Clustering of Large Time series Datasets

Saeed Aghabozorgi\textsuperscript{a,1}, Teh Ying Wah\textsuperscript{b}

\textsuperscript{a}Faculty of Computer Science & Information Technology University of Malaya
\textsuperscript{b}Faculty of Computer Science & Information Technology University of Malaya

Abstract. Time series clustering is a very effective approach in discovering valuable information in various systems such as finance, embedded bio-sensor and genome. However, focusing on the efficiency and scalability of these algorithms to deal with time series data has come at the expense of losing the usability and effectiveness of clustering. In this paper a new multi-step approach is proposed to improve the accuracy of clustering of time series data. In the first step, time series data are clustered approximately. Then, in the second step, the built clusters are split into sub-clusters. Finally, sub-clusters are merged in the third step. In contrast to existing approaches, this method can generate accurate clusters based on similarity in shape in very large time series datasets. The accuracy of the proposed method is evaluated using various published datasets in different domains.

Keywords: Data mining, Clustering, Time series, Large datasets.

1 Introduction

A time series is a sequence of continuous values which is naturally high dimensional and large in data size. Clustering such complex data is particularly advantageous for exploratory data analysis, for summary generation, and as a pre-processing step for either another time series mining task or as part of a complex system. Researchers have shown that generally, clustering by using well-known conventional algorithms generate clusters with acceptable structural quality and consistency, and are partially efficient in terms of execution time and accuracy for static data [42]. However, classic machine learning and data mining algorithms do not work well for time series due to their unique structure [59]. The high dimensionality, very high feature correlation, and (typically) the large amount of noise that characterize time series data present difficult challenges for clustering [47,59]. Accordingly, massive research efforts have been made to present an efficient approach for time series clustering. However, focusing on the efficiency and scalability of these methods to deal with time series data has come at the expense of losing the usability and effectiveness of clustering [66]. That is, they suffer from either overlooking of data or inaccurate similarity computation. Overlooking of data is caused by dimensionality reduction (sometimes reduction to very low resolutions of data). Inaccurate similarity computation is due to use inappropriate metrics. For example, Euclidean distance (ED) is adopted as a distance metric on most of the existing works

\textsuperscript{1} Corresponding author: Saeed Aghabozorgi, Department of Information Science, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
E-mail: saeed@siswa.um.edu.my
because it is very efficient whereas it is not accurate enough because it is only proper to calculate the similarity in time (i.e. similarity on each time step) [11,68]. In contrast, it is shown that clusters generated based on similarity in shape (i.e. the time of occurrence of patterns is not important), are very accurate and meaningful clusters. For instance, Dynamic Time Warping (DTW) [24] can compute the similarity in shape between time series. However, this metric is not usually used because it is very expensive in terms of time complexity.

In this paper, the problem of the low quality in existing works is taken into account, and a new Multi-step Time series Clustering model (MTC) is proposed which can make the clusters based on similarity in shape. This model facilitates the accurate clustering of time series datasets and is designed especially for very large time series datasets. It overcomes the limitations of conventional clustering algorithms in finding the clusters of similar time series in shape.

In the first step of the model, data are pre-processed, transformed into a low dimensional space, and grouped approximately. Then, the pre-clustered time series are split in the second step by using an accurate clustering method, and are represented by some prototypes. Finally, in the third step, the prototypes are merged to construct the ultimate clusters.

To evaluate the accuracy of the proposed model, MTC is tested extensively by using published time series datasets from diverse domains. This model is more accurate than any existing works and is also scalable (on large datasets) due to the use of multi-resolution of time series in different levels of clustering. It is shown that using MTC, clustering of time series based on similarity in shape does not need to calculate the exact distances between all time series in a dataset; instead, by using prototypes of similar time series, accurate clusters can be obtained. The contribution of this study is (1) capability of the model for accurate clustering of large time series based on similarity in shape, (2) providing a clear understanding of the domains by its ability to generate hierarchical clusters of large time series data.

The rest of this paper is organized as follows. In Section 3, the related works are described. The proposed model is explained in Section 4. The algorithm is applied on diverse time series datasets and the experimental results are reported in Sections 5. In Section 6, conclusions and future perspectives are drawn.

2 Related work

There are two main categories in time series clustering [48], “Subsequence clustering” and “Whole time series clustering”. Subsequence clustering is based on sliding window extractions of an individual time series and aims to find similarity and differences among different time windows of an individual time series. Time series clustering has become an important topic, motivated by the challenge of developing methods to recognize dynamic change [77] and similarity search of sequences. “Whole time series clustering” is the clustering performed on many individual time series to group similar series into clusters. The focus of this research is on whole time series clustering with a short or modest length, not on long time series because comparing time series that are too long is usually not very meaningful [56]. For long time series clustering, some global measures (e.g., seasonality, periodicity, skewness, chaos, etc.) which are obtained by statistic operations, are more important [84]. In general, whole time series clustering can be broadly classified into five groups according to conventional forms of clustering algorithms: Partitioning, Hierarchical, Grid-based, Model-based and Density-based clustering algorithms.

In hierarchical clustering of time series, nested hierarchy of similar groups is generated based on a pairwise distance matrix of time series [82]. Hierarchical clustering has a great
visualization power in time series clustering [50]. This characteristic of hierarchical clustering leads to be used for time series clustering to a great extent [38,62]. Additionally, in contrast to most algorithms, hierarchy clustering does not require the number of clusters as an initial parameter which is a well-known and outstanding feature of this algorithm. It is also a strength point in the time series clustering, because usually it is hard to define the number of clusters in real world problems. However, essentially hierarchical clustering cannot deal effectively with large time series datasets [84] due to its quadratic computational complexity. Accordingly, it leads to be restricted to the small datasets because of its poor scalability.

