Mining of Temporal Coherent Subspace Clusters in Multivariate Time Series Databases

Hardy Kremer, Stephan Günnemann, Arne Held, and Thomas Seidl

RWTH Aachen University, Germany
{kremer,guennemann,held,seidl}@cs.rwth-aachen.de

Abstract. Mining temporal multivariate data by clustering techniques is recently gaining importance. However, the temporal data obtained in many of today's applications is often complex in the sense that interesting patterns are neither bound to the whole dimensional nor temporal extent of the data domain. Under these conditions, patterns mined by existing multivariate time series clustering and temporal subspace clustering techniques cannot correctly reflect the true patterns in the data.

In this paper, we propose a novel clustering method that mines temporal coherent subspace clusters. In our model, these clusters are reflected by sets of objects and relevant intervals. Relevant intervals indicate those points in time in which the clustered time series show a high similarity. In our model, each dimension has an *individual* set of relevant intervals, which *together* ensure temporal coherence. In the experimental evaluation we demonstrate the effectiveness of our method in comparison to related approaches.

1 Introduction

Mining patterns from multivariate temporal data is important in many applications, as for example analysis of human action patterns [12], gene expression data [8], or chemical reactions [17]. Temporal data in general reflect the possibly changing state of an observed system over time and are obtained by sensor readings or by complex simulations. Examples include financial ratios, engine readings in the automotive industry, patient monitoring, gene expression data, sensors for forest fire detection, and scientific simulation data, as e.g. climate models. The observed objects in these examples are individual stocks, engines, patients, genes, spatial locations or grid cells in the simulations. The obtained data are usually represented by multivariate time series, where each attribute represents a distinct aspect of observed objects; e.g., in the health care example, each patient has a heart rate, a body temperature, and a blood pressure. The attributes are often correlated; e.g. for the forest fire, the attributes temperature and degree of smoke are both signs of fire. Unknown patterns in such databases can be mined by clustering approaches, where time series are grouped together by their similarity. Accordingly, clusters of time series correspond to groups of objects having a similar evolution over time, and clusters represent these evolutions.

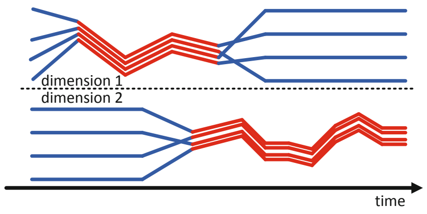


Fig. 1. Multivariate temporal pattern with two intervals

In many applications, however, existing approaches for clustering univariate or multivariate time series are ineffective due to a specific aspect of the analyzed data: patterns of interest that are neither bound to the whole dimensional nor temporal extent of the time series. Since our method is designed for effective mining under this scenario, we elaborate on this aspect in the next paragraph.

Temporal patterns of interest often only exist over a partial temporal extent of analyzed time series, i.e. they are constrained to an interval, and a single multivariate temporal pattern can have different intervals for each of its dimensions, as it is illustrated in Fig. 1. More concretely, time series belonging to one cluster only have similar values in these intervals, and values in the remaining intervals are noisy. Also, for some clusters there are dimensions in which there is no similarity between the time series. In the following, the intervals and dimensions belonging to a cluster are called relevant, while the remaining intervals and dimensions are called non-relevant. If non-relevant intervals are considered in distance measures that are used to decide which time series are grouped together, clusters can result that do not reflect the true patterns in the data.

Our novel, subspace clustering related approach handles this aspect by using an effective cluster model that distinguishes explicitly between relevant and non-relevant intervals for each cluster, and only relevant intervals are incorporated into the similarity function used for deciding which time series belong to a specific cluster. Our approach prevents incoherent time series clusters, i.e. clusters that have points in time that do not belong to any of the cluster's individual intervals. This ensures that there are no single (incoherent) cluster which would better be represented by several single (coherent) clusters.

Summarized, we propose a novel, subspace clustering related approach for effective clustering of multivariate time series databases that

– uses a cluster definition that distinguishes explicitly between relevant and non-relevant intervals for each individual dimension; the individual intervals as a whole form a temporal coherent cluster.
– is efficient due a approximate computation of our model that delivers high quality results.

