

Clustering time series with k -medoids based algorithms

Christopher Holder¹, David Guijo-Rubio^{1,2}, and Anthony Bagnall¹

¹School of Computing Sciences, University of East Anglia, NR4 7TQ, Norwich, UK

²Department of Computer Science, Universidad de Córdoba, Córdoba, Spain
c.holder@uea.ac.uk, dguijo@uco.es, ajb@uea.ac.uk

Abstract. Time Series Clustering (TSCL) involves grouping unlabelled time series into homogeneous groups. A popular approach to TSCL is to use the partitional clustering algorithms k -means or k -medoids in conjunction with an elastic distance function such as Dynamic Time Warping (DTW). We explore TSCL using nine different elastic distance measures. Both partitional algorithms characterise clusters with an exemplar series, but use different techniques to do so: k -means uses an averaging algorithm to find an exemplar, whereas k -medoids chooses a training case (medoid). Traditionally, the arithmetic mean of a collection of time series was used with k -means. However, this ignores any offset. In 2011, an averaging technique specific to DTW, called DTW Barycentre Averaging (DBA), was proposed. Since, k -means with DBA has been the algorithm of choice for the majority of partition-based TSCL and much of the research using medoids-based approaches for TSCL stopped. We revisit k -medoids based TSCL with a range of elastic distance measures. Our results show k -medoids approaches are significantly better than k -means on a standard test suite, independent of the elastic distance measure used. We also compare the most commonly used alternating k -medoids approach against the Partition Around Medoids (PAM) algorithm. PAM significantly outperforms the default k -medoids for all nine elastic measures used. Additionally, we evaluate six variants of PAM designed to speed up TSCL. Finally, we show PAM with the best elastic distance measure is significantly better than popular alternative TSCL algorithms, including the k -means DBA approach, and competitive with the best deep learning algorithms.

Keywords: Time series · clustering · k -means · k -medoids · PAM · UCR archive

1 Introduction

Time Series Clustering (TSCL) is an unsupervised technique where a set of time series, are partitioned into “clusters”, which contain time series considered to be homogeneous. By contrast, time series in different clusters are considered heterogeneous. However, there is no generally accepted definition of a cluster because “clusters are, in large part, in the eye of the beholder” [8]. This is because different users may have different enough needs and intentions to want a

different algorithm and notion of cluster [31]. Therefore, due to the nature of the various users problems and needs, hundreds of clustering algorithms have been proposed. Many of these have been adapted to deal with time series. For instance, alternative transformation based approaches [21], deep learning based clustering algorithms [17] or statistical model based approaches [4], among others, have been proposed for TSCL. Our focus is on partitional clustering based on distance functions used to measure dissimilarity between whole time series.

Measuring dissimilarity is critical to clustering techniques in order to fulfil the objective of any clustering algorithm: it must form internally homogeneous and externally heterogeneous clusters. Measuring homogeneity usually requires a measure of dissimilarity (or similarity) between cases, commonly known as a distance measure.

In traditional clustering, this is normally a correlation based or Minkowski metric such as Euclidean Distance (ED). However, these traditional distances do not take advantage of the unique traits and characteristics of time series data. There has been a popular research topic in designing time series specific distance measures that can be used in clustering (and classification). For example, elastic distances compensate for misalignment creating a path through a cost matrix by either warping or editing time series. The most common and famous elastic distance is Dynamic Time Warping (DTW) [28]. A comparison of nine elastic distance measures [22] found there was little difference in terms of accuracy of classification accuracy when used with a nearest neighbour classifier. For TSCL, DTW is the most popular elastic distance measure, as can be observed in these works [3,10,20,26]. It is most commonly used with k -means clustering (for example [14]), which iteratively assigns cases to clusters with the nearest exemplar, or centroid. Then, the centroid is recalculated from the new membership through averaging. One popular solution for DTW based k -means is to use Dynamic Barycentre Averaging (DBA) [27] to find centroids. This involves aligning cluster members to each other with DTW, then averaging along paths. This improves k -means clustering, but at a high computational cost. An alternative to averaging to find centroids is to select instances, known as medoids, to represent cluster exemplars. The most commonly used k -medoids algorithm tries all of the current cluster members as the exemplar and chooses the one that minimises a specific clusters distance to medoid. In common with the literature, we call this algorithm alternate or alternating k -medoids, although it is sometimes referred to as Lloyds algorithm [23]. k -medoids algorithms have been used much less frequently in the TSCL literature, particularly since DBA was proposed.