A partitioning clustering method, makes $k$ groups from $n$ unlabelled objects such that each group contains at least one object. One of the most used algorithms of partitioning clustering is k-Means [60] where each cluster has a prototype which is the mean value of its members. However, when it comes to the time series clustering, constructing an effective prototype is a challenging issue [61]. Another member of partitioning family is k-Medoids algorithm [43], where the prototype of each cluster is one of the nearest objects to the centre of the cluster. In both k-Means and k-Medoids clustering algorithms, number of clusters, $k$, has to be pre-assigned, which is not available or feasible to determine for many applications, so it is impractical in obtaining natural clustering results and is known as one of their drawbacks in static objects [84] and also time series data [9]. Even it is worse in time series because the datasets are very large and diagnostic checks for determining the number of clusters is not easy. However, k-Means and k-Medoid are very fast compared to hierarchical clustering [18] and it has made them very suitable for time series clustering. Therefore, they have been used in many works either in their “crispy” manner [11,14,35,37,59,69] or “fuzzy” manner (Fuzzy c-Means and Fuzzy c-Medoids) [3,5,80].

Model-based clustering assumes a model for each cluster, and finds the best fit of data to that model. The model that is recovered from the generated data defines clusters [78]. There are a few articles which use model based clustering of time series data which are composed of polynomial models [11], Gaussian mixed models [17], ARIMA [25], Markov chain [64] and Hidden Markov models [16,39]. However, model based clustering has two drawbacks: first, it needs to set parameters and it is based on user's assumptions which may be false and result in inaccurate clusters. For example, it needs window adjustment and window overlap. Second, it has a slow processing time (especially neural networks) on large datasets [7].

In density based clustering, clusters are subspaces of dense objects which are separated by subspaces in which objects have a low density. The model proposed by Chandra and Chandra [20] is one of the rare cases, where the authors propose a density based clustering method in kernel feature space for clustering multivariate time series data of varying length. However, density-based clustering has not been used broadly for time series data clustering (in the literature), because of its rather high complexity.

The grid-based clustering quantizes the space into a finite number of the cells that form a grid, and then perform clustering on the grid’s cells. To the best of our knowledge, there is no work in the literature applying grid-based approaches for clustering of time series.

Although different researches have been carried on time series clustering, most of them have focused on high dimensional characteristic of time series data and tried to represent the time series in a low dimensional space to be suitable for the conventional clustering algorithms. The common characteristic in these works is the clustering of the transferred, extracted or raw time series using conventional clustering algorithms such as k-Means, k-Medoid or hierarchical clustering. However, most of them suffer from overlooking of data (caused by dimensionality reduction) and inaccurate similarity calculation (due to the high complexity of accurate measures). That is, clusterings are either accurate which are constructed expensively, or inaccurate but made inexpensively.
3 Concepts and Definitions

The key terms that are used in this work are presented in this section. The objects in the dataset related to the problem at hand are time series of similar length.

Definition 1: Time series clustering, given a dataset of \( n \) objects \( D = \{F_1, F_2, \ldots, F_n\} \), where \( F_i = \{f_{i1}, f_{i2}, \ldots, f_{iT}\} \) is a time series, the process of unsupervised partitioning of \( D \) into \( C = \{C_1, C_2, \ldots, C_k\} \), in such a way that homogenous time series are grouped together based on their similarity in shape, is called time series clustering. Then, \( C_i \) is called a cluster, where \( D = \bigcup_{i=1}^{k} C_i \) and \( C_i \cap C_j = \emptyset \) for \( i \neq j \).

3.1 Representation Method

Many researches have been carried out focusing on the representation or dimensionality reduction of time series [28,50,58,89]. Considering all these works, it is understood that dimension reduction is necessary to some extent, however it is undeniable that as more dimensionality reduction occurs, more data is lost and becomes inaccurate. In fact, it is a trade-off between the accuracy and speed which is a controversial and non-trivial task in representation methods. However, among all these representation methods which have their strong points and weaknesses, in this paper, the focus is on Symbolic Aggregate ApproXimation (SAX) representation because of its strength in the representation of time series.

3.1.1 Brief Review of SAX

Two different studies [51,89], separately approximated the time series using segmentation approach. They use the mean value of the equal-length segmentations of time series as the approximate value of that part. This technique is called Piecewise Aggregate Approximation (PAA), and is quite fast [46], can be used for arbitrary length queries. If \( F_i = \{f_{i1}, f_{i2}, \ldots, f_{iT}\} \) is considered as a time series, then, PAA discretized time series to \( \hat{F} \) where \( \hat{F} = \hat{f}_{11}, \ldots, \hat{f}_{w} \). Each segment of \( \hat{F} \), i.e., \( \hat{f}_{ij} \), is a real value which is the mean of all data points in the \( j \)th segment of \( F \). Symbolic Aggregate ApproXimation (SAX) maps the PAA coefficients to symbols. SAX was developed by Keogh et al. in 2003 and has been used by more than 50 groups in different data mining researches [58]. Considering \( \hat{F} \) as discretized time series by PAA transformation, then, \( \hat{F} \) where \( \hat{F} = \hat{f}_{11}, \ldots, \hat{f}_{w} \), is defined by mapping the PAA coefficients to ‘a’ SAX symbols. ‘a’ is the alphabet size (e.g., for the alphabet= \{a, b, c, d, e, f\}, ‘a’ = 6), which are defined by “breakpoints”. Based on Keogh definition, a list of numbers \( B = \beta_1 \ldots \beta_{a-1} \) is defined as “breakpoints” to determine the area of each symbol in SAX transformation.

3.2 Similarity measure

Time series clustering relies on distance measures to a high extent. There are many distance measures proposed by researchers in the literature such as ED [31], DTW [74,75], ERP [21], TQuEST [10], LCSS [12,81], EDR [22] and CDM [49]. However, it can be drawn from literature that ED and DTW are the most common methods in the time series clustering because of the efficiency of ED and the effectiveness of DTW in similarity measurement. ED is a one-to-one matching measurement which is used in most of the works (about 80%) in the literature [19,31,44,51]. ED is simple, fast and is used as a benchmark in many works, because it is parameter free. However, ED is not the best choice as a distance function
because it is dependent on the domain of the problem in hand and the characteristics of
the dataset’s time series. In fact, it is very weak and sensitive to small shifts across the time axis
[15,52,67,88] which make it proper for finding time series which are similar in time. For example it is not accurate enough for calculating similarity of sequences such as: <abaa>,<aaba>.