This paper is structured as follows: in Section 2 we discuss related work. In Section 3 we introduce our approach for effective clustering of multivariate time series, for which an efficient algorithm is presented in Section 4. In Section 5 we evaluate our approach and Section 6 contains concluding remarks.

2 Related Work

Clustering of temporal data can roughly be divided into clustering of incoming data stream, called stream clustering [9], and clustering of static databases, the topic of this paper. Our method is related to two static clustering research areas, namely time series clustering and subspace clustering, and we discuss these areas in the following. In the experiments, we compare to methods from both areas.

Time Series Clustering. There is much research on clustering *univariate* time series data, and we suggest the comprehensive surveys on this topic [4,11]. Clustering of *multivariate* time series is recently gaining importance: Early work in [14] uses clustering to identify outliers in multivariate time series databases. Multivariate time series clustering approaches based on statistical features were introduced in [2,19,20]. There are no concepts in these approaches to discover patterns hidden in parts of the dimensional or temporal extents of time series.

Most clustering approaches are based on an underlying *similarity measure* between time series. There is work on noise-robust similarity measures based on partial comparison, called Longest Common Subsequences (LCSS) [18]. We combined k-Medoid with LCSS as a competing solution.

There is another type of time series clustering methods, designed for applications in which single, long time series are mined for frequently appearing subsequences. These methods perform *subsequence clustering*, with subsequences being generated by a sliding window. Since we are interested in patterns that occur in several time series at similar points in time and not in patterns that occur in a single time series at arbitrary positions, those approaches cannot be applied in our application scenario.

Subspace Clustering (2D) and TriClustering (3D). Subspace clustering [1,10,15,21] was introduced for high-dimensional (non-temporal) vector data, where clusters are hidden in individual dimensional subsets of the data. Since subspace clustering is achieved by simultaneous clustering of the objects and dimensions of dataset, it is also known as 2D clustering. When subspace clustering is applied to 3D data (objects, dimensions, time points), the time series for the individual dimensions are concatenated to obtain a 2D space (objects, concatenated dimensions). While subspace clustering is good for excluding irrelevant points in time, there are problems when it is applied to temporal data: First, by the transformation described above, the correlation between the dimensions is lost. Second, subspace clustering in general cannot exploit the natural correlation between subsequent points in time, i.e. temporal coherence is lost.

Accordingly, for 3D data, *Triclustering* approaches were introduced [7,8,16,22], which simultaneously cluster objects, dimensions, and points in time. Special Triclustering approaches are for clustering two related datasets together [6], which is a fundamentally different concept than the one in this paper. Generally, Triclustering approaches can only find block-shaped clusters: A cluster is defined by a set of objects, dimensions, and points in time [22] or intervals [7,16]. The points in time or intervals hold for all objects and dimensions of a cluster.

In contrast, our approach can mine clusters where each dimension has different, independent relevant intervals.

3 A Model for Effective Subspace Clustering of Multivariate Time Series Data

In the following we introduce our model for subspace clustering of multivariate time series data. In Section 3.1, we introduce our definition for subspace clusters of complex multivariate time series data, and in Section 3.2 we formalize an optimal clustering that is redundancy-free and contains clusters of maximal interestingness.

3.1 Time Series Subspace Cluster Definition

As input for our model we assume a database DB of multivariate time series where $Dim = \{1, \ldots, Dim_{max}\}$ denotes the set of dimensions and $T = \{1, \ldots, T_{max}\}$ the set of points in time for each time series. We use $o[d,t] \in \mathbb{R}$ to refer to the attribute value of time series $o \in DB$ in dimension $d \in Dim$ at time $t \in T$. As an abbreviation, $o[d, t_1 \ldots t_2]$ denotes the univariate subsequence obtained from object o by just considering dimension d between points in time t_1 and t_2. Our aim is to detect temporal coherent patterns, i.e. similar behaving objects, in this kind of multivariate data.

Since we cannot expect to find temporal patterns over the whole extent of the time series or within all dimensions, we have to restrict our considerations to subsets of the overall domain. Naively, a cluster could be defined by a tuple (O, S, I) where the objects O show similar behavior in subspace $S \subseteq Dim$ and time points $I \subseteq T$. This is straightforward extension of subspace clustering to the temporal domain and is used in triclustering approaches like [7,16,22].

