COBRAS$^{\text{TS}}$: A new approach to Semi-Supervised Clustering of Time Series

Toon Van Craenendonck, Wannes Meert, Sebastijan Dumančić and
Hendrik Blockeel

KU Leuven, Department of Computer Science
{firstname.lastname}@kuleuven.be

Abstract. Clustering is ubiquitous in data analysis, including analysis of time series. It is inherently subjective: different users may prefer different clusterings for a particular dataset. Semi-supervised clustering addresses this by allowing the user to provide examples of instances that should (not) be in the same cluster. This paper studies semi-supervised clustering in the context of time series. We show that COBRAS, a state-of-the-art active semi-supervised clustering method, can be adapted to this setting. We refer to this approach as COBRAS$^{\text{TS}}$. An extensive experimental evaluation supports the following claims: (1) COBRAS$^{\text{TS}}$ far outperforms the current state of the art in semi-supervised clustering for time series, and thus presents a new baseline for the field; (2) COBRAS$^{\text{TS}}$ can identify clusters with separated components; (3) COBRAS$^{\text{TS}}$ can identify clusters that are characterized by small local patterns; (4) actively querying a small amount of semi-supervision can greatly improve clustering quality for time series; (5) the choice of the clustering algorithm matters (contrary to earlier claims in the literature).

1 Introduction

Clustering is ubiquitous in data analysis. There is a large diversity in algorithms, loss functions, similarity measures, etc. This is partly due to the fact that clustering is inherently subjective: in many cases, there is no single correct clustering, and different users may prefer different clusterings, depending on their goals and prior knowledge [17]. Depending on their preference, they should use the right algorithm, similarity measure, loss function, hyperparameter settings, etc. This requires a fair amount of knowledge and expertise on the user's side.

Semi-supervised clustering methods deal with this subjectiveness in a different manner. They allow the user to specify constraints that express their subjective interests [18]. These constraints can then guide the algorithm towards solutions that the user finds interesting. Many such systems obtain these constraints by asking the user to answer queries of the following type: *should these two elements be in the same cluster?* A so-called must-link constraint is obtained if the answer is yes, a cannot-link otherwise. In many situations, answering this type of questions is much easier for the user than selecting the right algorithm, defining the similarity measure, etc. Active semi-supervised clustering methods

aim to limit the number of queries that is required to obtain a good clustering by selecting informative pairs to query.

In the context of clustering *time series*, the subjectiveness of clustering is even more prominent. In some contexts, the time scale matters, in other contexts it does not. Similarly, the scale of the amplitude may (not) matter. One may want to cluster time series based on certain types of qualitative behavior (monotonic, periodic, . . .), local patterns that occur in them, etc. Despite this variability, and although there is a plethora of work on time series clustering, semi-supervised clustering of time series has only very recently started receiving attention [7].

In this paper, we show that COBRAS, an existing active semi-supervised clustering system, can be used practically "as-is" for time series clustering. The only adaptation that is needed, is plugging in a suitable similarity measure and a corresponding (unsupervised) clustering approach for time series. Two plug-in methods are considered for this: spectral clustering using dynamic time warping (DTW), and k-Shape [11]. We refer to COBRAS with one of these plugged in as COBRAS^{TS} (COBRAS for Time Series). We perform an extensive experimental evaluation of this approach.

The main contributions of the paper are twofold. First, it contributes a novel approach to semi-supervised clustering of time series, and two freely downloadable, ready-to-use implementations of it. Second, the paper provides extensive evidence for the following claims: (1) COBRAS^{TS} outperforms cDTW^{SS} (the current state of the art) by a large margin; (2) COBRAS^{TS} can identify clusters with separated components; (3) COBRAS^{TS} can identify clusters that are characterized by small local patterns; (4) actively querying a small amount of supervision can greatly improve results in time series clustering; (5) the choice of clustering algorithm matters, it is not negligible compared to the choice of similarity. Except for claim 4, all these claims are novel, and some are at variance with the current literature. Claim 4 has been made before, but with much weaker empirical support.

2   Related work

Semi-supervised clustering has been studied extensively for clustering attribute-value data, starting with COP-KMeans [18]. Most semi-supervised methods extend unsupervised ones by adapting their clustering procedure [18], their similarity measure [20], or both [2]. Alternatively, constraints can also be used to select and tune an unsupervised clustering algorithm [13].

Traditional methods assume that a set of pairwise queries is given prior to running the clustering algorithm, and in practice, pairs are often queried randomly. Active semi-supervised clustering methods try to query the most informative pairs first, instead of random ones [9]. Typically, this results in better clusterings for an equal number of queries. COBRAS [15] is a recently proposed method that was shown to be effective for clustering attribute-value data. In this paper, we show that it can be used to cluster time series with little modification. We describe COBRAS in more detail in the next section.

In contrast to the wealth of papers in the attribute-value setting, only one method has been proposed specifically for semi-supervised time series clustering with active querying. cDTW^{SS} [7] uses pairwise constraints to tune the warping width parameter w in constrained DTW. We compare COBRAS^{TS} to this method in the experiments.