Recent research [12] compared the performance of nine elastic distance measures using both k -means and a k -medoids (only using alternating k -medoids). The main conclusion of this work was that two distance functions, Move Split Merge (MSM) [34] and Time Warp Edit (TWE) [24], performed better than other distances with both clustering algorithms. A secondary conclusion was that k -medoids approach generally outperformed k -means. One key feature of k -medoids algorithms is that they require the calculation of the distance matrix between instances prior to clustering. The $O(n^2)$ space complexity can introduce

an unacceptable overhead for large problems. Nevertheless, k -medoids algorithms clearly have a role to play in a large majority of TSCL studies.

Our aim is to explore k -medoids based TSCL. We assess whether k -medoids based TSCL is better than k -means based, regardless of the elastic distance function used. We then explore some of the large number of variants for k -medoids clustering that have not been used in the TSCL literature before. Finally, we compare the performance of the best k -medoids clustering approach to those of popular alternative TSCL algorithms and show them to be significantly better on the UCR archive [5]. Thus, our contributions are summarised as follows:

1. We compare the performance of k -means and standard k -medoids on 112 UCR problems using nine elastic distance measures, focussing on the clustering algorithm rather than the distance function.
2. We provide a survey of variants of k -medoids, aligned with implementations in the `aeon` toolkit¹.
3. We show that the Partition Around Medoids (PAM) [19] algorithm is significantly better than the standard k -medoids approach.
4. We evaluate the impact of a range of PAM refinements.
5. We show that PAM using MSM and TWE is significantly better than popular alternative approaches, and is not worse than the best deep learning model out of over 300 evaluated in [17].

The rest of this paper is structured as follows. Section 2 provides background information into k -medoids based clustering. Section 3 describes the set of elastic distance functions, standing out MSM and TWE. In Section 4, we give an overview of the experimental settings, performance measures and statistical tests used for comparing the methodologies. In Section 5, experimental results for the aforementioned comparisons are presented. Finally, Section 6 summarises our findings and highlights future work.

2 k -medoids based clustering background

k -means and k -medoids are partition based clustering algorithms and share the same basic components. Firstly, the algorithm selects time series, which we call exemplars, that are meant to characterise a cluster. This is known as the *initialisation* stage. After initialisation, there is a process of assigning membership based on distances to exemplars (the *assign* method). Then, exemplars are updated based on new cluster assignments (the *update* stage). These three steps are repeated until some convergence condition is met.

The iteration aims to minimise an error objective function of within class deviation, or Total Deviation (TD), given as follows:

$$TD = \sum_{i=1}^k \sum_{x_c \in C_i} d(x_c, e_i) \quad (1)$$

¹ <https://github.com/aeon-toolkit/aeon/>

where k is the number of clusters, C_i is the set of cases in the i th cluster, d is the dissimilarity measure, x_c is a case in cluster C_i and e_i is the exemplar (representative) of cluster C_i . One disadvantage of k -means for clustering is that because the exemplars are centroids (averaged cluster members), repeated calls to the distance function are required.

k -medoids clustering algorithms use instances from the train data (known as medoids) as the cluster exemplars, and hence, they can use precomputed distances. The *assign* and *update* operations can be performed independently of the time series and distance function. It is worth noting that this need for a pairwise distance matrix introduces memory overhead quadratic in train set size n , needing $O(n^2)$ distance function calls. The key algorithmic design component for k -medoids based clustering is how to choose the medoids and what objective function to use.

2.1 Alternate k -medoids

Given a crisp cluster label to each instance, the simplest approach to choose the medoid m_i for cluster C_i is to try all current members of the cluster, and choose the one minimising the within cluster distance. At any iteration, the medoid for each cluster is chosen independently based on currently assignment, as follows:

$$m_i = \arg \min_{x_m \in C} \sum_{x_c \in C} d(x_c, x_m). \quad (2)$$

where m_i is the i th medoid, C is a set of cases, x_m and x_c are time series in C , and d is a dissimilarity measure. This alternate k -medoids is the simplest form of medoids clustering. It is closely aligned with k -means (Lloyds [23]) and gets its name because of the alternating stages of the assignment and update. The main difference between alternate k -medoids and k -means is when calculating new cluster centres, k -means computes an average whereas alternate k -medoids finds medoids.