A model based on this extension is limited because each selected dimension $d \in S$ has to be relevant for each selected point in time $t \in I$. For example, if the objects O are similar in d_1 at time t_1 and in d_2 at t_2 but not similar in d_1 at time t_2, we cannot get a single cluster for the objects O. We either have to exclude dimension d_1 or time point t_2 from the cluster. Thus, important information is lost and clustering effectiveness degrades.

Our novel model avoids this problem by selecting per dimension an individual set of intervals in which the time series are similar in (cf. Fig. 1). Such an interval, which contains a specific temporal pattern, is denoted as interval pattern.

Definition 1. *An* ***interval pattern*** $IP = (O, d, Int)$ *is defined by:*

- *an object set* $O \subseteq DB$
- *one selected dimension* $d \in Dim$
- *an interval* $Int = \{start, \ldots, end\} \subseteq T$ *with* $length(Int) > 1$ *for* $length(Int) := end - start + 1$*, i.e. we only permit non-trivial intervals.*
- *a specific* ***cluster property*** *the corresponding subsequences* $o[d, start \ldots end]$ *with* $o \in O$ *have to fulfill. We use the compactness of clusters based on the Maximum Norm: i.e.,* $\forall o, p \in O\ \forall t \in Int:\ |o[d,t] - p[d,t]| \leq w$

To avoid isolated points in time, i.e. where time series are rather similar by chance, we require that $length(Int) > 1$. The cluster property defines how similarity between subsequences is measured and how similar they need to be in order to be included in the same interval pattern. This property can be chosen by specific application needs. Besides the cluster compactness, which is also used by other subspace clustering methods [13,15], other distance measures applicable for time series including DTW [3] can be used.

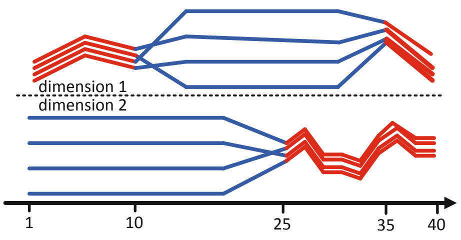


Fig. 2. Example for incoherent patterns

Based on the introduced interval patterns, clusters are generated: for each dimension, zero, a single, or even several interval patterns can exist in the cluster. This allows our method to systematically exclude non-relevant intervals from the cluster to better reflect the existing patterns in the analyzed data. However, not all combinations of interval patterns correspond to reasonable temporal clusters. The temporal coherence of the pattern is crucial. For example, let us consider a set of objects O forming three interval patterns in the time periods 1-10, 25-40, and 35-40, as illustrated in Fig. 2. Since for the remaining points in time no pattern is detected, there is no temporal coherence of the patterns. In this case, two individual clusters would reflect the data correctly.

To ensure temporal coherence, each point in time $t \in T$ that is located between the beginning $a \in T$ and ending $b \in T$ of a cluster, i.e. $a \leq t \leq b$, has to be contained in at least one interval pattern of an arbitrary dimension. Thus, by considering all dimensions simultaneously, the cluster has to form a single connected interval, and each point in time can be included in several dimensions.

Definition 2. ***TimeSC***. *A coherent time series subspace cluster (TimeSC) $C = (O, \{(d_i, Int_i)_{\{1...m\}}\})$, i.e. an object set together with intervals in specific dimensions, is defined by:*

- *for each interval $i \in \{1, \ldots, m\}$ it holds that (O, d_i, Int_i) is a valid interval pattern.*
- *the intervals per dimension are disjoint, i.e.*
  $\forall i, j \in \{1, \ldots, m\}, i \neq j : \; Int_i \cap Int_j = \emptyset \vee d_i \neq d_j$
- *the cluster is temporal coherent, i.e. combined, we have a single connected interval:*
  $\exists a, b \in T, a \leq b : \; \bigcup_{i=1}^{m} Int_i = \{a, \ldots, b\}$.