In contrast to semi-supervised time series clustering, semi-supervised time series *classification* has received significant attention [19]. Note that these two settings are quite different: in semi-supervised classification, the set of classes is known beforehand, and at least one labeled example of each class is provided. In semi-supervised clustering, it is not known in advance how many classes (clusters) there are, and a class may be identified correctly even if none of its instances have been involved in the pairwise constraints.

3   Clustering time series with COBRAS

3.1   COBRAS

We describe COBRAS only to the extent necessary to follow the remainder of the paper; for more information, see Van Craenendonck et al. [14, 15].

COBRAS is based on two key ideas. The first [14] is that of *super-instances*: sets of instances that are temporarily assumed to belong to the same cluster in the unknown target clustering. In COBRAS, a clustering is a set of clusters, each cluster is a set of super-instances, and each super-instance is a set of instances. Super-instances make it possible to exploit constraints much more efficiently: querying is performed at the level of super-instances, which means that each instance does not have to be considered individually in the querying process. The second key idea in COBRAS [15] is that of the *automatic detection of the right level* at which these super-instances are constructed. For this, it uses an iterative refinement process. COBRAS starts with a single super-instance that contains all the examples, and a single cluster containing that super-instance. In each iteration the largest super-instance is taken out of its cluster, split into smaller super-instances, and the latter are reassigned to (new or existing) clusters. Thus, COBRAS constructs a clustering of super-instances at an increasingly fine-grained level of granularity. The clustering process stops when the query budget is exhausted.

We illustrate this procedure using the example in Figure 1. Panel A shows a toy dataset that can be clustered according to several criteria. We consider differentiability and monotonicity as relevant properties. Initially, all instances belong to a single super-instance (S_0), which constitutes the only cluster (C_0). The second and third rows of Figure 1 show two iterations of COBRAS.

In the first step of iteration 1, COBRAS refines S_0 into 4 new super-instances, which are each put in their own cluster (panel B). The refinement procedure uses k-means, and the number of super-instances in which to split is determined based on constraints; for details, see [15]. In the second step of iteration 1, COBRAS determines the relation between new and existing clusters. To determine the
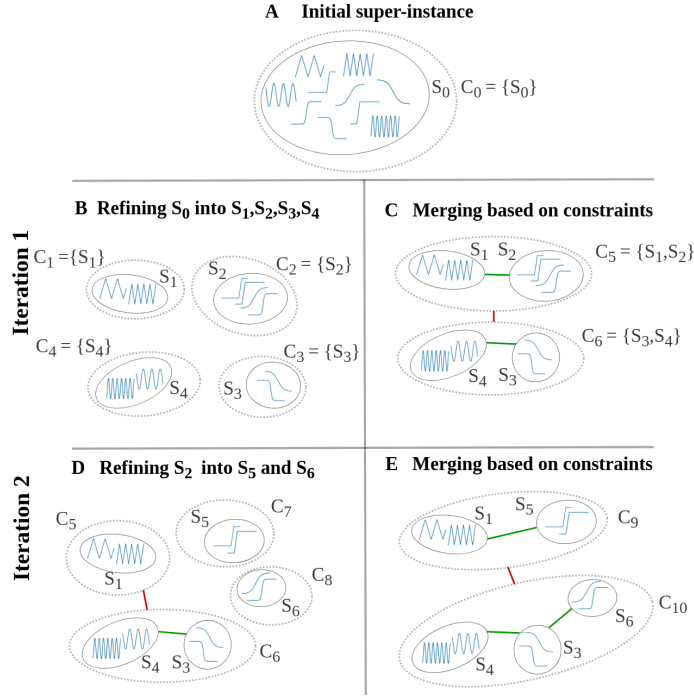
Fig. 1. An illustration of the COBRAS clustering procedure.

relation between two clusters, COBRAS queries the pairwise relation between the medoids of their closest super-instances. In this example, we assume that the user is interested in a clustering based on differentiability. The relation between $C_1 = \{S_1\}$ and $C_2 = \{S_2\}$ is determined by posing the following query: *should* ⧠ *and* ⧠ *be in the same cluster?* The user answers yes, so C_1 and C_2 are merged into C_5. Similarly, COBRAS determines the other pairwise relations between clusters. It does not need to query all of them, many can be derived through transitivity or entailment [15]. The first iteration ends once all pairwise relations between clusters are known. This is the situation depicted in panel C. Note that COBRAS has not produced a perfect clustering at this point, as S_2 contains both differentiable and non-differentiable instances.

