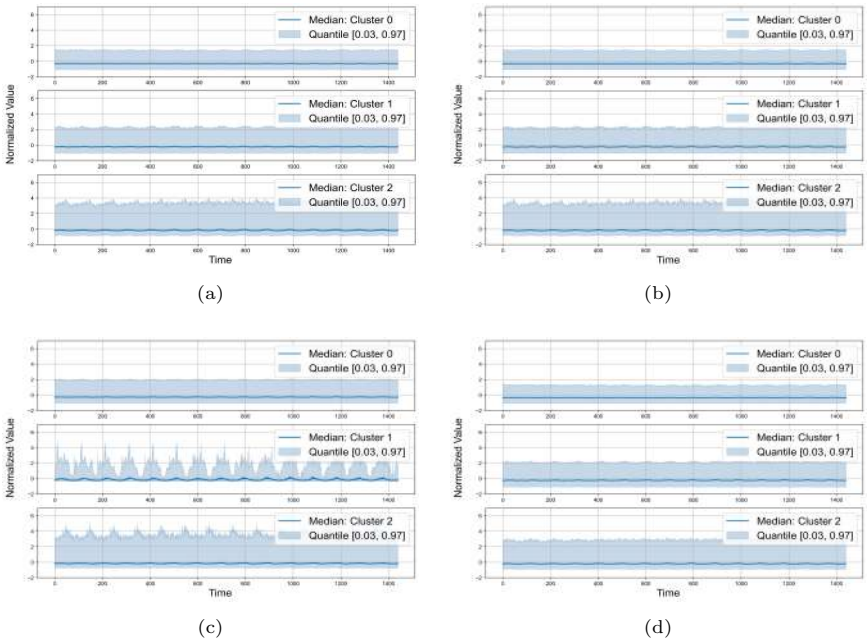


Figure 4: Comparison of backbone algorithms (time series of downlink usage are plotted). From top to bottom: cluster 0, cluster 1 and cluster 2. The blue line is the median value calculated within the cluster, and the shaded region indicates the upper bound 0.97 quantile and the lower bound 0.03 quantile of each cluster. (a) K-means; (b) K-medoids; (c) Hierarchical clustering: Average Linkage; (d) Hierarchical clustering: Ward Linkage.

4.3. Comparison of Clustering Approaches

In this subsection, all experiments have been done on the first dataset. We first apply TDC to generate three clusters. For studying the behavior of this algorithm, Figure 5 provides a visual representation of clusters. It can be seen that cluster 2 is the cluster whose temporal dynamics is the highest one. If we move from cluster 2 to cluster 0, the oscillation of time series becomes much weaker, and a decreasing trend of temporal dynamics is observed. According to the plots, cluster 2 is more important than the others for mobile networks, as the sequences belonging to this cluster suffer a heavier mobile demand because of the occurrence of spikes. In this case, it is very likely that spikes represent specific events which could be caused by user movement or social event. To have a comprehensive view of our algorithm, we compare the behavior of clusters generated by TDC and K-medoids clustering, and the result is shown in Figure 6. From Figure 6, the median value and variation range of each cluster are plotted to provide a global perspective of cluster behavior. Based on the figures, TDC generates clusters based on the variability of time series. Among three clusters, cluster 2 has the highest, and cluster 0 exhibits the weakest oscillation. Comparing with TDC, the behavior of K-medoids is very different. K-medoids generates very similar clusters, as the algorithm measures the similarity between