In contrast to ED which proposes a one-to-one matching, DTW is suggested as a one-to-
many metric. DTW is a generalization of ED which solves the local shift problem in the
time series to be compared. Local shift problem is a time scale issue which is a characteristic of
most time series. Handling local shifts allows similar shapes to be matched even if they are
out of phase in the time axis, i.e., similar in shape. Using this definition, clusters of time
series with similar patterns of change are constructed regardless of time points, for example,
to cluster share price related to different companies which have a common pattern in their
stock independent of its occurrence in time series [11]. It makes DTW superior to ED
[1,22,24,81,89] which is only able to find time series which are similar in time.

DTW uses “warping” the time axis in order to achieve the best alignment between the
data points within the series (Figure 1). Given two time series, \( F_i = \{ f_1, f_2, ..., f_x \} \), \( F_j = \{ f_1, f_2, ..., f_y \} \), a \( x \times y \) matrix is defined where the \((s, k)\) element of the matrix is the Euclidean
distance, \( \text{dis}(f_s, f_k) \), between two time points \( f_s \) and \( f_k \). The warping path \( w_u \) forms a set of warping paths \( \{ w_1, w_2, ..., w_u \} \), that has the minimum distance between the two series of
\( F_i \) and \( F_j \) is of interest.

\[
\text{dis}_{\text{DTW}}(F_i, F_j) = \min\left( \sum_{u=1}^{U} \frac{w_u}{U} \right)
\]

[Figure 1]

Generally, dynamic programming is used in order to find the warping path effectively. However, it causes scalability problem which is a big challenge for DTW because it requires quadratic computation [76]. As a result, many researchers, [28,52,54,83,90], try to speed it up usually by proposing an efficient lower bound approximations of the DTW distance to reduce its complexity. Nevertheless, most of these works are under the classification problem (the search area is pruned using a lower bound distance of DTW) and is not suitable for the clustering problem where the distance between all objects should be calculated. However, in this paper DTW can be adopted whiteout violating the efficiency of clustering using a novel approach explained further.

4 A New Multi-step Approach

MTC includes three steps: pre-clustering, splitting and merging. Figure 2 shows the
overall view of the process in MTC briefly. In the first step (i.e. pre-clustering step), data are
clustered to reduce the search space. In this step, time series data are used in a low-resolution
mode. In the second step, time series data are used in their high-resolution mode. A new
approach is designed to split the pre-clusters into sub-clusters. Finally, in the third step, sub-
clusters are merged using an arbitrary scheme.

[Figure 2]

4.1. Step 1: Pre-clustering

Pre-clustering is performed to group the low-resolution time series (which can fit in
memory) rather than original (raw) time series dataset. It reduces the search area for the
second step of MTC. At first, time series are standardized using z-score (z-Normalization)
[36] which make time series invariant to scale and offset. Then, SAX is adopted as a representation method to reduce the dimensionality of normalized time series. The major reasons for using dimensionality reduced time series in the pre-clustering step is the problem of disk I/O constraint especially in the large datasets [45]. The generic solution for this problem is to create an approximation of the data [51,57], which will fit in the main memory, yet retains the essential features of interest. As a result, the whole data can be loaded into main memory and the problem at hand is solved approximately [18,45]. However, the size of a dataset can be very large such that even with dimensionality reduction, it cannot fit in the memory. The solution for this case is utilizing an incremental clustering where clusters are updated (or expanded) incrementally [2]. Moreover, when we talk about large datasets, in addition to a large number of instances, it could include a long duration in the time axis. However, because MTC is a multi-step algorithm, the first step can be done by different distance measures depend on the nature of time series. For example, the method proposed in [29,85] can be adopted which takes into account dynamic changes in the data characteristics.

In order to make the pre-clusters, an appropriate distance measure compatible with SAX is desirable. In the literature, Euclidean [55] or MINDIST [58] measure are used in order to calculate the similarity between two time series represented by SAX. Lin et al [58] introduced MINDIST as a compatible distance metric for SAX. However, MINDIST has been introduced to address the indexing problem in the time series domain and is not enough accurate for calculation of distance among time series in the clustering problem, as a result, in this paper, a new approach, Approximate Distance (APXDIST) is introduced.

As mentioned, symbols in SAX are defined based on the location of PAA coefficients in some equi-probable regions made by some break points [57]. MINDIST considers the distance between the neighbour symbols as zero, and ignores the maxima and minima points of time series. Instead, in APXDIST, the distance between regions is calculated as the distance between indicators of the regions. The indicator of a region is defined in such a way that the closeness of PAA coefficients (in the region) to the indicator is the highest in that region. For defining the indicator, the arithmetic mean of each area (minimum and maximum) is defined as an indicator of the area as the best estimator of the regions as:

$$Ind_i = \frac{\beta_{i-1} + \beta_i}{2}$$

where $\beta_0$ is the global minimum and $\beta_m$ is the global maximum. Figure 3 illustrates a visual intuition of the APXDIST metric. The black line shows the MINDIST between two symbols, while dashed line indicates the APXDIST distance between the same symbols.

[Figure 3]

Based on this definition, the APXDIST between each pairs of symbolized time series is defined as following:

$$dis_{APXDIST}(\hat{F}_x, \hat{F}_y) = \sqrt{\frac{\pi}{w}} \sqrt{\sum_{i=1}^{w} (dis(Ind_{i}, Ind_{j}))^2}$$

where $ind_i$ is the indicator of $i$th region and the $dis()$ function is calculated by

$$dis(Ind_{i}, Ind_{j}) = \begin{cases} 0, & i = j \\ \frac{\beta_{j} + \beta_{j-1} - \beta_{i} - \beta_{i-1}}{2}, & \text{otherwise} \end{cases}$$

which is pre-computed and read-off by a lookup table. In this part, APXDIST is evaluated against the MINDIST and ED. To make a comparison, some datasets are adopted which are explained in Section 5.2. At first, time series of each dataset is transferred into discrete space (SAX). Then, three distance matrices are made by each distance measure (i.e., APXDIST, MINDIST and ED). At that point, the difference between the obtained distance matrices and
the distance matrix calculated by ED using raw time series \( ED_{RAW} \) is computed as tightness of each metric to \( ED_{RAW} \). The tightness of the metrics, for example APXDIST, to \( ED_{RAW} \) is calculated by the following equation.