In the second iteration, COBRAS again starts by refining its largest super-instance. In this case, S_2 is refined into S_5 and S_6, as illustrated in panel D. A new cluster is created for each of these super-instances, and the relation between new and existing clusters is determined by querying pairwise constraints. A must-link between S_5 and S_1 results in the creation of $C_9 = \{S_1, S_5\}$. Similarly, a must-link between S_6 and S_3 results in the creation of $C_{10} = \{S_3, S_4, S_6\}$. At this point, the second iteration ends as all pairwise relations between clusters are known. The clustering consists of two clusters, and a data granularity of 5 super-instances was needed.
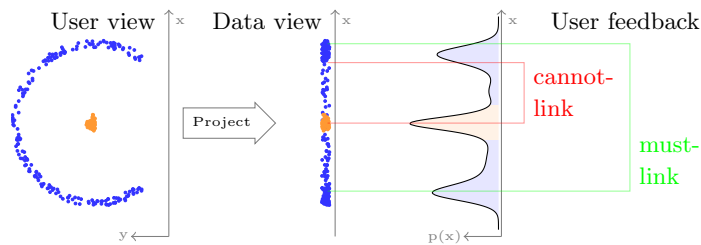
Fig. 2. Clusters may contain separated components when projected on a lower-dimensional subspace.

In general, COBRAS keeps repeating its two steps (refining super-instances and querying their pairwise relations) until the query budget is exhausted.

Separated components A noteworthy property of COBRAS is that, by interleaving splitting and merging, it can split off a subcluster from a cluster and reassign it to another cluster. In this way, it can construct clusters that contain *separated components* (different dense regions that are separated by a dense region belonging to another cluster). It may, at first, seem strange to call such a structure a "cluster", as clusters are usually considered to be coherent high-density areas. However, note that a coherent cluster may become incoherent when projected onto a subspace. Figure 2 illustrates this. Two clusters are clearly visible in the XY-space, yet projection on the X-axis yields a trimodal distribution where the outer modes belong to one cluster and the middle mode to another. In semi-supervised clustering, it is realistic that the user evaluates similarity on the basis of more complete information than explicitly present in the data; coherence in the user's mind may therefore not translate to coherence in the data space[1].

The need for handling clusters with multi-modal distributions has been mentioned repeatedly in work on time series anomaly detection [5], on unsupervised time series clustering [11], and on attribute-value semi-supervised constrained clustering [12]. Note, however, a subtle difference between having a multi-modal distribution and containing separated components: the first assumes that the components are separated by a low-density area, whereas the second allows them to be separated by a dense region of instances from another cluster.

3.2 COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$

COBRAS is not suited out-of-the-box for time series clustering, for two reasons. First, it defines the super-instance medoids w.r.t. the *Euclidean* distance, which is well-known to be suboptimal for time series. Second, it uses k-means to refine

---

[1] Note that Figure 2 is just an illustration; it can be difficult to express the more complete information explicitly as an additional dimension, as is done in the figure.

super-instances, which is known to be sub-state-of-the-art for time series clustering [11]. Both of these issues can easily be resolved by plugging in distance measures and clustering methods that are developed specifically for time series. We refer to this approach as COBRAS$^{\text{TS}}$. We now present two concrete instantiations of it: COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$. Other instantiations can be made, but we develop these two as DTW and k-Shape represent the state of the art in unsupervised time series clustering.

Algorithm 1 COBRAS$^{\text{DTW}}$

Input: A dataset, the DTW warping window width w, the γ parameter used in converting distances to similarities and access to an oracle answering pairwise queries
Output: A clustering
1: Compute the full pairwise DTW distance matrix
2: Convert each distance d to an affinity a: $a_{i,j} = e^{-\gamma d_{i,j}}$
3: Run COBRAS, substituting k-means for splitting super-instances with spectral clustering on the previously computed affinity matrix

---

COBRAS$^{\text{DTW}}$ uses DTW as its distance measure, and spectral clustering to refine super-instances. It is described in Algorithm 1. DTW is commonly accepted to be a competitive distance measure for time series analysis [1], and spectral clustering is well-known to be an effective clustering method [16]. We use the constrained variant of DTW, cDTW, which restricts the amount by which the warping path can deviate from the diagonal in the warping matrix. cDTW offers benefits over DTW in terms of both runtime and solution quality [11, 7], if run with an appropriate window width.

COBRAS$^{\text{k-Shape}}$ uses the shape-based distance (SBD, [11]) as its distance measure, and the corresponding k-Shape clustering algorithm [11] to refine super-instances. k-Shape can be seen as a k-means variant developed specifically for time series. It uses SBD instead of the Euclidean distance, and comes with a method of computing cluster centroids that is tailored to time series. k-Shape was shown to be an effective and scalable method for time series clustering in [11]. Instead of the medoid, COBRAS$^{\text{k-Shape}}$ uses the instance that is closest to the SBD centroid as a super-instance representative.

4 Experiments

In our experiments we evaluate COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$ in terms of clustering quality and runtime, and compare them to state-of-the-art semi-supervised (cDTW$^{\text{SS}}$ and COBS) and unsupervised (k-Shape and k-MS) competitors. Our experiments are fully reproducible: we provide code for COBRAS$^{\text{TS}}$ in a public repository[2], and a separate repository for our experimental setup[3]. The experiments are performed on the public UCR collection [6].