We require disjoint intervals per dimension because for overlapping intervals single points in time could be included multiple times in the cluster, which is obviously not beneficial for describing the cluster.

Overall, our novel cluster model avoids the drawbacks of previous methods and flexibly identifies the coherent temporal patterns in complex multivariate time series data.

3.2 Clustering Model: Redundancy Avoidance

Accounting for the properties of temporal data, an object can naturally occur in several clusters. Definition 2 allows for grouping various objects within different dimensions and time intervals. By generating the set $Clusters = \{C_1, \dots, C_k\}$ of all TimeSC C_i, we a priori permit overlapping clusters. Overlap, however, poses a novel challenge: the set *Clusters* potentially is very large and the contained clusters differ only marginally. In the worst case, two clusters differ only by few objects and hence one of theses clusters provides no novel information. Thus, some clusters may be highly redundant and are not beneficial for the user. As a solution, we aim to extract a subset $Result \subseteq Clusters$ that contains no redundant information.

In a set of clusters $M \subseteq Clusters$ redundancy can be observed, if at least one cluster $C \in M$ exists whose structural properties can be described by the other clusters. More precisely: if we are able to find a set of clusters $M' \subseteq M$ which together group almost the same objects as C and that are located in similar intervals, then C's grouping does not represent novel knowledge.

Definition 3. ***Structural Similarity.*** *A single time series subspace cluster TimeSC* $C = (O, \{(d_i, Int_i)_{\{1\dots m\}}\})$ *is structural similar to a set of TimeSC* M*, abbreviated* $C \approx M$*, iff*

– $\frac{|Obj(M) \cap O|}{|Obj(M) \cup O|} \geq \lambda_{obj}$ *(object coverage)*

– $\forall C_i \in M : \frac{|Int(C_i) \cap Int(C)|}{|Int(C_i) \cup Int(C)|} \geq \lambda_{int}$ *(interval similarity)*

with redundancy parameters $\lambda_{obj}, \lambda_{int} \in [0, 1]$*,* $Obj(M) = \bigcup_{(O_i, \cdot) \in M} O_i$ *and* $Int(C)$ *representing* C*'s intervals via 2-tuples of dimension and point in time:* $Int(C) = Int((O, K)) = \{(d, t) \mid \exists (d, Int) \in K : t \in Int\}$.

The higher the redundancy parameter values λ_{obj} and λ_{int} are set, the more time series (λ_{obj}) or intervals (λ_{int}) of C have to be covered by M so that M is considered structural similar to C. In the extreme case of $r_{obj} = r_{dim} = 1$, C's time series and intervals have to be completely covered by M; in this setting, only few clusters are categorized as redundant. By choosing smaller values, redundancy occurs more often.

The final clustering *Result* must not contain structural similar clusters to be redundancy-free. Since, however, several clusterings fulfill this property, we introduce a second structural property that allows us to choose the most-interesting redundancy-free clustering. On the one hand, a cluster is interesting if it contains many objects, i.e. we get a strong generalization. On the other hand, a cluster can represent a long temporal pattern but with less objects, corresponding to a high similarity within the cluster. Since simultaneously maximizing both criteria is contradictory, we introduce a combined objective function that realizes a trade-off between the number of objects and the pattern length:

Definition 4. *The **Interestingness** of a TimeSC $C = (O, \{(d_i, Int_i)_{\{1\ldots m\}}\})$ is defined by*

$$Interest(C) = |O| \cdot \sum_{i=1}^{m} length(Int_i)$$

By adding up the lengths of all intervals, overlap of intervals between different dimensions is rewarded. The optimal clustering result is defined by demanding the two introduced properties:

Definition 5. *The **Optimal Clustering** of the set of all valid clusters $Clusters$, i.e. $Result \subseteq Clusters$, fulfills*

(1.) redundancy-free property:
$\forall C \in Result : \neg \exists M \subseteq Result : C \approx M \wedge C \notin M$

(2.) maximal interestingness:
For all redundancy-free clusterings $Res' \subseteq Clusters$ it holds
$\sum_{C \in Result} Interest(C) \geq \sum_{C' \in Res'} Interest(C')$.