\[
Tightness(\hat{F}_i, \hat{F}_j) = 1 - \frac{|ED_{RAW}(F_i,F_j) - APXDIST(\hat{F}_i, \hat{F}_j)|}{ED_{RAW}(F_i,F_j)}
\]

where \( \hat{F}_i \) is the SAX representation of time series \( F_i \). The average tightness of all UCR datasets over the cross product of the alphabet [5-8] and compression-ratio [4-8] is shown in the Figure 4. The bigger tightness indicates more accurate approximation. As the results show, APXDIST is closer to \( ED_{RAW} \) rather than the other two approaches for most of the datasets. It implies that APXDIST is more accurate than other approaches.

Figure 4

To cluster the approximated data, k-Modes [40] is used in order to divide \( N \) time series into \( k \) partitions. k-Modes is based on k- Means family algorithm, and is used for clustering categorical data. Considering that time series represented by SAX is categorical data in each segment, and k-Modes work fine with categorical data [40,41], it is a good choice for pre-clustering. Moreover, similar to other partitioning algorithms, it has a high speed (especially in large datasets) [41] and provides a good quality by choosing the centroids on a low dimension approximation of data (e.g. SAX) which increases the quality in the partitioning clustering [27].

4.2. Step 2: Sub-clustering

The clustering results provided by dimension reduction approaches (especially in sensitive datasets) are not accurate enough because of the overlooking of data in the dimensionality reduction process [55]. As a result, regardless of adopted techniques, the clustering should be applied to high-resolution time series. The main objective of this step of MTC is to split the pre-clusters and making prototypes. Moreover, the number of time series in the dataset (cardinality) is reduced by defining a prototype for each group of very similar time series. It decreases the complexity of MTC to a high extent.

For this step, ED is used as the similarity measure to calculate the similarity in time between high-resolution time series data in each pre-cluster. Figure 5 depicts the intuition behind using ED in the second step.

Figure 5

Now, the pre-clusters constructed in the first step are split. Given pre-clusters which include time series which are similar approximately, we can split them based on similarity in time which means very close time series which are very probable to belong to a class of time series. The purity of sub-clusters is calculated further to show the error rate of summarization in prototyping step.

Definition 2: Sub-cluster, a sub-cluster \( SC_{ij} \) is a set of individual time series that are similar in time, created by the splitting of pre-clusters \( PC_i \), and is represented as a single prototype.

For illustrative purposes, a simple diagram in 2-dimensional space is used in order to describe the intuition behind the process of splitting approximated pre-clusters (see Figure 6).

Figure 6

Considering the pre-cluster \( PC_i \), a similarity measure between members of \( PC_i \) is calculated and stored in an m-by-m similarity matrix, \( M_{m \times m} \), where \( M_{ij} \) is the distance (similarity) measure between time series \( F_i \) and time series \( F_j \). Then, a developed algorithm, SubClusterSearch, is carried out on the similarity matrix to find whether cluster members are
scattered solidly (with the same affinity). Then, for pre-cluster $PC_i$ which its members are not scattered solidly (low affinity time series), it divides the members into some sub-clusters. Sub-clusters are constructed based on the affinity of time series in the pre-cluster. The concept of cluster affinity is borrowed from Cluster Affinity Search Technique (CAST) [13] which used to find the close objects.

Definition 3: **Cluster affinity**, the affinity of a time series $F_x$ to a sub-cluster $SC$ is defined as follows (6):

$$a(F_x) = \frac{\sum_{y \in SC} M_{xy}}{|SC|}$$

where, $M_{xy}$ is the similarity between time series $F_x$ and $F_y$, and $|SC|$ is the number of time series that exist in the sub-cluster.

*SubClusterSearch* constructs sub-clusters (from pre-clusters) sequentially with a dynamic affinity threshold. The output of *SubClusterSearch* is discrete clusters which are separated without predetermining the number of sub-clusters. In this process, each sub-cluster is constructed with a time series and gradually is completed by new time series added to sub-cluster based on the average similarity (affinity) between unassigned time series (in pre-cluster) and the current sub-cluster members. Defining a specific threshold value, the cluster accepts high affinity time series. The affinity threshold, $\alpha$, is specified to determine what is considered significantly similar. This parameter controls the number and sizes of the produced sub-clusters. After forming a sub-cluster, *SubClusterSearch* deletes the low affinity objects from sub-clusters. These adding and removing to a sub-clusters is performed consecutively until no more changes occur in the sub-cluster.

As mentioned, an affinity threshold, $\alpha$, of pre-cluster $PC_i$ is defined dynamically which is very important because would limit our ability to impose our prejudices, expectations, and presumptions on the the size and shape of sub-clusters. That is, *SubClusterSearch* is proposed as an algorithm which works without predetermining the parameters. The affinity threshold, $\alpha$, is calculated dynamically based on the remaining time series in a pre-cluster, i.e., unassigned time series in the pre-cluster $PC_U$, before constructing each new sub-cluster $SC_{new}$ as:

$$\alpha = \frac{\sum_{xy \in PC_U} M_{xy}\mu (M_{xy} - \mu)}{|PC_U|} + \mu$$

where $\mu$ is the mean of the similarities of each time series to other time series in a pre-cluster ($\mu$ is initialized one time at the start of the algorithm). The Equation 7 calculates a threshold based on the within variance of affinities in the pre-cluster dynamically before creating each new sub-cluster. This value is used to distinguish the time series which has a low affinity by putting them into a new sub-cluster. Given a set of $n$ time series in a sub-cluster $CS_2$, the time series are represented by a time series $R_j = \{r_1, \ldots, r_x, \ldots, r_T\}$ as:

$$r_x = \frac{\sum_{i=1}^{n} f_{ix}}{n}$$

where $F_i = \{f_{i1}, \ldots, f_{ix}, \ldots, f_{iT}\}$ is a time series in $CS_j$. As a result, the output of this algorithm is some prototypes which represent different sub-clusters. Algorithm 1 shows the pseudo code related to *SubClusterSearch*.