---

[2] https://bitbucket.org/toon_vc/cobras_ts or using `pip install cobras_ts`
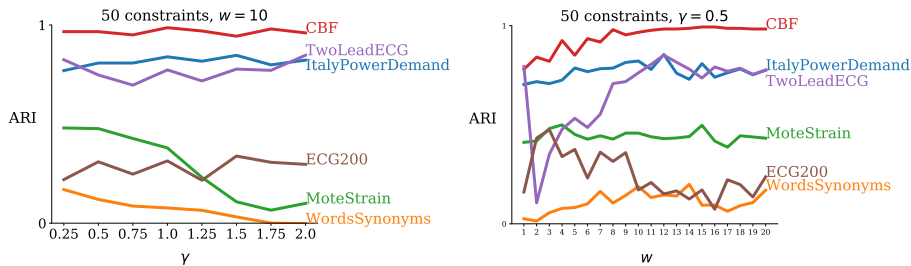[3] https://bitbucket.org/toon_vc/cobras_ts_experiments

Fig. 3. Sensitivity to γ and w for several datasets.

4.1 Methods

COBRAS^{TS} COBRAS^{k-Shape} has no parameters (the number of clusters used in k-Shape to refine super-instances is chosen based on the constraints in CO-BRAS). We use a publicly available Python implementation[4] to obtain the k-Shape clusterings. COBRAS^{DTW} has two parameters: γ (used in converting distances to affinities) and w (the warping window width). We use a publicly available C implementation to construct the DTW distance matrices [10]. In our experiments, γ is set to 0.5 and w to 10% of the time series length. The value $w = 10\%$ was chosen as Dau et al. [7] report that most datasets do not require w greater than 10%. We note that γ and w could in principle also be tuned for COBRAS^{DTW}. There is, however, no well-defined way of doing this. We cannot use the constraints for this, as they are actively selected during the execution of the algorithm (which of course requires the affinity matrix to already be constructed). We did not do any tuning on these parameters, as this is also hard in a practical clustering scenario, but observed that the chosen parameter values already performed very well in the experiments. We performed a parameter sensitivity analysis, illustrated in Figure 3, which shows that the influence of these parameters is highly dataset-dependent: for many datasets their values do not matter much, for some they result in large differences.

cDTW^{SS} cDTW^{SS} uses pairwise constraints to tune the w parameter in cDTW. In principle, the resulting tuned cDTW measure can be used with any clustering algorithm. The authors in [7] use it in combination with TADPole [4], and we do the same here. We use the code that is publicly available on the authors' website[5]. The cutoff distances used in TADPole were obtained from the authors in personal communication.

COBS COBS [13] uses constraints to select and tune an unsupervised clustering algorithm. It was originally proposed for attribute-value data, but it can trivially be modified to work with time series data as follows. First, the full pairwise distance matrix is generated with cDTW using $w = 10\%$ of the time series length. Next, COBS generates clusterings by varying the hyperparame-

---

[4] https://github.com/Mic92/kshape

[5] https://sites.google.com/site/dtwclustering/

ters of several standard unsupervised clustering methods, and selects the resulting clustering that satisfies the most pairwise queries. We use the active variant of COBS, as described in [13]. Note that COBS is conceptually similar to cDTW^{SS}, as both methods use constraints for hyperparameter selection. The important difference is that COBS uses a fixed distance measure and selects and tunes the clustering algorithm, whereas cDTW^{SS} tunes the similarity measure and uses a fixed clustering algorithm. We use the following unsupervised clustering methods and corresponding hyperparameter ranges in COBS: spectral clustering ($K \in [\max(2, K_{true} - 5), K_{true} + 5]$), hierarchical clustering ($K \in [\max(2, K_{true} - 5), K_{true} + 5]$, with both average and complete linkage), affinity propagation (`damping` $\in [0.5, 1.0]$) and DBSCAN ($\epsilon \in$ [`min pairwise dist.`, `max. pairwise dist`], `min_samples` $\in [2, 21]$). For the continuous parameters, clusterings were generated for 20 evenly spaced values in the specified intervals. Additionally, the γ parameter in converting distances to affinities was varied in $[0, 2.0]$ for clustering methods that take affinities as input, which are all of them except DBSCAN, which works with distances. We did not vary the warping window width w for generating clusterings in COBS. This would mean a significant further increase in computation time, both for generating the DTW distance matrices, and for generating clusterings with all methods and parameter settings for each value of w.

k-Shape and k-MS Besides the three previous semi-supervised methods, we also include k-Shape [11] and k-MultiShape (k-MS) [11] in our experiments as unsupervised baselines. k-MS [11] is similar to k-Shape, but uses multiple centroids, instead of one, to represent each cluster. It was found to be the most accurate method in an extensive experimental study that compares a large number of unsupervised time series clustering methods on the UCR collection [11]. The number of centroids that k-MS uses to represent a cluster is a parameter; following the original paper we set it to 5 for all datasets. The k-MS code was obtained from the authors.