2.2 Partition Around Medoids (PAM)

Alternating k -medoids optimises the medoid within the current cluster assignment. This may miss the opportunity for taking medoids from other clusters and it may also converge prematurely since exemplars are less likely to change than with k -means [32]. PAM [19] is an alternative approach designed to overcome these problems. PAM follows a similar structure to alternate k -medoids but uses a different evaluation function to choose new medoids. It allows cases to become medoids of clusters they did not previously belong to, and when evaluating a new candidate medoid for any cluster, the total within cluster distance of all clusters is considered.

The original PAM used a bespoke initialisation function called *build*, which is similar to the restart methods used with k -means. However, for our experiments we use random initialisation with PAM. The reason for this is outlined in Section 4. The *swap* stage of PAM is described in Algorithm 1.

Algorithm 1 PAM *swap*: Iterative improvement, where X is a collection of time series, n is the number of cases in X , medoids is the current set of medoids, k is the number of medoids and d is a dissimilarity measure

```

1:  $init \leftarrow \text{findTD}(X, \text{medoids})$ 
2:  $best \leftarrow TD$ 
3:  $cm \leftarrow \text{medoids}$ 
4:  $\text{continue} \leftarrow \text{true}$ 
5: while  $\text{continue}$  do
6:   for  $i \leftarrow 1$  to  $k$  do
7:      $a \leftarrow cm_i, b \leftarrow best$ 
8:     for  $j \leftarrow 1$  to  $n$  do
9:       if  $x_j \notin cm$  then
10:         $cm_i \leftarrow x_j$ 
11:         $current \leftarrow \text{findTD}(X, cm)$ 
12:        if  $current < best$  then
13:           $best \leftarrow current$ 
14:       if  $best = b$  then
15:          $cm_i = a$ 
16:       if  $best = init$  then
17:          $\text{continue} \leftarrow \text{false}$ 
18: return  $best, cm$ 

```

Function *findTD* implements Equation 1. PAM uses a greedy algorithm that operates cluster by cluster (line 6). For each cluster, it tries all cases that are not currently medoids (lines 8-11) keeping the case that gives the lowest TD (lines 12-13). If there is no better candidate, the current medoids is retained (lines 14-15). The process terminates if the medoids have not changed (lines 16-17).

Finding the global optimum of the k -medoids problem is NP-hard [15], which is why PAM uses a greedy approximation. The algorithm requires a distance matrix ($O(n^2)$ memory) and each iteration has time complexity $O(kn^2)$. As this is both computationally and memory expensive many variations of PAM have been proposed to reduce memory, time complexity, or both.

2.3 PAM Variants

A range of refinements of the PAM algorithm have been proposed to improve PAM efficiency in both computational complexity and memory:

The Clustering LARge Applications (CLARA) [16] algorithm repeatedly applies PAM on a random subset of cases (with the recommended number being $s = 40 + 2k$). Once PAM is performed on the subset of cases and medoids obtained, the remaining cases are assigned to their closest medoid. This is repeated for multiple iterations and the iteration that has the lowest TD is returned. The time complexity is reduced to $O(k^3 + s)$.

CLARA based on raNdomised Search (CLARANS) [25] adapts the swap operation of PAM to use a more greedy approach. This is done by only performing the first swap which results in a reduction in TD before continuing evaluation. It limits the number of attempts known as max neighbours to randomly select and check if TD is reduced. This random selection gives CLARANS an advantage when handling large datasets by avoiding local minima.

PAM Silhouette (PAMSIL) [7] adapts the PAM algorithm to minimise the Silhouette score [29] rather than the TD.

PAM Medoid Silhouette (PAMMEDSIL)[7] is a variation on PAMSIL where Silhouette score is calculated by using the medoids rather than the arithmetic mean.