One of the strengths of MTC is its flexibility in making different cluster shapes by choosing an arbitrary algorithm for merging. That is, given the sub-clusters and defining a similarity measure, many distance based algorithms such as partitioning, hierarchical, or density based clustering can be used for merging the sub-clusters in the third step. However,
instead of applying the clustering algorithm on the entire data, only the prototype of each sub-cluster is contributed in merging process which reduces the complexity of process to a high extend. Moreover, prototypes can increase the accuracy to some extent. That is, highly unintuitive results may be garnered using raw time series because some distance measures are very sensitive to some “distortions” in the raw time series. However, because each prototype is made by averaging of time series, it decreases the effect of distortions or outliers in time series.

Algorithm 1 SubClusterSearch (M)

**Input:** M : an n-by-n similarity matrix  
**Output:** C: a collection of sub-clusters  

Initializations:
1. $SC_{new} \leftarrow \emptyset$ // the constructing sub-cluster  
2. $PC_U \leftarrow \{F_1, \ldots, F_n\}$ // time series inside the pre-cluster  
3. $\mu \leftarrow \frac{\sum M_{xy}}{n}$ // average similarity between time series in the pre-cluster  
4. while $(PC_U \cup SC_{new} \neq \emptyset)$ do  
5. $\alpha = \mu + \frac{\sum M_{xy} - \mu}{\left|PC_U\right|}$ // each $F_x, F_y$ in $PC_U$  
6. end if  
7. if (a($F_U$) >= $\alpha$) // $F_U$ with maximal affinity in $PC_U$  
8. $SC_{new} \leftarrow SC_{new} \cup \{F_U\}$ // insert $F_U$ into $SC_{new}$  
9. $PC_U \leftarrow PC_U \setminus \{F_U\}$ // remove $F_U$ from $PC_U$  
10. for all $x$ in $PC_U \cup SC_{new}$ and $F_U$ in $SC_{new}$ do  
11. $a(F_x) = \sum M_{xy} / \left|SC_{new}\right|$ // update the affinity  
12. end  
13. else // no high affinity elements outside $C_{new}$  
14. if (a($F_v$) < $\alpha$) // let $F_v$ be a vertex with minimal affinity  
15. $SC_{open} \leftarrow SC_{new} \setminus \{F_v\}$  
16. $PC_U \leftarrow PC_U \cup \{F_v\}$ // insert $F_v$ into $PC_U$  
17. for all $F_x$ in $PC_U \cup SC_{new}$ and $F_v$ in $SC_{new}$ do  
18. $a(F_x) = \sum M_{xy} / \left|SC_{new}\right|$ // update the affinity  
19. end  
20. else  
21. $R \leftarrow \text{Average}(SC_{new})$ // summarize the cluster  
22. $C \leftarrow C \cup R$  
23. $SC_{new} \leftarrow \emptyset$ // start a new cluster  
24. $a(.) \leftarrow 0$ // reset the affinity  
25. end  
26. end  
27. end  
28. return the collection of prototypes, C.

The following experiment verifies that SubClusterSearch can reduce the size of data but not reduce its effectiveness greatly.

Sub-clusters (made from a pre-cluster) are desirable such that most of their members are members of a natural cluster (class). That is, because the pre-clusters are generated approximately, they are often mixed with time series of different classes. Therefore, they should be recognized and separated by searching in the pre-clusters. As a result, pre-clusters are broken into sub-clusters. To calculate the purity of sub-clusters, the purity index is used.
Purity index is a way of measuring the quality of a clustering solution [92]. Purity is a simple and transparent evaluation measure. Considering \( G = \{ G_1, G_2, ..., G_M \} \) as the ground truth clusters, and \( SC = \{ SC_1, SC_2, ..., SC_M \} \) as the sub-clusters made by SubClusterSearch algorithm under evaluations, in order to compute the purity of sub-cluster\( SC \) with respect to \( G \), each sub-cluster is assigned to the class which is most frequent in the sub-cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned data which is similar in shape are very close to ground truth and more meaningful.

In the first two steps of MTC, similar time series were grouped using APXDIST and ED based on their similarity in time. However, two time series which are not similar in time may be similar in shape. As mentioned, the similarity in shape is desirable in the clustering of time series, because finding clusters of time series which are similar in shape are very close to ground truth and more meaningful. This fact can be seen in the clustering of time series using...
ED which is based on the similarity in time and DTW which is based on the similarity in shape. To show the superiority of DTW in the clustering, the quality of clustering of different datasets using ED and DTW is depicted in Figure 8.

[Figure 8]

Considering the results shown in Figure 8, expectedly, DTW is a better choice in comparison with ED in terms of quality because of handling shifts in calculating the similarity. However, the methods such as DTW are mostly costly [76] for similarity evaluation of time series. Moreover, it was explained that pruning techniques are not suitable for clustering purpose because dissimilarity matrix must be fully calculated. For example, in clustering algorithms such as the well-known Unweighted Pair-Group Method with Arithmetic Mean (UPGMA) [79], all distances must be calculated and no pruning can be done. In such cases, clustering process would benefit from a fast and accurate similarity measure [34]. However, in the MTC, we don’t have to calculate the similarity in shape between all time series, because the very close time series (similar in time) are similar in shape as well. That is, the dissimilarity matrix does not need to be fully calculated by an expensive similarity measure such as DTW. As a result, only a small part of the matrix is calculated by DTW using the prototypes of sub-clusters in the third step of MTC (instead of whole data) which are small in size. It can calculate all possible mappings between two time series' data points, finding the smallest possible warping distance. As a result, inter similarity between the time series inside the sub-clusters is computed based on the similarity in time and inter similarity is calculated based on similarity in shape.