With this definition of an optimal clustering, the formalization of our novel cluster model for subspace clustering multivariate time series data is complete. In the next section, we will present an efficient algorithm for this model.

4 Efficient Computation

In this section we present an efficient algorithm for the proposed model. Due to space limitations we just present a short overview. Since calculating the optimal clustering according to Def. 5 is NP-hard, our algorithm determines an approximative solution. The general processing scheme is shown in Fig. 3 and basically consists of two cyclically processed phases to determine the clusters. Thus, instead of generating the whole set of clusters *Clusters* and selecting the subset *Result* afterwards, we iteratively generate promising clusters, which are added to the result.

Phase 1: In the first phase of each cycle a set of cluster candidates is generated based on the following procedure: A time series p acting as a prototype for these candidates is randomly selected, and this prototype is contained in each cluster candidate of this cycle. The cluster candidates, i.e. groups of time series O_i, are obtained by successively adding objects x_i to the previous group, i.e. $O_{i+1} = O_i \cup \{x_i\}$ with $O_0 = \{p\}$. Since the interestingness of a cluster depends

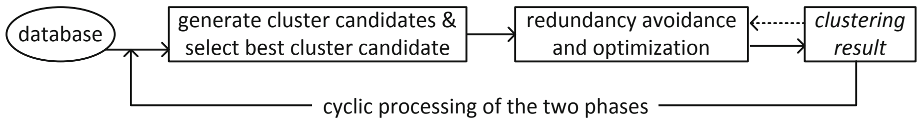

database
generate cluster candidates & select best cluster candidate
redundancy avoidance and optimization
clustering result
cyclic processing of the two phases

Fig. 3. Processing scheme of the algorithm

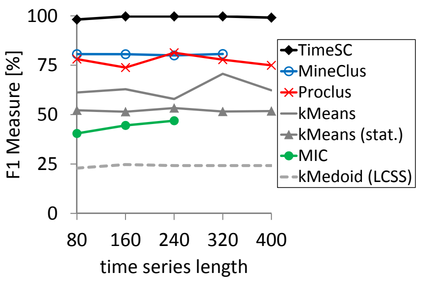
on its size, which is constant for O_i, and the length of the intervals, the choice of x_i is completely determined based on the latter. Accordingly, the best object x_0 is the one which would induce the longest interval interval patterns w.r.t. p (summed over all dimension).

An interval pattern for the prototype p and x_0 at the beginning can potentially include each point in time. Interval patterns for the subsequent objects x_i, however, have to be restricted to the relevant intervals of O_i. Overall, we generate a chain of groups O_i containing objects with a high similarity to p. Based on these candidates we select the set O_+ with the highest interestingness, i.e. according to Def. 4 we combine the size with the interval lengths.

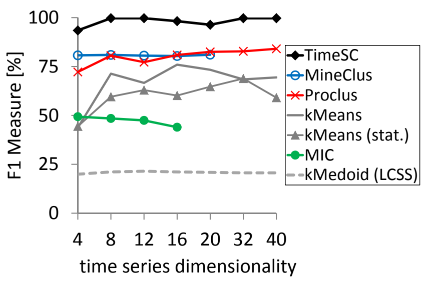
Phase 2: In the second phase, a cluster C for the object set O_+ should be added to the current result Res_j. In the first cycle of the algorithm the result is empty ($Res_0 = \emptyset$), whereas in later cycles it is not. Thus, adding new clusters could induce redundancy. Accordingly, for cluster C we determine those clusters $C' \in Res_j$ with similar relevant intervals (cf. Def. 3). For this set we check if a subset of clusters M covering similar objects as C exists. If not, we can directly add C to the result. In case such a set M exists, we test whether the (summed) interestingness of M is lower than the one of C. In this case, selecting C and removing M is beneficial. As a further optimization we determine the union of C's and M's objects, resulting in a larger cluster U with potentially smaller intervals. If U's interestingness exceeds the previous values, we select this cluster. This procedure is especially useful if clusters of previous iterations are not completely detected, i.e. some objects of the clusters were missed. This step improves the quality of these clusters by adding further objects. Overall, we generate a redundancy-free clustering solution and simultaneously maximize the interestingness as required in Def. 5.