FasterPAM [32] focuses on optimising the PAM swap stage. It does this by combining optimisations made by FastPam1 [31] with a local hill-climbing approach that means any swap that reduces TD is immediately performed (eager swapping). However, while a swap is performed for any candidate that reduces TD, FasterPAM considers multiple candidates at a time in batches. The main reason for this is it allows the FasterPAM to be better parallelised. In addition FasterPAM uses the same technique for speed up that FastPam1 does by considering a swap across all medoids at once rather than a single medoid. This allows for expensive conditional logic to be moved outside the inner most loop further reducing computational time.

Faster Medoid Silhouette Clustering (FasterMSC) [18] is a variation on PAMMEDSIL that combines FasterPAM with PAMMEDSIL.

3 Elastic Distance Measures

Time series require bespoke distance functions because small offsets between series can lead to large distances between series that are conceptually similar. Elastic distances compensate for misalignment by creating a path through a cost matrix through either warping or editing time series. There have been many elastic distances proposed that attempt to align time series in different ways. We evaluate k -medoids with the nine elastic distance measures used in [22,12]. We provide a very brief overview of one of the nine elastic distances and direct the interested reader to these other publications [12,22,33]. The distance functions we use (with associated parameter setting) are listed in Table 1.

The best performing distance function according to [12] is MSM, which we briefly review below.

3.1 Move Split Merge (MSM)

At any step, elastic distances can use one of three costs: diagonal, horizontal or vertical, in forming an alignment. The alignment path is a series of moves across

Table 1. Summary of distance functions, their parameters and the default values.

Algorithm	Acronym	Parameters
Dynamic Time Warping	DTW	$w = 0.2$
Derivative DTW	DDTW	$w = 0.2$
Weighted DTW	WDTW	$g = 0.05$
Weighted derivative DTW	WDDTW	$g = 0.05$
Longest Common SubSequence	LCSS	$\epsilon = 0.05$
Edit distance with Real Penalty	ERP	$g = 0.05$
Edit Distance on Real sequences	EDR	$\epsilon = 0.05$
Move Split Merge	MSM	$c = 1$
Time Warp Edit	TWE	$\nu = 0.05, \lambda = 1$

the cost matrix. DTW assigns no explicit penalty for moving off the diagonal. Instead, it uses an implicit penalty (long paths have longer total distance) and a hard cut off on window size to stop large warpings. An alternative family of distance functions are based on the concept of edit distance. An edit distance considers a diagonal move as a match, a vertical move as an insertion and an horizontal move as a deletion. MSM [34] follows this structure, where move is a match (diagonal), split is a insertion (vertical) and merge is deletion (horizontal).

The move operation in MSM uses the absolute difference rather than the squared euclidean distance for matching in DTW. The cost of the split operation is given by cost function C (Equation 3) with a call to $C(a_i, a_{i-1}, b_j, c)$. If the value being inserted, b_j , is between the two values a_i and a_{i-1} being split, the cost is a constant value c . If not, the cost is c plus the minimum deviation from the furthest point a_i and the previous point a_{i-1} or b_j . The delete/merge is given by $C(b_j, b_{j-1}, a_i, c)$, which is simply the same operation as split but applied to the second series. Thus, the cost of splitting and merging values depends on the value itself and adjacent values.

$$C(x, y, z, c) = \begin{cases} c & \text{if } y \leq x \leq z \text{ or } y \geq x \geq z \\ c + \min(|x - y|, |x - z|) & \text{otherwise.} \end{cases} \quad (3)$$

Algorithm 2 describes how to calculate the MSM distance between two time series \mathbf{a} and \mathbf{b} . MSM satisfies triangular inequality and is a metric. In Algorithm 2, the first return value is the MSM distance between \mathbf{a} and \mathbf{b} , the second is the cost matrix used to compute the MSM distance.

4 Methodology

The different TSCL methods are compared using the whole set of 112 univariate, equal-length time series in the UCR archive [5]. Default train/test splits have been used, with data normalised to zero mean and unit standard deviation prior to the clustering stage. The training data is used to train an algorithm and the performance is assessed on the testing set. The number of clusters, k , is equal