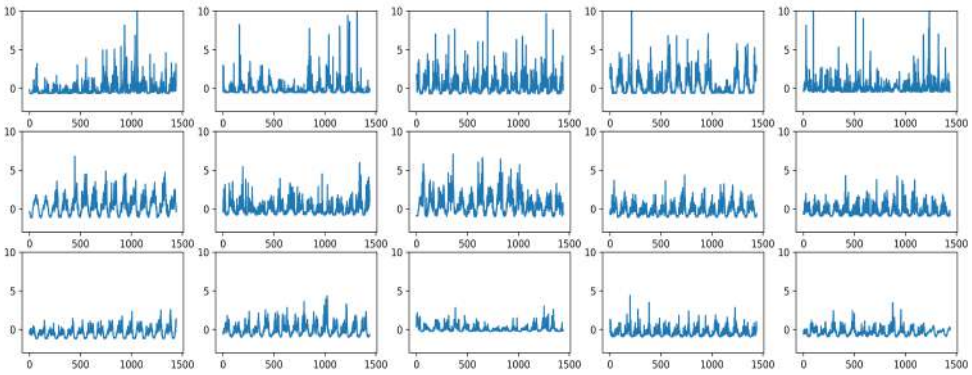


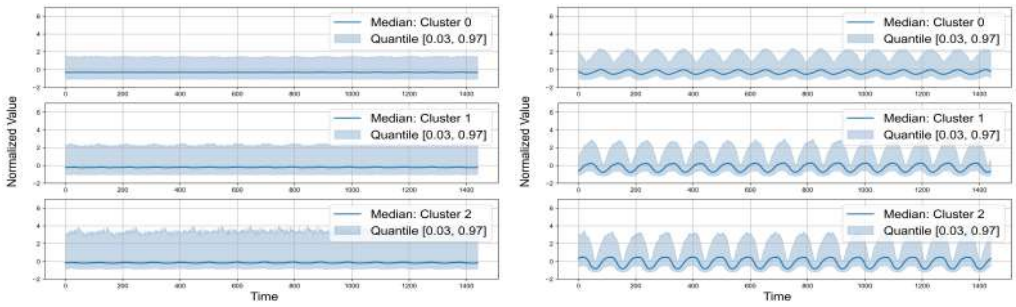
Figure 5: Snapshot of clusters generated by TDC (time series of downlink usage are plotted), where x axis represents time and y axis represents the normalized score. The first row refers to cluster 2, the second row refers to cluster 1 and the last row refers to cluster 0.

original time series with Euclidean distance, which results in waveform-oriented grouping. Even through we can observe the difference between resulting clusters, their patterns are quite similar, and the grouping rule is mainly derived by the horizontal shift of the waveform. Comparing the behavior of clusters created by TDC and K-medoids, TDC performs clustering with "high dimensional" features which are only available after mining knowledge from the original input, and the steps of feature extraction provide the clustering engine a different way to understand the same input, enabling a more creative separation of clusters. To evaluate the performance of our algorithm, we compare the performance of TDC and baselines based on the metrics mentioned above, and the result is reported in Table 2.

Algorithm	Silhouette	DB Index	CH Index
K-means Clustering	0.137	1.949	2652.16
K-medoids Clustering	0.125	2.089	2223.17
Hierarchical Clustering	**0.616**	1.769	84.21
TDC	0.435	**0.785**	**18417.76**

Table 2: Comparison of clustering algorithms.

From Table 2, TDC outperforms other algorithms on two metrics. The result of K-means and K-medoids are similar, and their performance is quite poor. Hierarchical clustering has the highest silhouette score among these algorithms, but its CH index is much lower than the others. Overall, TDC is the best one among them. According to the results, the use of the proposed feature extractor significantly boosts the clustering performance, and the clustering engine can generate clusters in a better separated way in the latent space. Based on the previous results, a well-designed feature extractor can be more important than



(a) (b)

Figure 6: Comparison of TDC and K-medoids clustering (time series of downlink usage are plotted). From top to bottom: cluster 0, cluster 1 and cluster 2. The blue line is the median value calculated within the cluster, and the shaded region indicates the upper bound 0.97 quantile and the lower bound 0.03 quantile of each cluster. (a) TDC; (b) K-medoids Clustering.