The result of merging process is some clusters which are constructed from prototypes of sub-clusters. Thus, a mapping activity is carried out to assign the original time series to their corresponding prototypes. All time series of each sub-cluster are assigned to the cluster which the corresponding prototype is assigned.

5 Experimental Evaluation

5.1. Datasets

The proposed model is experimented with 19 different datasets from the UCR Time series Data Mining Archive [53] where the number of clusters, dimensionality, and number of instances have been explained. This set is chosen because it is of various numbers of clusters, different cluster shapes and density, contains noise points, and used in many articles in the literature as a benchmark.

5.2. Evaluation method

In general, evaluating of extracted clusters (patterns) is not easy in the absence of the data labels [91]. However, all these datasets have class labels and can be used for evaluation of MTC using external indices such as Dunn [30], DB [26], Silhouette [72], etc. Complete reviews and comparisons of some popular techniques exist in the literature [6,71]. However, these indices are dependent on the adopted clustering algorithm and structure of clustering [8], and there is not a compromise and universally accepted technique to evaluate clustering approaches. Therefore, the most common-used external indices in the time series clustering domain are used for evaluation of accuracy of MTC, i.e., Rand Index, Adjusted Rand Index, Entropy, Purity, Jaccard, F-measure, FM, CSM, and MNI (the interested reader is referred to [6,87] for definitions and compression). To avoid biased evaluation, the conclusions are
drawn based on the average value of the indices. Considering $G = \{G_1, G_2, ..., G_M\}$ as ground truth clusters, and $C = \{C_1, C_2, ..., C_M\}$ as the clusters made by MTC under evaluation, some of these indices are explained in the following.

**Folkes and Mallows index (FM):** To calculate the FM, at first, the following quantities are considered as ground truth and errors in the clustering process:

- Let $|TP|$ (True Positive) be the number of pairs, each belongs to one class in $G$ (ground truth) and are clustered together in $C$.
- The $|TN|$ (True Negative) is the number of pairs, each neither belongs to the same class in $G$, nor clustered together in $C$.
- Then, the error clusterings are the $|FN|$ (False Negative) which is the number of pairs that are belong to one class in $G$, but are not clustered together in $C$, and $|FP|$ (False Positive) which is the number of pairs that are not belong to one class in $G$ (dissimilar time series), but are clustered together in $C$. Then the FM measure $[32]$ is defined as:

\[
FM(C, G) = \sqrt{\frac{|TP|}{|TP| + |FN|}} 
\]

**Jaccard Score:** Jaccard [32] is one of the metrics that has been used in various studies as external index [23,68,91]. Considering the defined parameters for FM index, the jaccard index is defined as:

\[
Jaccard(C, G) = \sqrt{\frac{|TP|}{|TP| + |FP| + |FN|}} 
\]

**Rand index (RI):** A popular quality measure [23,68,91] for evaluation of time series clusters is the Rand index [65,87], which measures the agreement between two partitions, that is, how the clustering results are close to the ground truth. The agreement between $C$ and $G$ can be estimated using:

\[
RI(C, G) = \sqrt{\frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}} 
\]

**F-measure:** F-measure [70] is a well-established measure for assessing the quality of any given clustering solution with respect to ground truth. F-measure ($F \in [0, 1]$) is defined based on precision and recall. The precision of an object indicates how many items in the same cluster belong to the same class which is estimated as:

\[
Precision(C, G) = \frac{|TP|}{|TP| + |FP|} 
\]

The recall of an object reflects how many objects of the same class (in ground truth) are assigned to the same cluster:

\[
Recall(C, G) = \frac{|TP|}{|TP| + |FN|} 
\]

Then, F-measure is calculated as the harmonic mean between $Precision(C, G)$ and $Recall(C, G)$:

\[
F-measure(C, G) = \frac{2 \times Precision(C, G) \times Recall(C, G)}{Precision(C, G) + Recall(C, G)} 
\]

Moreover, to report the accuracy of MTC, the average quality of 100 runs is calculated to prevent the bias of random initialization [33,59,63,82]. Although the focus of this study is on improving the accuracy of MTC, the scalability of the proposed model is also calculated to prove its feasibility theoretically.
5.3. Comparing MTC with Partitioning Clustering

In this section, the accuracy of MTC is compared with different methods of partitioning clustering. At first, k-Medoid is chosen which has been shown that is effective in time series clustering domain [30–34]. To calculate the distance between time series data, ED and DTW are used as distance metric. In Figure 9, the quality of MTC approach in front of quality of k-Medoids on raw time series is shown. As mentioned, the datasets adopted in this experiment are labelled, therefore, the number of clusters \((k)\) in k-medoid is chosen based on the number of clusters determined in the repository [53].

[Figure 9]

First, as expected, comparing MTC with raw time series and Euclidian distance, i.e., \(_k\)-Medoids\(_{ED}\), MTC finds the right clusters for most of the datasets. Raw time series have quite good quality in this sense that they are not overlooked or approximated by representation methods (high fidelity). However, ED cannot handle the local shifts in time series data which decreases the accuracy of final clusters.

Second, using raw time series and DTW, i.e., \(_k\)-Medoids\(_{DTW}\), MTC is still superior to \(_k\)-Medoids\(_{DTW}\) on many datasets. It is because of the mechanism of using a prototype of similar time series instead of the original time series. In fact, DTW is very sensitive to outliers because in DTW, all points have to be matched. Using prototypes of the raw time series mitigates the effect of the noises and outliers in the time series datasets, and so, it decreases the sensitivity of DTW. Besides, using raw time series and DTW, i.e., \(_k\)-Medoids\(_{DTW}\), is not a fair comparison because it is not practically feasible in the real world due to its very high complexity.

Based on the literature review, most of the studies about clustering of time series use a representation method to reduce the dimension of time series before performing clustering. Therefore, in the next experiment, as a fair comparison, the raw time series are represented by SAX, and the MTC is compared in front of dimensionally reduced time series. However, because SAX accuracy depends on the compression-ratio value, the raw time series are represented using three different compression-ratios \((compression_{ratio} = [4,6,8])\). As a result, for each dataset, mean of three accuracies is calculated as average accuracy of k-Medoids. Likewise, as MTC also can accept different resolutions in the first step, the same resolutions have been used for MTC. The average of accuracies is reported as the quality of clustering of data using MTC (with k-Medoid scheme) as it is depicted in Figure 10.