Algorithm 2 MSM(**a** (of length m), **b** (of length m), **c** (minimum cost))

```

1: Let  $CM$  be an  $m \times m$  matrix initialised to zero.
2:  $CM_{1,1} = |a_1 - b_1|$ 
3: for  $i \leftarrow 2$  to  $m$  do
4:    $CM_{i,1} = CM_{i-1,1} + C(a_i, a_{i-1}, b_1, c)$ 
5: for  $i \leftarrow 2$  to  $m$  do
6:    $CM_{1,i} = CM_{1,i-1} + C(b_i, a_1, b + i - 1, c)$ 
7: for  $i \leftarrow 2$  to  $m$  do
8:   for  $j \leftarrow 2$  to  $m$  do
9:      $move \leftarrow CM_{i-1,j-1} + |a_i - b_j|$ 
10:     $split \leftarrow CM_{i-1,j} + C(a_i, a_{i-1}, b_j, c)$ 
11:     $merge \leftarrow CM_{i,j-1} + C(b_j, b_{j-1}, a_i, c)$ 
12:     $CM_{i,j} \leftarrow \min(move, split, merge)$ 
13: return  $CM_{m,m}, CM$ 

```

to the number of classes for classification. This choice is in line with the TSCL literature, such as [12,17].

The performance of the different clusterers is evaluated using the following measures: **CLustering ACCuracy (CL-ACC)** is the number of correct predictions divided by the total number of cases. For this, each cluster is assigned to its best matching class value by taking the maximum accuracy from every permutation of cluster and class value. The **Rand Index (RI)** measures the similarity between two sets of labels such as the predicted and actual class values. An improved version known as **Adjusted Rand Index (ARI)** avoids the inflation of the RI when dealing with a high number of clusters. For this, ARI adjusts the RI based on the expected scores on a purely random model. The **Mutual Information (MI)** score uses the entropy to measure the agreement of the two clusterings or a clustering and a true labelling. Finally, **Normalised Mutual Information (NMI)** rescales MI onto $[0, 1]$.

Some of the results are expressed using an adaptation of the critical difference diagram [6], replacing the post-hoc Nemenyi test with a comparison of all classifiers using pairwise Wilcoxon signed-rank tests, and cliques formed using the Holm correction [2,9].

Experiments are run with the open source python software packages `aeon`, `tslearn` [35], and `kmedoids` [30]. To enhance the reproducibility of this work, specific code and a guide to reproduce results will be available after blind review, as well as the results achieved.

The original PAM algorithm specifies a bespoke initialisation algorithm (*build*). However, we found random initialisation with ten restarts to be as effective as *build*, simpler and computationally less expensive. Given some PAM variants specify the use of random initialisation (e.g. FasterPAM) for speed, we use the same initialisation method for all algorithms in order to control factors of variation: we use random initialisation with ten restarts for all k -medoid and k -means variants.

The rationale behind using random initialisation is that random selection is likely to pick points from dense regions. The reason for rerunning the model multiple times with random initialisation and taking the best clusters (as measured by the sum of distances to their closest cluster centres) is that it reduces the chance results are skewed by poor random initial selections. Ten restarts is the most common number of restarts in the literature, and is the default value when using Lloyds algorithm in `scikit-learn`.

5 Results

The issue of which distance function is better overall is covered in depth in [12]. Our concern with these experiments is to detect differences between the two clustering algorithms over a range of distance functions. We focus first on the difference between alternate k -medoids and standard k -means in Section 5.1. We then evaluate a range of variants of the k -medoids algorithm in Sections 5.2 and 5.3. Finally, the best k -medoids variant is compared against several alternative TSCL approaches in Section 5.4.

5.1 Alternate k -medoids vs k -means

Table 2 summarises the difference in performance of k -means and k -medoids clustering algorithms. The mean difference is the average difference in the metric over 112 datasets on the test data. There is no significant difference in accuracy when using ED with the two clusterers (p value = 0.233 with a paired T-test or 0.14 with a binomial test). k -medoids gives a significantly more accurate clustering for all nine elastic clusterers (test with $\alpha=0.05$ with paired t-test, sign rank test and binomial test on wins/losses).

Table 2. Differences in CL-ACC, ARI and NMI between alternate k -medoids and k -means using 10 different distance functions. A positive value indicates that k -medoids is better. W/D/L figures are for CL-ACC.