Running Time	Precomputation	Clustering	Total
`K-means Clustering`	—	0.30 mins	**0.30 mins**
`K-medoids Clustering`	7.79 hours	2.13 mins	7.82 hours
`Hierarchical Clustering`	7.79 hours	0.12 mins	7.79 hours
`TDC`	**4.50 mins**	**0.01 mins**	4.51 mins

Table 3: Running time of clustering algorithms.

clustering algorithm in the sense of improving clustering performance or customizing the behavior of clusters. Besides the metrics related to the separation of clusters, running time is another key issue of applying clustering algorithm on large dataset, as some algorithms are essentially not capable to deal effectively with large time-series. To study the behavior of these algorithms, the running time of baselines and TDC are shown in Table 3. There are three terms in the table: precomputation time is the time used to perform precomputation, such as the time of calculating pairwise distance matrix (K-medoids and hierarchical clustering) or generating representation of time series (TDC). According to this table, the computational cost of creating distance matrix is very expensive when the dataset is large. The calculation takes approximate 7.8 hours even with Euclidean distance, which means DTW is impractical to use considering its complexity is quadratic. Among these algorithms, K-means clustering has the shortest running time which is 0.3 minutes. For K-means, there is no need to calculate pairwise distance between all objects, and this reduces the computational cost significantly. Because of this reason, K-means is pretty common and widely used in many works [9, 40, 41]. Comparing K-medoids and hierarchical clustering, hierarchical clustering is computationally cheaper due to the lower time complexity $O(m^2)$ [10]. K-medoids is much more expensive, as the complexity is $O(m^2 z)$ for BUILD phase and $O(i(m-z)^2 z)$ for SWAP phase, where

m is the number of objects, z is the number of clusters and i is the number of needed iterations [31, 11]. The result proves that the similarity measurement is more important than the clustering itself if we want to reduce overall computational cost, and that is one reason why TDC generates new representation before applying K-means. For TDC, the creation of representation takes 4.50 minutes, and its K-means phase is 30x faster (0.01 minutes) than the basic one (0.30 minutes) with the new representation. In Algorithm 1, the time complexity of creating representation is $O(mqg^2)$, where q is the number of variables and g is the length of each sequence. The calculation of quantile is the most computational consuming operation because it requires to sort the sequence first, the time complexity of sorting is sequence-length dependent which usually ranges from $O(g\log(g))$ to $O(g^2)$ depending on sorting algorithms. The total running time of TDC is slower than K-means but 100x faster than hierarchical clustering and K-medoids. Even though K-means is a fast algorithm, it cannot generate well-separated clusters and compare the variability properly. Considering both the quality of clusters and total running time, our algorithm is the best approach among them.

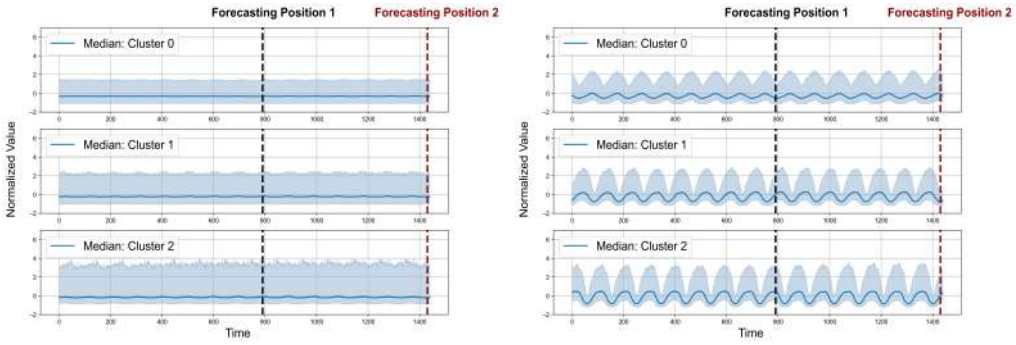


Figure 7: Two predefined forecasting positions to evaluate the forecasting difficulty of clusters. (a) TDC; (b) K-medoids Clustering.