By completing the second phase we initiate the next cycle. In our algorithm the number of cycles is not a priori fixed but it is adapted to the number of detected clusters. We perform $c \cdot |Res_j|$ cycles. The more clusters are detected, the more prototypes should be drawn and the more cycles are performed. Thus, our algorithm automatically adapts to the given data.

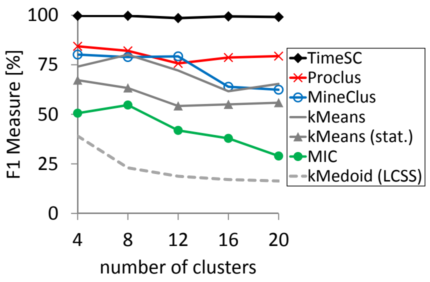
In the experimental evaluation, we will demonstrate the efficiency and effectiveness of this algorithm w.r.t. large scale data.



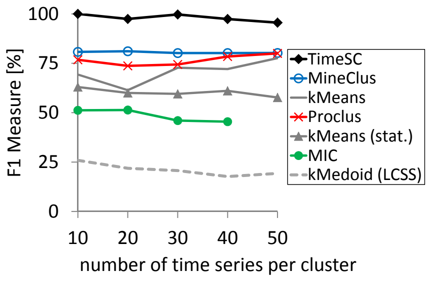
(a) data set length



(b) data set dimensionality



(c) number of clusters

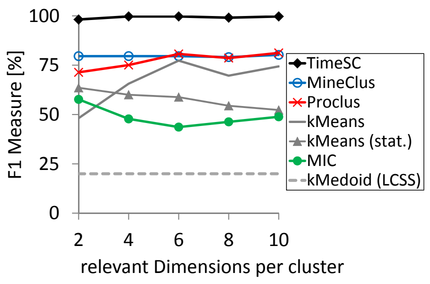


(d) time series per cluster

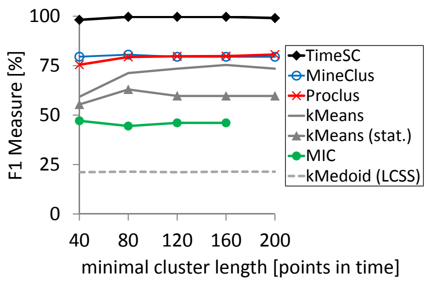
Fig. 4. Performance under different parameter settings

5 Experiments

We evaluate our TimeSC in comparison to six competing solutions, namely kMeans, a kMeans using statistical features for multivariate time series [19], Proclus [1], MineClus [21], and MIC [16]. We also included kMedoid, where we used the Longest Common Subsequences (LCSS) [18] as a distance measure to allow for partial comparison in the distance computation. We also compared to TriCluster [22], which was provided by the authors on their webpage, but it either delivered no result or the obtained accuracy was very low ($\leq 3\%$); therefore we no longer included it in the experiments. For TimeSC, we used $w = 30$ for the compactness parameter. For the redundancy model, we used $\lambda_{obj} = 0.5$ and $\lambda_{int} = 0.9$. Some of the competing algorithms are not suitable for large datasets; thus, in some experiments, we could not obtain all values. If not stated otherwise, we use the following settings for our synthetic data generator: The dataspace has an extend of [-100,+100], time series length is 200, clusters length is 100, the dataset dimensionality is 10, the number of relevant dimensions per cluster is 5, the number of clusters is 10, the average number of time series per cluster is 25, and there is 10% noise (outliers) in the data. The experiments were performed on AMD Opteron servers with 2.2GHz per core and 256GB RAM. In the experiments, the F1 measure is used to measure accuracy of the obtained clusterings [5,13], and the values are averages of three runs.