4.2   Data

We perform experiments on the entire UCR time series classification collection [6], which is the largest public collection of time series datasets. It consists of 85 datasets from a wide variety of domains. The UCR datasets come with a predefined training and test set. We use the test sets as our datasets as they are often much bigger than the training sets. This means that whenever we refer to a dataset in the remainder of this text, we refer to the test set of that dataset as defined in [6]. This procedure was also followed by Dau et al. [7].

As is typically done in evaluating semi-supervised clustering methods, the classes are assumed to represent the clusterings of interests. When computing rankings and average ARIs, we ignored results from 21 datasets where cDTW^{SS} either crashed or timed out after 24h.[6]

[6] These datasets are listed at https://bitbucket.org/toon_vc/cobras_ts_experiments

4.3 Methodology

We use 10-fold cross-validation, as is common in evaluating semi-supervised clustering methods [3, 9]. The full dataset is clustered in each run, but the methods can only query pairs of which both instances are in the training set. The result of a run is evaluated by computing the Adjusted Rand Index (ARI) [8] on the instances of the test set. The ARI measures the similarity between the generated clusterings and the ground-truth clustering, as indicated by the class labels. It is 0 for a random clustering, and 1 for a perfect one. The final ARI scores that are reported are the average ARIs over the 10 folds.

We ensure that $cDTW^{SS}$ and COBS do not query pairs that contain instances from the test set by simply excluding such candidates from the list of constraints that they consider. For $COBRAS^{TS}$, we do this by only using training instances to compute the super-instance representatives.

$COBRAS^{TS}$ and COBS do not require the number of clusters as an input parameter, whereas $cDTW^{SS}$, k-Shape and k-MS do. The latter three were given the correct number of clusters, as indicated by the class labels. Note that this is a significant advantage for these algorithms, and that in many practical applications the number of clusters is not known beforehand.
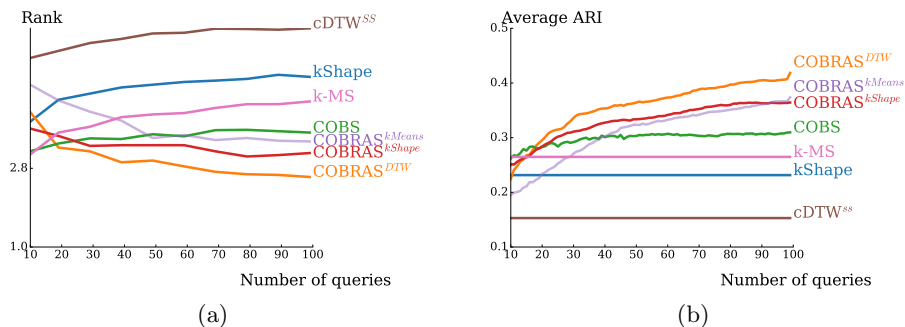
Fig. 4. (a) Average rank over all clustering tasks. Lower is better. (b) Average ARI. Higher is better.

4.4 Results

Clustering quality Figure 4(a) shows the average ranks of the compared methods over all datasets. Figure 4(b) shows the average ARIs. Both plots clearly show that, on average, $COBRAS^{TS}$ outperforms all the competitors by a large margin. Only when the number of queries is small (roughly < 15), is it outperformed by COBS and k-MS.

For completeness, we also include vanilla COBRAS (denoted as $COBRAS^{kMeans}$) in the comparison in Figure 4. Given enough queries (roughly > 50), $COBRAS^{kMeans}$ outperforms all competitors other than $COBRAS^{TS}$. This indicates that the CO-

BRAS approach is essential. As expected, however, COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{kShape}}$ significantly outperform COBRAS$^{\text{kMeans}}$.

These observations are confirmed by Table 1, which reports the number of times COBRAS$^{\text{DTW}}$ wins and loses against the alternatives. The differences with cDTW$^{\text{SS}}$ and k-Shape are significant for all the considered numbers of queries (Wilcoxon test, $p < 0.05$). The difference between COBRAS$^{\text{DTW}}$ and COBS is significant for 50 and 100 queries, but not for 25. The same holds for COBRAS$^{\text{DTW}}$ vs. k-MS. This confirms the observation from Figure 4(a), which showed that the performance gap between COBRAS$^{\text{DTW}}$ and the competitors becomes larger as more queries are answered. The difference between COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$ is only statistically significant for 100 queries.

Table 1. Wins and losses over the 64 datasets. An asterisk indicates that the difference is significant according to the Wilcoxon test with $p < 0.05$.