For TDC, besides the aforementioned advantages, the algorithm can also identify the forecasting difficulty of time series which provides supplemental information for improving and evaluating prediction models. If we want to make accurate time series predictions, how to make a forecasting model better learning the temporal dynamics of time series is a critical problem. Usually the time series with strong temporal dynamics are more difficult to predict compared to the others, and the prediction performance can be improved a lot if the architecture of model is optimized for predicting these sequences. However, due to the fact that we do not know which time series are more difficult to predict, it is hard to improve the forecasting model and study the forecasting behavior in detail. From this point of view, TDC is a useful tool which provides us the knowledge of the forecasting difficulty of time series. By considering clusters are created from the variability of time series, each cluster can be seen

as a collection of time series which has different amount of dynamics. For time series forecasting, it is very difficult to predict a time series when it is highly stochastic and strong oscillating. In this case, the evaluation of forecasting difficulty is consistent with the underlying grouping rule of TDC. To verify this, we perform experiments aimed to evaluate the prediction difficulty of different clusters. In Figure 7, we define two forecasting positions at which a purposely trained neural network is tested for prediction accuracy. At forecasting position 1, the goal is to use the preceding 796 historical measurements to predict the values of the next 4 steps. At forecasting position 2, the goal is to use the preceding 1436 historical measurements to predict the values of the last 4 steps. The reason why we define different forecasting positions is to prevent forecasting error from being biased by local waveform. For example, If we observe Figure 7(b), we can find that the oscillating strength of clusters changes over time; for the peak parts of the signal, the median value is not close to the 0.97 quantile, and the distance between them decreases a lot at the valley parts of the signal. In this case, the forecasting difficulty of each cluster is not consistent over time by considering the temporal dynamics changes periodically, and it is essential to define multiple forecasting positions to evaluate the performance of predictor.

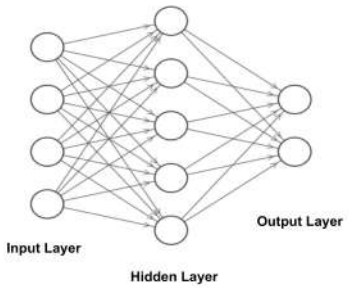


Figure 8: MLP

Following this idea, we train Multilayer Perceptron (MLP) [42] individually on each cluster generated by TDC and K-medoids. As shown in Figure 8, MLP is a feedforward neural network which is composed of input layer, hidden layer and output layer, and it is widely used in research because it can approximate solutions for complex problems. In our experiments, MLP consists of three fully connected layers and ReLU is used as the activation function; MLP is trained to predict the downlink usage and the average number of connected users in next hour by using their historical measurements. For each cluster, we shuffle its time series and use 80% of them as train set, 10% of them as validation set and the other 10% of them as test set. We first train MLP on the train set, and the training is stopped if the Mean Absolute Error (MAE) cannot be decreased on validation set. The hyperparameters of MLP are tuned based on its performance evaluated on validation set, then with these hyperparameters, MLP is retrained on the merged original train set and validation set. The performance of MLP is evaluated on the corresponding test set and the results are reported in Table

4 and Table 5.

	Cluster 0	**Cluster 1**	**Cluster 2**
`Forecasting Position 1`	**0.199 ± 0.034**	0.362 ± 0.051	0.376 ± 0.040
`Forecasting Position 2`	0.289 ± 0.044	**0.169 ± 0.018**	0.390 ± 0.037

Table 4: MAE of MLP evaluated on the clusters created by K-medoids.

	Cluster 0	**Cluster 1**	**Cluster 2**
`Forecasting Position 1`	**0.214 ± 0.022**	0.329 ± 0.047	0.444 ± 0.055
`Forecasting Position 2`	**0.220 ± 0.030**	0.312 ± 0.056	0.438 ± 0.083

Table 5: MAE of MLP evaluated on the clusters created by TDC.

The two tables show MAE calculated on the test set of each cluster, and the performance of predictor differs a lot on clusters generated by different algorithms. For the clusters generated by K-medoids (Table 4), at forecasting position 1, the predictor gets the best performance on cluster 0, and the easiest dataset changes from cluster 0 to cluster 1 if we switch to the second forecasting position. When we go back to Figure 7(b), it is easy to notice that cluster 0 has the lowest variation at forecasting position 1, and for forecasting position 2, the lowest variation can be found on cluster 1. This verifies our assumption which is mentioned in previous discussion: the difficulty of forecasting could change with the forecasting position of the waveform. In this case, the clusters generated by K-medoids cannot be used to describe the difficulty of time series because the performance of predictor evaluated on each cluster is not consistent over time. Comparing to K-medoids, the behavior of clusters created by TDC is consistent; irrespective of which forecasting position is used to evaluate, cluster 0 is always the easiest one which relates to weak temporal dynamics, and we can observe an increasing trend of MAE with the increment of temporal dynamics level.