[Figure 10]

As Figure 10 shows, the accuracy of MTC is much better than conventional k-Medoids for all datasets. The result reveals that a better quality of clustering is obtainable without reducing the dimension of time series, using the prototypes of very similar time series. It is the proof of the researcher’s claim that the MTC model can outperform the conventional algorithms which use dimensionality reduction approaches for clustering purpose.

5.4. Comparing MTC with Hierarchical Clustering

In this section, accuracy of MTC is compared with hierarchical clustering algorithms. In the case that a hierarchical clustering algorithm is used as a merging scheme for the third stage of MTC, a dendrogram is produced of possible clustering solutions at different steps of granularity. It is very easy to visually show the dendrogram of clusters because merging is on prototypes of each sub-cluster which are small (in comparison with original data). Nevertheless, we can go one step further, and cut the dendrogram (tree) and explicitly partition prototypes into specific clusters, as with k-Medoids. Cutting a single dendrogram at
different levels gives a different number of clusters. However, in this experiment, given the number of clusters \( k \) (as mentioned in section 5.1, the number of clusters is available for all datasets), we can partition the dendrogram. That is, using the hierarchy from the distance matrix to create clusters, a linkage height is specified that will cut the tree below the \( k-1 \) highest nodes, and create \( k \) clusters. For example, the conventional hierarchical clustering is experimented on the CBF dataset including three clusters and 30 time series represented by SAX with compression ratio = 4. Figure 11 shows the generated dendrogram using hierarchical clustering (average linkage) of CBF dataset which is cut to be divided into three clusters.

[Figure 11]

As Figure 11 shows, in this clustering solution, hierarchical clustering has failed to select the correct pair of clusters to merge them together, and as a result, the red-colored time series and green-colored time series are merged in the first and second clusters. Table 1 depicts the scalar quality of this solution.

[Table 1]

In contrast, a successful experiment made from MTC methodology is shown in Figure 12. This figure represents a pre-cluster which is broken into the sub-clusters of the similar time series in time. It is the result of applying \( \text{SubClusterSearch} \) on pre-clusters. As it is shown, all the sub-clusters in this example are quite pure. Consequently, for each sub-cluster a prototype is made which represents the whole time series in the sub-cluster.

[Figure 12]

The output of the second step of MTC is 10 time series (prototypes) which have the maximum purity. In the third step, using the hierarchical scheme, the prototypes are clustered. The result is clusters of prototypes which are made based on the similarity in shape.

[Figure 13]

As Figure 13 shows, hierarchical clustering is performed only on the provided prototypes which are less than the original data (e.g. 30% of the whole time series). As a result, calculating of similarity in shape (which is very costly) is carried out only on a few sets of time series. Then the prototypes are replaced through the mapping process. To show how effective this approach is, the proposed model is applied on different datasets. Figure 14 shows the result of MTC clustering by the hierarchical scheme with average linkage.

[Figure 14]

As the results in Figure 14 shows, the accuracy of MTC for most of the datasets outperforms the conventional hierarchical algorithm using the average linkage. This experiment was repeated for hierarchical clustering using single linkage also where MTC outperforms the conventional single linkage for most of the cases.

5.5. Evaluation of MTC on Large Datasets

In order to further confirm the effectiveness of MTC, some experiments are carried out on large synthetic datasets. For this purpose, CC dataset [4] and CBF dataset [73] with different cardinalities are generated up to 12000 records. In order to evaluate the results of the proposed model on large datasets, MTC (with different schemes of merging) was experimented in front of conventional k-Medoids and hierarchical approach (with single and average linkages). The average quality of MTC with different schemes is depicted in the Figure 15 and Figure 16.

[Figure 15]
As the result shows, the quality of MTC is superior to other algorithms. However, the fluctuation of MTC in both datasets is a bit more than other algorithms (except hierarchical algorithm using single linkage). It is the effect of, 1) showing the result by averaging of accuracy per different parameter values of SAX, and 2) using prototypes of data instead of the original time series, i.e., the error of the second step in purifying the clusters. However, for the most of the cardinalities of the dataset, it is seen that the minimum quality of MTC is still more than other algorithms. Moreover, the trend shows better quality as the cardinality increased. Therefore, it can be concluded that there is no need to use very low dimensional time series for clustering of the large time series datasets; instead, the clustering can be applied on smaller sets of high dimensional time series through prototyping. That is, approximated sub-clusters, has a less destructive effect on the accuracy of the final clusters compared with the use of the approximated time series.

5.6. Time Complexity

The overall computational complexity of MTC depends on the amount of time it requires to construct the first step clusters, to split the pre-clusters in the second step, and to perform the merging of the prototypes in the last step. Suppose that the number of time series is \(N\), and the length of time series is \(d\), which changes into \(r\) after dimensionality reduction process, where \(r < d\). The complexity of dimensionality reduction is \(O(Nd)\) and the time complexity of k-Modes is \(O(Ik_{PC}Nr)\), where \(k_{PC}\) is the number of clusters, and \(I\) is the number of iterations takes to converge. Therefore, the complexity of the first step is linear in the number of time series \(N\), because \(I\) is not very big as the most changes happened in the first iterations running on dimensionality reduced time series. Complexity of second step, in the best case is same as CAST algorithm[13] which is a bit more than \(O(N.\log(N))\) and in the worst case is \(O(N^2)\). Considering the complexity of ED as \(O(d)\), the complexity of second step is \(O\left(\frac{N^2}{k_{PC}}d\right)\). Since there are \(k_{PC}\) such partitions, the overall complexity becomes \(O\left(\frac{N^2}{k_{PC}}d\right)\). Then, if the final clustering scheme is for example a hierarchical clustering (e.g., average linkage) which its complexity is \(O(n^2\log(n))\) where \(n\) is the number of input objects, and \(O(d^2)\) for DTW between each pair of the time series, then, the total computation of merging (third step) is \(O(n^2\log(n).d^2)\).