	25 queries		50 queries		100 queries	
	win	loss	win	loss	win	loss
COBRAS$^{\text{DTW}}$ vs. COBRAS$^{\text{k-Shape}}$	**35**	29	**37**	27	**41***	23
COBRAS$^{\text{DTW}}$ vs. COBRAS$^{\text{k-Means}}$	**41***	23	**36**	28	**40***	24
COBRAS$^{\text{DTW}}$ vs. k-MS	**35**	29	**40***	24	**47***	14
COBRAS$^{\text{DTW}}$ vs. COBS	**37**	27	**42***	22	**45***	19
COBRAS$^{\text{DTW}}$ vs. cDTW$^{\text{SS}}$	**62***	2	**53***	11	**55***	9
COBRAS$^{\text{DTW}}$ vs. k-Shape	**40***	24	**46***	18	**50***	14

Surprisingly, the unsupervised baselines outperform the semi-supervised cDTW$^{\text{SS}}$. This is inconsistent with the claim that the choice of w dwarfs any improvements by the k-Shape algorithm [7]. To ensure that this is not an effect of the evaluation strategy (10-fold CV using the ARI, compared to no CV and the Rand index (RI) in [7]), we have also computed the RIs for all of the clusterings generated by k-Shape and compared them directly to the values provided by the authors of cDTW$^{\text{SS}}$ on their webpage[7]. In this experiment k-Shape attained an average RI of 0.68, cDTW$^{\text{SS}}$ 0.67. We note that the claim in [7] was based on a comparison on two datasets. Our experiments clearly indicate that it does not generalize towards all datasets.

Runtime COBRAS$^{\text{DTW}}$, cDTW$^{\text{SS}}$ and COBS require the construction of the pairwise DTW distance matrix. This becomes infeasible for large datasets. For example, computing one distance matrix for the ECG5000 dataset took ca. 30h in our experiments, using an optimized C implementation of DTW.

k-Shape and k-MS are much more scalable [11], as they do not require computing a similarity matrix. COBRAS$^{\text{k-Shape}}$ inherits this scalability, as it uses k-Shape to refine super-instances. In our experiments, COBRAS$^{\text{k-Shape}}$ was on average 28 times faster than COBRAS$^{\text{DTW}}$.

---

[7] https://sites.google.com/site/dtwclustering/

# 5   Case studies: CBF, TwoLeadECG and MoteStrain

To gain more insight into why COBRAS$^{\text{TS}}$ outperforms its competitors, we inspect the clusterings that are generated for three UCR datasets in more detail: CBF, TwoLeadECG and MoteStrain. CBF and TwoLeadECG are examples for which COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$ significantly outperform their competitors, whereas MoteStrain is one of the few datasets for which they are outperformed by unsupervised k-Shape clustering. These three datasets illustrate different reasons why time series clustering may be difficult: CBF because one of the clusters comprises two separated subclusters; TwoLeadECG, because only limited *subsequences* of the time series are relevant for the clustering at hand, and the remaining parts obfuscate the distance measurements; and MoteStrain because it is noisy.

CBF   The first column of Figure 5 shows the "true" clusters as they are indicated by the class labels. It is clear that the classes correspond to three distinct patterns (horizontal, upward and downward). The next columns show the clusterings that are produced by each of the competitors. Semi-supervised approaches are given a budget of 50 queries. COBRAS$^{\text{DTW}}$ and COBRAS$^{\text{k-Shape}}$ are the only methods that provide a near perfect solution (ARI = 0.96). cDTW$^{\text{SS}}$ mixes patterns of different types in each cluster. COBS find pure clusters, but too many: the plot only shows the largest three of 15 clusters for COBS. k-Shape and k-MS mix horizontal and downward patterns in their third cluster. To clarify this mixing of patterns, the figure shows the instances in the third k-Shape and k-MS clusters again, but separated according to their true class.
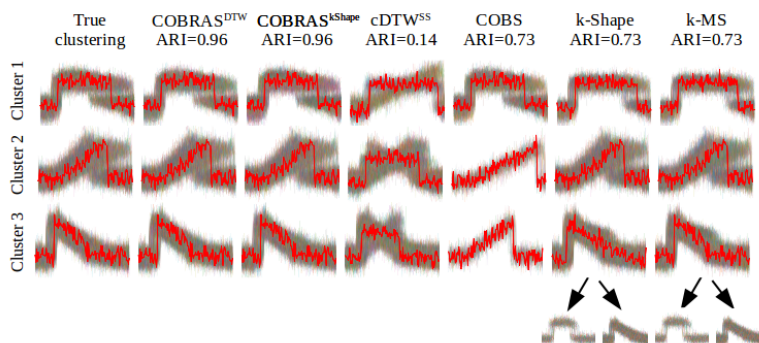
Fig. 5. The first column shows the true clustering of CBF. The remaining columns show the clusterings that are produced by all considered methods. For COBS, only the three largest of 15 clusters are shown. All the cluster instances are plotted, the prototypes are shown in red. For COBRAS$^{\text{DTW}}$, cDTW$^{\text{SS}}$ and COBS the prototypes are selected as the medoids w.r.t. DTW distance. For the others the prototypes are the medoids w.r.t. the SBD distance.