4.4. Analysis of Cell Behavior in Urban Environment

In the previous section we discussed the characteristic of TDC and evaluated its performance using different metrics. In this section, we focus on the real usage case of this algorithm of analyzing the cell behavior in intricate urban environment. Due to complex usage patterns of mobile users, the distribution of mobile traffic is not homogeneous in a telecommunication network, whereby some cells carry more traffic and serve more users than the others. For better allocating network resources, it is crucial to understand the importance of each cell, so that suitable resources can be allocated. In this paper, we analyze the traffic pattern of cells located in Turin. We first study the overall behavior of cells with two weeks measurements, and then we explore the time-specific behavior of cells by using records collected in different time slots. Figure 9 visualizes the overall importance of cells in Turin; the cells belonging to cluster

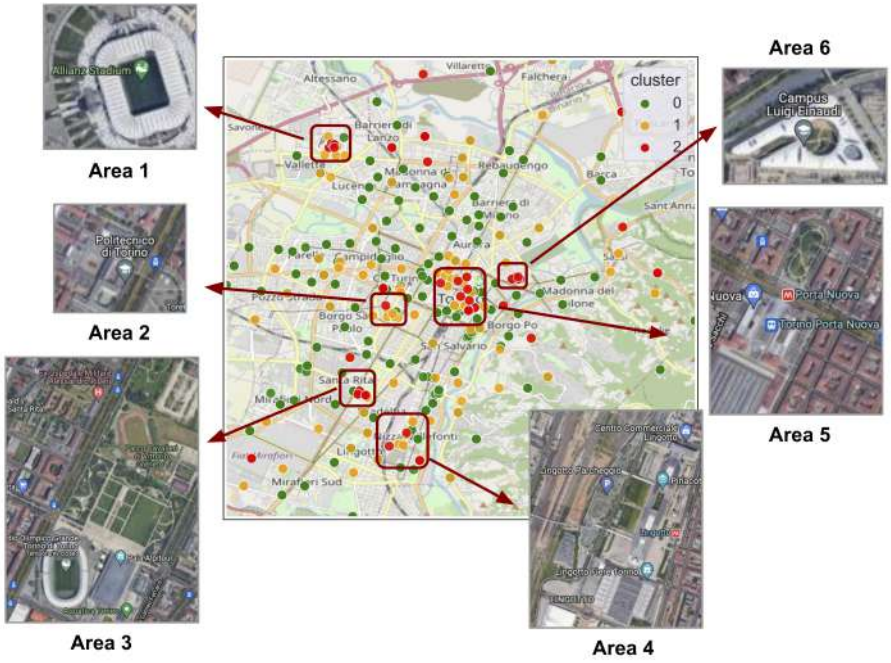
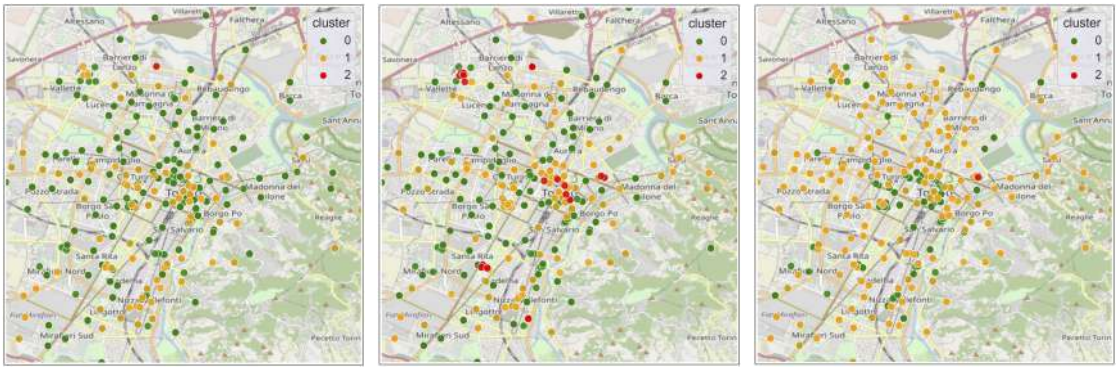


Figure 9: Visualization of geographical distribution of clusters in urban areas, each dot represents one deploying location of cell. Among three clusters, cluster 2 has the strongest temporal dynamics while cluster 0 has the weakest one.