(a) relevant dimensions



(b) relevant points in time

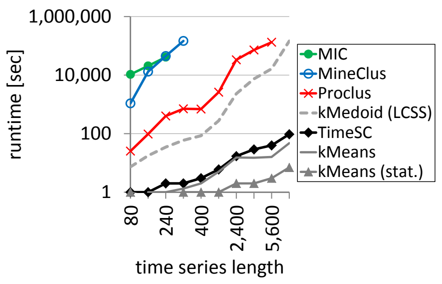
Fig. 5. Performance w.r.t. different number of relevant dimensions and points in time

5.1 Evaluation w.r.t. Effectiveness

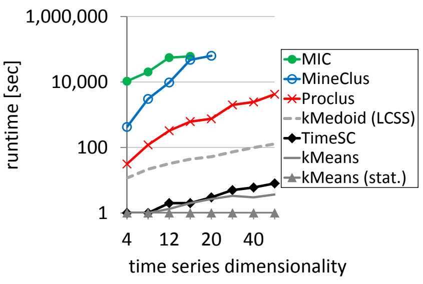
Performance on different variations of our standard dataset is analyzed in Fig. 4 and Fig. 5. In Fig. 4(a) and 4(b) we change the dataset size by enlarging either the length of the included time series or the number of attributes per time series. In both experiments, our TimeSC outperforms all competing solutions by a significant margin. Runner ups are the 2D subspace clustering algorithms MineClus and Proclus, which achieve about 80% accuracy. The standard fullspace clustering approach kMeans also performs surprisingly well with about 60% accuracy. The statistical kMeans approach, which was specifically introduced for clustering multivariate time series, however, performs worse than the original kMeans approach. The Triclustering (3D) approach MIC is outperformed by both kMeans variants. This was not expected, as MIC is designed for temporal data. And finally, the LCSS-based kMedoid only achieves about 20% accuracy in both experiments.

Next, we enlarge the data by increasing the number of clusters (Fig. 4(c)) and by increasing the number of time series per cluster (Fig. 4(d)). With an increasing number of clusters, TimeSC and Proclus achieve stable results, while the accuracies of the other competing approaches continuously sink. For an increasing number of time series per cluster, all the algorithms achieve stable results. Overall, TimeSC outperforms the competing solutions in all settings.

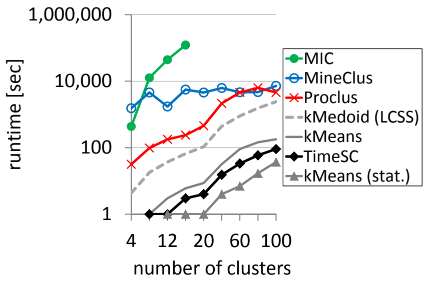
In Fig. 5(a) and 5(b) we change the number of relevant dimensions per cluster and the number of relevant points in time per cluster, i.e. the minimal cluster length. Overall, the obtained accuracies are similar to the preceding experiments. TimeSC outperforms the other methods, and from these methods only the 2D subspace clustering algorithms can achieve stable results of about 80% accuracy. Also, as expected, with increasing relevant dimensions and relevant points in time, finding clusters in the data becomes simpler which is expressed by the strong increase in accuracy for the standard kMeans algorithm. This, however, does not hold for the statistical kMeans, whose accuracy sinks with increasing relevant dimensions and points in time.



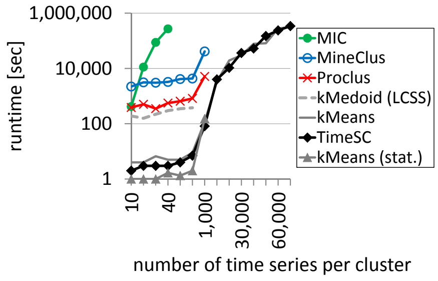
(a) data set length



(b) data set dimensionality



(c) number of clusters



(d) number of time series per cluster

Fig. 6. Efficiency comparison (log. scale)

5.2 Evaluation w.r.t. Efficiency

Our algorithm is designed for larger datasets. To show this, we scaled the experiments from Fig. 4 to much higher values, as shown in Fig. 6. For example, the database size for the last step (70,000 time series) in Fig. 6(d) is 12.32 GB. The experiments illustrate that only the kMeans algorithms and TimeSC are suitable for the larger datasets. Due to the high runtimes, many values could not be obtained for the other approaches. From the subspace and triclustering algorithms, only Proclus shows acceptable runtimes, while the results of MineClus and MIC indicate that these algorithms are not applicable for larger datasets.