**Overall complexity:** The number of time series in the second step is reduced by the reduction-factor of \(R_f\) which is defined based on reduction-rate (equation 3) as:

\[
R_f = 1 - R_{rate}
\]

Substituting \(N, R_f\) (i.e., number of prototypes) in the equation of the third step, the overall time required for three steps is

\[
O(M\TC) = O \left( N^2. R_f^2 \log(NR_f). d^2 + \frac{N^2}{k_{PC}}. d + l k_{PC}NR + Nd \right)
\]

Given the complexity of MTC (using the hierarchical scheme) as \(O(M\TC)\), and conventional hierarchical algorithm which is in order of \(O(N^2\log(N).d^2)\) as \(O(Hier)\), for comparing MTC with hierarchical algorithm, the ratio of complexity of MTC on hierarchical algorithm is calculated as:

\[
\text{Ratio} = \frac{O(M\TC)}{O(Hier)} = \frac{R_f^2\log(NR_f)}{\log(N)} + \frac{1}{k_{PC}\log(N)d} + \frac{l k_{PC}r}{N\log(N).d^2} + \frac{1}{N\log(N).d}
\]

because \(R_f < 1\) then \(\log(N. R_f) < \log(N)\), cancelling out the \(\log(N)\), the ratio of
complexity of MTC on hierarchy algorithm becomes:

\begin{equation}
\text{Ratio} = R_f^2 + \frac{1}{k_{pc} \log(N)} + \frac{1}{N \log(N) \cdot d^2} + \frac{1}{N \log(N) \cdot d}
\end{equation}

Considering for example the dimension reduction of \( r=d/4 \) (i.e., compression-ratio=4, e.g., SAX4), and \( k_{pc} = \sqrt{\frac{N}{2}} \), then

\begin{equation}
\text{Ratio} = R_f^2 + \frac{1}{\sqrt{\frac{N}{2}} \log(N)} + \frac{1}{4 \sqrt{2} N \log(N) d} + \frac{1}{N \log(N) d}
\end{equation}

Because \( R_f < 1 \) (based on Equation 19) and \( I \) is small, then \( \text{Ratio} < 1 \), that is, \( \text{Ratio} \) is always less than 1. Therefore, it can be concluded that the complexity of MTC is better than hierarchical algorithm.

6 Conclusion

In this paper, we provided a detailed overview of various techniques used for clustering of time series data. Focusing on whole time series clustering, the results obtained by applying MTC on different datasets were evaluated visually and objectively. It was visually shown that clustering can be applied on large time series datasets to generate a hierarchy of meaningful and accurate clusters using MTC.

In order to evaluate the validity of the clusters formed, different evaluation methods were used to show the accuracy of MTC. The accuracy of the proposed method was evaluated using various datasets. Moreover, MTC was applied on large time series datasets. Finally, the time complexity of the proposed model was computed.

From the results presented in the previous sections, the following points are concluded.

- Considering the experimental results, DTW is a better choice in comparison with ED in terms of quality because of handling shifts in calculating the similarity. However, the methods such as DTW are costly for similarity evaluation of time series in conventional algorithms such as k-Medoids or hierarchical algorithms.
- Experimental results show that MTC outperforms other conventional clustering algorithms experimenting on various datasets. The results revealed that user does not require very low resolution time series for clustering of large datasets; instead, the clustering can be applied on smaller sets of high dimension time series by prototyping process. That is, the cost of using representatives is much less than dimension reduction in terms of accuracy.
- In large datasets, for the most of the cardinalities of the dataset, it is seen that the minimum quality of MTC is still more than other algorithms. The computation shows that the execution time of the model is acceptable.

In future work we will be extending the ideas presented here to consider long time series datasets to evaluate the approach. We will also be exploring how various representation methods will affect the final accuracy.

Acknowledgements

This paper is supported by RG097-12ICT University of Malaya Research Grant.
References


[41] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining and Knowledge Discovery. 304 (1998) 283–304.


Series Representation with Implications for Similarity Search and Clustering, in: In Proc. 9th Pacific-Asian Int. Conf. on Knowledge Discovery and Data Mining (PAKDD’05), Springer, 2005: pp. 771–777.


Table 1: The quality of wrong clustering of CBF using Hierarchical clustering by average linkage and SAX4

| RI, ARI, Purity, Entropy, f-measure, Jaccard, FM, NMI, CSM, Hier(Linkage.Avg) (SAX4) |
|---|---|---|---|---|---|---|---|---|
| 0.55 | 0.09 | 0.53 | 0.2 | 0.71 | 0.28 | 0.45 | 0.23 | 0.46 | 0.39 |
Figure captions:

Figure 1: Calculating of similarity in shape between two time series using DTW

Figure 2: The overall view of steps of MTC

Figure 3: Computing the distance between the time series using APXDIST.

Figure 4: The tightness of APXDIST to the Euclidean distance

Figure 5: Similarity calculation in different steps of MTC

Figure 6: A 2-dimensional view of pre-clusters and sub-clusters

Figure 7: SubClusterSearch approach reduction-rate in front of gained purity

Figure 8: Quality of clustering in front of ED and DTW

Figure 9: Quality of MTC approach in front of standard k-Medoids on raw time series

Figure 10: Quality of MTC (k-Medoids) in front of ground truth for UCR dataset

Figure 11: Dendogram of hierarchical clustering of CBF using average linkage

Figure 12: Selecting a time series as the prototype of a sub-cluster in the second step

Figure 13: Third step of MTC: Merging prototypes (hierarchical merging of prototypes)

Figure 14: Average quality of MTC approach in front of running hierarchical clustering (average linkage) on dimensionally reduced time series.

Figure 15: Accuracy of MTC in front of other algorithms for CBF dataset

Figure 16: Accuracy of MTC in front of other algorithms for CC dataset
[Figure 1] Simil. in Shape (DTW)

[Figure 2]
Figure 6

Figure 7

Figure 8
[Figure 9]

[Figure 10]

[Figure 11]
[Figure 15]

[Figure 16]