2 are the most important ones, they exhibit strong temporal dynamics which means they suffer heavily changing mobile demand. On the map, most cells are grouped into cluster 0 (green dots) which is the least important cluster from a global view, these cells are mostly distributed among residential area and suburb. For moderately important cells (yellow dots) and the most important cells (red dots), the size of these clusters is much smaller than that of the green one. Observing the geolocation of cells, it can be seen that some areas are more important because the neighboring cells have higher temporal dynamics. To understand the hidden patterns, we explore the land usage of 6 different areas in detail. Employing Google Maps we find that the importance of cell mainly relates to human activities and functionalities of corresponding area, especially for the regions whose functionalities are related to sport, work/study, commute and leisure. For example, Area 1, Area 3 and Area 4 play key roles in sport: Area 1 is the stadium of the Juventus football club, Area 3 contains the olympics stadium and one large park of Turin where is a popular place for jogging. In Area 4 there is another large stadium which was used for figure skating and short track speed skating events at the 2006 Winter Olympics. Among these areas, Area 4 and Area 5 are comprehensive areas, and both of them have complicated functionalities. Area 5 is the city center of Turin, which has two train stations,



(a) (b) (c)

Figure 10: Visualization of geographical distribution of clusters in different time slots. (a) Morning (from 6:00 to 9:30); (b) Afternoon (from 13:30 to 17:30); (c) Night (from 22:00 to 24:00).

plenty of restaurants, offices and historical buildings. Another train station can be found in Area 4, which also has a large mall. Area 2 and Area 6 only have the functionality of education, and include the main campus of Polytechnic of Turin and University of Turin. Based on this verification, it is obvious that the importance of cell is highly related to human activities.

Human activities are time-dependent, and this characteristic makes the usage pattern evolve over time, which results in the change of cell importance. To investigate this aspect, we perform clustering for three time slots: morning (commuting time), afternoon (working time) and night (entertainment time). From Figure 10, it can be seen that the cell importance differs a lot in different time slots. In the morning, most of important cells are near to train stations, subway stations and the main roads of Turin (e.g., the roads along East-West and South-West directions). Afternoon is usually seen as working time, and the pattern shown in Figure 10(b) is very similar to Figure 9; city center and stadiums are highlighted in this time slot. During night, most of cells are grouped into cluster 1 and cluster 0. Comparing with the afternoon, cells distribute in a more homogeneous way. Basically the moderately important cells (yellow dots) include transport facilities and residential areas.

5. Conclusion

In this paper we introduced TDC, a novel time series clustering algorithm for analyzing the cell behavior in mobile network. This algorithm measures the variation between adjacent elements through creating first-order differences sequences, and a new representation is generated by summarizing the distribution of differences sequences to represent the temporal dynamics level of time series. The created representation is employed by K-means to split time series into different groups based on their variability. We evaluated the performance of our algorithm on real-world datasets of mobile networks, the results showed that

our algorithm is the best one among baselines. Through TDC, we can generate clusters based on temporal dynamics level in a computationally efficient way, which is very beneficial for performing clustering analysis on large dataset of mobile network. Our algorithm can also identify the forecasting difficulty of time series, providing supplemental information for improving and evaluating prediction models. To study the cell behavior in urban environment, we applied TDC to discover the usage pattern of mobile network cells in Turin, Italy. Through detailed analysis of several relevant areas, we found that the importance of cell relates to the land usage of that region. The areas whose functionality is closely tied to sport, work/study, commute and leisure play a key role in urban environment, the cells which are deployed in these regions are more important in mobile network, and the behavior of these cells evolves over time following the time-dependent nature of human activities. Through our research, we proved that our algorithm is effective for addressing real-world problems, and provided a deeper understanding of traffic usage pattern in intricate urban environment, which is very valuable for Internet service providers.