Figure 6 illustrates how repeated refinement of super-instances helps COBRAS[TS] deal with the complexities of clustering CBF. It shows a super-instance in the root, with its subsequent refinements as children. The super-instance in the root, which is itself a result of a previous split, contains horizontal and upward patterns. Clustering it into two new super-instances does not yield a clean separation of these two types: a pure cluster with upward patterns is created, but the other super-instance still mixes horizontal and upward patterns. This is not a problem for COBRAS[TS], as it simply refines the latter super-instance again. This time the remaining instances are split into nearly pure ones separating horizontal from upward patterns. Note that the two super-instances containing upward patterns correspond to two distinct subclusters: some upward patterns drop down very close to the end of the time series, whereas the drop in the other subcluster occurs much earlier.
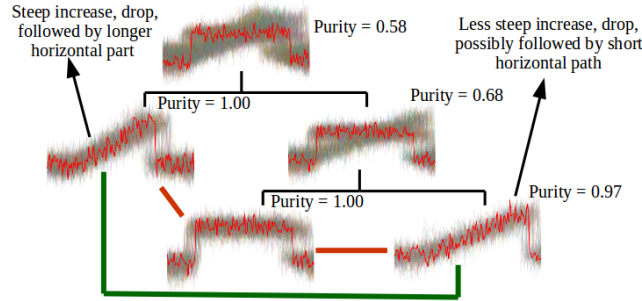
Fig. 6. A super-instance that is generated while clustering CBF, and its refinements. The green line indicates a must-link, and illustrates that these two super-instances will be part of the same multi-modal cluster (that of upward patterns). The red lines indicate cannot-links. The purity of a super-instance is computed as the ratio of the occurrence of its most frequent class, over its total number of instances.

The clustering process just mentioned illustrates the point made earlier, in Section 3.1, about COBRAS's ability to construct clusters with separated components. It is clear that this ability is advantageous in the CBF dataset. Note that being able to deal with *separated* components is key here; k-MS, which is able to find multi-modal clusters, but not clusters with modes that are separated by a mode from another cluster, produces a clustering that is far from perfect for CBF.

Figure 6 also illustrates that COBRAS's super-instance refinement step is similar to top-down hierarchical clustering. Note, however, that COBRAS uses constraints to guide this top-down splitting towards an appropriate level of granularity. Furthermore, this refinement is only one of COBRAS's components; it is interleaved with a bottom-up merging step to combine the super-instances into actual clusters [15].
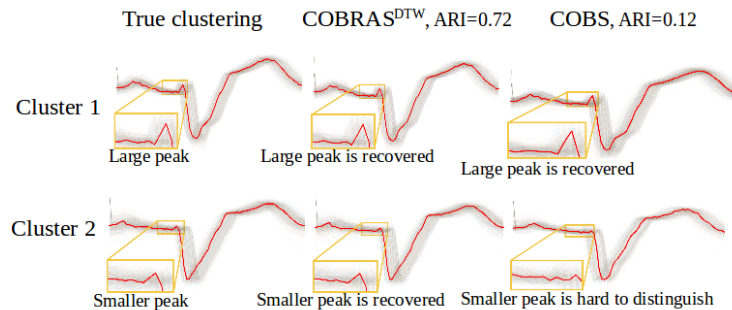
Fig. 7. The first column shows the "true" clustering of TwoLeadECG. The second column shows the clustering produced by COBRAS$^{\mathrm{DTW}}$. The third column shows the clustering produced by COBS, which is the best competitor for this dataset. Prototypes are shown in red, and are the medoids w.r.t. the DTW distance.

TwoLeadECG The first column in Figure 7 shows the "true" clusters for TwoLeadECG. Cluster 1 is defined by a large peak before the drop, and a slight bump in the upward curve after the drop. Instances in cluster 2 typically only show a small peak before the drop, and no bump in the upward curve after the drop. For the remainder of the discussion we focus on the peak as the defining pattern, simply because it is easier to see than the more subtle bump.

The second column in Figure 7 shows the clustering that is produced by COBRAS$^{\mathrm{DTW}}$; the one produced by COBRAS$^{\mathrm{k\text{-}Shape}}$ is highly similar. They are the only methods able to recover these characteristic patterns. The last column in Figure 7 shows the clustering that is produced by COBS, which is the best of the competitors. This clustering has an ARI of 0.12, which is not much better than random. From the zoomed insets in Figure 7, it is clear that this clustering does not recover the defining patterns: the small peak that is characteristic for cluster 2 is hard to distinguish.

This example illustrates that by using COBRAS$^{\mathrm{TS}}$ for semi-supervised clustering, a domain expert can discover more accurate explanatory patterns than with competing methods. None of the alternatives is able to recover the characteristic patterns in this case, potentially leaving the domain expert with an incorrect interpretation of the data. Obtaining these patterns comes with relatively little additional effort, as with a good visualizer answering 50 queries only takes a few minutes. This time would probably be insignificant compared to the time that was needed to collect the 1139 instances in the TwoLeadECG dataset.