6 Conclusion

We introduced a novel model for subspace clustering of multivariate time series data. The clusters in our model are formed by individual sets of relevant intervals per dimension, which together fulfill temporal coherence. We develop a redundancy model to avoid structurally similar clusters and introduce an approximate algorithm for generating clusterings according to our novel model. In the experimental comparison, we showed that our approach is efficient and generates clusterings of higher quality than the competing methods.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University.

References

1. Aggarwal, C.C., Procopiuc, C.M., Wolf, J.L., Yu, P.S., Park, J.S.: Fast algorithms for projected clustering. In: ACM SIGMOD, pp. 61–72 (1999)
2. Dasu, T., Swayne, D.F., Poole, D.: Grouping Multivariate Time Series: A Case Study. In: IEEE ICDMW, pp. 25–32 (2005)
3. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB 1(2), 1542–1552 (2008)
4. Fu, T.: A review on time series data mining. Engineering Applications of Artificial Intelligence 24(1), 164–181 (2011)
5. Günnemann, S., Färber, I., Müller, E., Assent, I., Seidl, T.: External evaluation measures for subspace clustering. In: ACM CIKM, pp. 1363–1372 (2011)
6. Hu, Z., Bhatnagar, R.: Algorithm for discovering low-variance 3-clusters from real-valued datasets. In: IEEE ICDM, pp. 236–245 (2010)
7. Jiang, D., Pei, J., Ramanathan, M., Tang, C., Zhang, A.: Mining coherent gene clusters from gene-sample-time microarray data. In: ACM SIGKDD, pp. 430–439 (2004)
8. Jiang, H., Zhou, S., Guan, J., Zheng, Y.: gTRICLUSTER: A More General and Effective 3D Clustering Algorithm for Gene-Sample-Time Microarray Data. In: Li, J., Yang, Q., Tan, A.-H. (eds.) BioDM 2006. LNCS (LNBI), vol. 3916, pp. 48–59. Springer, Heidelberg (2006)
9. Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., Pfahringer, B.: An effective evaluation measure for clustering on evolving data streams. In: ACM SIGKDD, pp. 868–876 (2011)
10. Kriegel, H. P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM TKDD 3(1) (2009)
11. Liao, T.W.: Clustering of time series data - a survey. Pattern Recognition 38(11), 1857–1874 (2005)
12. Minnen, D., Starner, T., Essa, I.A., Isbell, C.: Discovering characteristic actions from on-body sensor data. In: IEEE ISWC, pp. 11–18 (2006)
13. Müller, E., Günnemann, S., Assent, I., Seidl, T.: Evaluating clustering in subspace projections of high dimensional data. PVLDB 2(1), 1270–1281 (2009)
14. Oates, T.: Identifying distinctive subsequences in multivariate time series by clustering. In: ACM SIGKDD, pp. 322–326 (1999)
15. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.M.: A monte carlo algorithm for fast projective clustering. In: ACM SIGMOD, pp. 418–427 (2002)
16. Sim, K., Aung, Z., Gopalkrishnan, V.: Discovering correlated subspace clusters in 3D continuous-valued data. In: IEEE ICDM, pp. 471–480 (2010)
17. Singhal, A., Seborg, D.: Clustering multivariate time-series data. Journal of Chemometrics 19(8), 427–438 (2005)
18. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: IEEE ICDE, pp. 673–684 (2002)
19. Wang, X., Wirth, A., Wang, L.: Structure-based statistical features and multivariate time series clustering. In: IEEE ICDM, pp. 351–360 (2007)
20. Wu, E.H.C., Yu, P.L.H.: Independent Component Analysis for Clustering Multivariate Time Series Data. In: Li, X., Wang, S., Dong, Z.Y. (eds.) ADMA 2005. LNCS (LNAI), vol. 3584, pp. 474–482. Springer, Heidelberg (2005)
21. Yiu, M.L., Mamoulis, N.: Frequent-pattern based iterative projected clustering. In: IEEE ICDM, pp. 689–692 (2003)
22. Zhao, L., Zaki, M.J.: TriCluster: An effective algorithm for mining coherent clusters in 3D microarray data. In: ACM SIGMOD, pp. 694–705 (2005)