Acknowledgement

This work is supported by the PhD research program of TIM S.p.A (Italy).

References

[1] Cisco, Cisco annual Internet report (2018–2023) white paper (2020).

[2] F. Xu, Y. Li, H. Wang, P. Zhang, D. Jin, Understanding mobile traffic patterns of large scale cellular towers in urban environment, IEEE/ACM Transactions on Networking 25 (2) (2017) 1147–1161. doi:10.1109/TNET.2016.2623950.

[3] K. Sultan, H. Ali, Z. Zhang, Call detail records driven anomaly detection and traffic prediction in mobile cellular networks, IEEE Access 6 (2018) 41728–41737. doi:10.1109/ACCESS.2018.2859756.

[4] D. Kim, B. Shin, D. Hong, J. Lim, Self-configuration of neighbor cell list utilizing E-UTRAN NodeB scanning in LTE systems, in: 2010 7th IEEE Consumer Communications and Networking Conference, IEEE, 2010, pp. 1–5.

[5] L. Ruiz, M. Pegalajar, R. Arcucci, M. Molina-Solana, A time-series clustering methodology for knowledge extraction in energy consumption data, Expert Systems with Applications 160 (2020) 113731. doi:https://doi.org/10.1016/j.eswa.2020.113731.
URL https://www.sciencedirect.com/science/article/pii/S0957417420305558

[6] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 43–49. doi:10.1109/TASSP.1978.1163055.

[7] X. Cai, T. Xu, J. Yi, J. Huang, S. Rajasekaran, DTWNet: a Dynamic Time Warping Network, Advances in Neural Information Processing Systems 32 (2019).

[8] M. Cuturi, M. Blondel, Soft-DTW: a differentiable loss function for time-series, in: International Conference on Machine Learning, PMLR, 2017, pp. 894–903.

[9] S. Aghabozorgi, A. S. Shirkhorshidi, T. Y. Wah, Time-series clustering–a decade review, Information Systems 53 (2015) 16–38.

[10] E. Keogh, J. Lin, Clustering of time-series subsequences is meaningless: implications for previous and future research, Knowledge and Information Systems 8 (2) (2005) 154–177.

[11] R. T. Ng, J. Han, CLARANS: A method for clustering objects for spatial data mining, IEEE Transactions on Knowledge and Data Engineering 14 (5) (2002) 1003–1016.

[12] J. Lin, M. Vlachos, E. Keogh, D. Gunopulos, Iterative incremental clustering of time series, in: International Conference on Extending Database Technology, Springer, 2004, pp. 106–122.

[13] C. Guo, H. Jia, N. Zhang, Time series clustering based on ICA for stock data analysis, in: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE, 2008, pp. 1–4.

[14] V. Hautamaki, P. Nykanen, P. Franti, Time-series clustering by approximate prototypes, in: 2008 19th International Conference on Pattern Recognition, IEEE, 2008, pp. 1–4.

[15] M. Elangasinghe, N. Singhal, K. Dirks, J. Salmond, S. Samarasinghe, Complex time series analysis of PM10 and PM2. 5 for a coastal site using artificial neural network modelling and k-means clustering, Atmospheric Environment 94 (2014) 106–116.

[16] E. Mosca, G. Bertoli, E. Piscitelli, L. Vilardo, R. A. Reinbold, I. Zucchi, L. Milanesi, Identification of functionally related genes using data mining and data integration: a breast cancer case study, Bmc Bioinformatics 10 (12) (2009) 1–11.

[17] Z. Jiang, W. Bai, W. Bin, Social network users clustering based on multivariate time series of emotional behavior, The Journal of China Universities of Posts and Telecommunications 21 (2) (2014) 21–31.

[18] F. Zheng, Q. Liu, Anomalous telecom customer behavior detection and clustering analysis based on ISP's operating data, IEEE Access 8 (2020) 42734–42748.

[19] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki, Z. Smoreda, Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage, in: Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, 2017, pp. 180–186.