MoteStrain In our third case study we discuss an example for which COBRAS$^{\mathrm{TS}}$ does not work well, as this provides insight into its limitations. We consider the MoteStrain dataset, for which the unsupervised methods perform best. k-MS attains an ARI of 0.62, and k-Shape of 0.61. COBRAS$^{\mathrm{k\text{-}Shape}}$ ranks third with an ARI of 0.51, and COBRAS$^{\mathrm{DTW}}$ fourth with an ARI of 0.48. These results are surprising, as the COBRAS algorithms have access to more information than
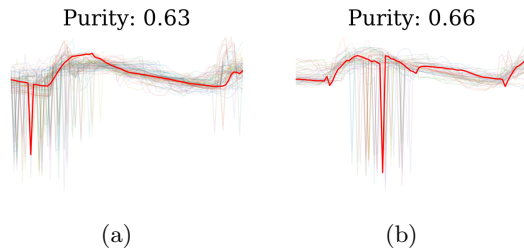
Fig. 8. Two super-instances generated by COBRAS$^{\text{DTW}}$. The super-instances are based on the location of the noise.

the unsupervised k-Shape and k-MS. Figure 8 gives a reason for this outcome; it shows that COBRAS$^{\text{TS}}$ creates super-instances that are based on the location of the noise. The poor performance of the COBRAS$^{\text{TS}}$ variants can in this case be explained by their large variance. The process of super-instance refinement is much more flexible than the clustering procedure of k-Shape, which has a stronger bias. For most datasets, COBRAS$^{\text{TS}}$'s weaker bias led to performance improvements in our experiments, but in this case it has a detrimental effect due to the large magnitude of the noise. In practice, the issue could be alleviated here by simply applying a low-pass filter to remove noise prior to clustering.

6   Conclusion

Time series arise in virtually all disciplines. Consequently, there is substantial interest in methods that are able to obtain insights from them. One of the most prominent ways of doing this, is by using clustering. In this paper we have presented COBRAS$^{\text{TS}}$, an novel approach to time series clustering. COBRAS$^{\text{TS}}$ is semi-supervised: it uses small amounts of supervision in the form of must-link and cannot-link constraints. This sets it apart from the large majority of existing methods, which are unsupervised. An extensive experimental evaluation shows that COBRAS$^{\text{TS}}$ is able to effectively exploit this supervision; it outperforms unsupervised and semi-supervised competitors by a large margin. As our implementation is readily available, COBRAS$^{\text{TS}}$ offers a valuable new tool for practitioners that are interested in analyzing time series data.

Besides the contribution of the COBRAS$^{\text{TS}}$ approach itself, we have also provided insight into why it works well. A key factor in its success is its ability to handle clusters with separated components.

Acknowledgements

References

1. A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.
2. S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of SDM 2004*.
3. S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004.
4. N. Begum, L. Ulanova, J. Wang, and E. Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *Proc. of SIGKDD 2015*.
5. H. Cao, V. Y. F. Tan, and J. Z. F. Pang. A parsimonious mixture of Gaussian trees model for oversampling in imbalanced and multimodal time-series classification. *IEEE Trans. on Neural Networks and Learning Systems*, 25(12):2226–2239, 2014.
6. Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The UCR time series classification archive, July 2015. http://www.cs.ucr.edu/~eamonn/time_series_data/.
7. H. A. Dau, N. Begum, and E. Keogh. Semi-supervision dramatically improves time series clustering under dynamic time warping. In *Proc. of CIKM 2016*.
8. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 1985.
9. P. K. Mallapragada, R. Jin, and A. K. Jain. Active query selection for semi-supervised clustering. In *Proc. of ICPR 2008*.
10. W. Meert. DTAIDistance, Mar. 2018. https://doi.org/10.5281/zenodo.1202379.
11. J. Paparrizos and L. Gravano. Fast and accurate time-series clustering. *ACM Trans. Database Syst.*, 42(2):8:1–8:49, June 2017.
12. M. Śmieja and M. Wiercioch. Constrained clustering with a complex cluster structure. *Advances in Data Analysis and Classification*, 11(3):493–518, 2017.
13. T. Van Craenendonck and H. Blockeel. Constraint-based clustering selection. In *Machine Learning*. 2017.
14. T. Van Craenendonck, S. Dumancic, and H. Blockeel. COBRA: A fast and simple method for active clustering with pairwise constraints. In *Proc. of IJCAI 2017*.
15. T. Van Craenendonck, S. Dumančić, E. Van Wolputte, and H. Block-eel. COBRAS: Fast, Iterative, Active Clustering with Pairwise Constraints. *https://arxiv.org/abs/1803.11060, under submission*, Mar. 2018.
16. U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007.
17. U. von Luxburg, R. C. Williamson, and I. Guyon. Clustering: Science or Art? In *Workshop on Unsupervised Learning and Transfer Learning, 2014*.
18. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proc. of ICML 2001*.
19. L. Wei and E. Keogh. Semi-supervised time series classification. In *Proc. ACM SIGKDD 2006*.
20. E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS 2003*.