[20] E. A. Walelgne, A. S. Asrese, J. Manner, V. Bajpai, J. Ott, Clustering and predicting the data usage patterns of geographically diverse mobile users, Computer Networks 187 (2021) 107737. doi:https://doi.org/10.1016/j.comnet.2020.107737.
URL https://www.sciencedirect.com/science/article/pii/S1389128620313244

[21] Q. Zhu, L. Sun, Big data driven anomaly detection for cellular networks, IEEE Access 8 (2020) 31398–31408.

[22] H. D. Trinh, E. Zeydan, L. Giupponi, P. Dini, Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach, IEEE Access 7 (2019) 152187–152201.

[23] A. J. McNeil, Extreme value theory for risk managers, Departement Mathematik ETH Zentrum 12 (5) (1999) 121–237.

[24] C. Scarrott, A. MacDonald, A review of extreme value threshold estimation and uncertainty quantification, REVSTAT–Statistical Journal 10 (1) (2012) 33–60.

[25] M. Kratz, S. I. Resnick, The qq-estimator and heavy tails, Communications in Statistics. Stochastic Models 12 (4) (1996) 699–724. arXiv:https://doi.org/10.1080/15326349608807407, doi:10.1080/15326349608807407.
URL https://doi.org/10.1080/15326349608807407

[26] H. Drees, S. Resnick, L. de Haan, How to make a Hill plot, The Annals of Statistics 28 (1) (2000) 254 – 274. doi:10.1214/aos/1016120372.
URL https://doi.org/10.1214/aos/1016120372

[27] C. Neves, M. Fraga Alves, Reiss and Thomas' automatic selection of the number of extremes, Computational Statistics and Data Analysis 47 (4) (2004) 689–704. doi:https://doi.org/10.1016/j.csda.2003.11.011.
URL https://www.sciencedirect.com/science/article/pii/S0167947303002858

[28] M. Loretan, P. C. Phillips, Testing the covariance stationarity of heavy-tailed time series: An overview of the theory with applications to several

financial datasets, Journal of Empirical Finance 1 (2) (1994) 211–248. doi:https://doi.org/10.1016/0927-5398(94)90004-3.
URL https://www.sciencedirect.com/science/article/pii/0927539894900043

[29] A. K. F. Ho, A. T. K. Wan, Testing for covariance stationarity of stock returns in the presence of structural breaks: an intervention analysis, Applied Economics Letters 9 (7) (2002) 441–447. arXiv:https://doi.org/10.1080/13504850110090210, doi:10.1080/13504850110090210.
URL https://doi.org/10.1080/13504850110090210

[30] M. Omran, E. McKenzie, Testing for covariance stationarity in the uk all-equity returns, Journal of the Royal Statistical Society: Series D (The Statistician) 48 (3) (1999) 361–369.

[31] E. Schubert, P. J. Rousseeuw, Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms, in: International Conference on Similarity Search and Applications, Springer, 2019, pp. 171–187.

[32] A. K. Jain, R. C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., USA, 1988.

[33] P. J. R. Leonard Kaufman, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, Inc., 1990.

[34] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65. doi:https://doi.org/10.1016/0377-0427(87)90125-7.
URL https://www.sciencedirect.com/science/article/pii/0377042787901257

[35] D. L. Davies, D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1 (2) (1979) 224–227. doi:10.1109/TPAMI.1979.4766909.

[36] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics 3 (1) (1974) 1–27. doi:10.1080/03610927408827101.

[37] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, E. Woods, Tslearn, a machine learning toolkit for time series data, Journal of Machine Learning Research 21 (118) (2020) 1–6.
URL http://jmlr.org/papers/v21/20-091.html

[38] A. Novikov, PyClustering: Data mining library, Journal of Open Source Software 4 (36) (2019) 1230. doi:10.21105/joss.01230.
URL https://doi.org/10.21105/joss.01230

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[40] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.

[41] P. S. Bradley, U. Fayyad, C. Reina, Scaling clustering algorithms to large databases, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, KDD'98, AAAI Press, 1998, p. 9–15.

[42] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.