Distance	CL-ACC	ARI	NMI	k -medoids wins	k -means wins	Ties
MSM	1.54%	1.22%	1.80%	59	47	6
TWE	2.78%	3.91%	3.65%	63	45	4
ERP	3.94%	4.35%	6.34%	66	33	13
WDTW	1.33%	1.94%	2.29%	66	42	7
DTW	3.88%	4.07%	5.72%	72	31	9
ED	-0.38%	-0.38%	-0.38%	46	58	8
DDTW	7.65%	6.17%	11.68%	75	32	5
DWDTW	2.57%	1.52%	3.56%	64	38	10
LCSS	4.13%	3.55%	7.33%	73	35	4
EDR	4.50%	4.37%	7.04%	74	35	3

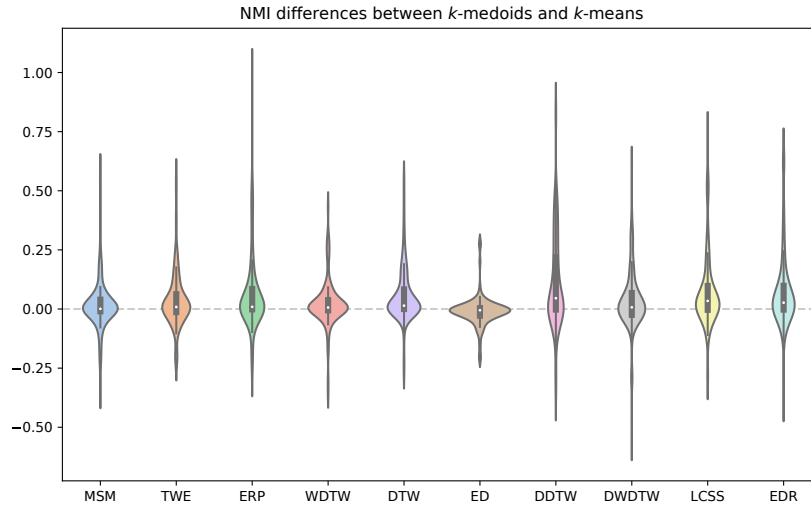


Fig. 1. Distributions of the differences between alternate k -medoids and k -means on the UCR data.

Figure 1 expands the data from Table 2 to show the distribution of differences for each distance measure. It shows a violin plot of the differences between alternate k -medoids and k -means for 10 distance functions in terms of NMI. It demonstrates that there is little difference when using ED. However, there is wide variation between k -medoids and k -means for the nine elastic distances, and the bulk of the distributions are positive.

These results indicate that, on average, the alternate k -medoids produces better clusters than k -means using the arithmetic mean to compute new centroids.

5.2 Alternate k -medoids vs PAM

The alternate technique used for the experiments in Section 5.1 is the simplest and easiest k -medoids algorithm. However, in standard clustering, PAM is a popular alternative and has found significantly better results than alternate k -medoids. As such we repeated the same experiments using the PAM algorithm described in Section 2.2 to see if the findings in standard clustering holds true for time series data. Table 3 summarises the differences between alternate k -medoids and PAM for each distance measure. With the exception of ERP, PAM significantly outperforms alternate k -medoids.

5.3 PAM variants

PAM significantly outperforms both k -means and alternate k -medoids. However, it is computationally more expensive. In Section 2.3, we describe several variants

Table 3. Differences in CL-ACC, ARI, NMI between alternate k -medoids and PAM using 10 different distance functions. A positive value indicates that PAM is better. W/D/L figures are for CL-ACC

Distance	CL-ACC	ARI	NMI	PAM wins	k -medoids wins	Ties
MSM	1.75%	1.57%	1.56%	39	22	51
TWE	2.15%	1.51%	1.69%	59	31	22
ERP	-2.34%	-3.95%	-3.95%	40	66	6
WDTW	1.92%	2.09%	2.15%	47	35	30
DTW	2.02%	1.62%	2.03%	57	36	19
DDTW	3.81%	4.41%	4.70%	67	38	7
DWDTW	3.26%	4.10%	3.58%	60	46	6
LCSS	0.31%	-0.60%	-0.22%	57	50	5
EDR	5.73%	5.91%	6.15%	76	32	4

of PAM meant to improve the runtime. Runtime complexity is a significant consideration when working with time series data. As such we compare six PAM variants to the original version when using MSM as a distance function. We include the following variants:

1. **clara** [16] and **clarans** [25]: subsampling techniques.
2. **fasterpam** performs eager swaps, improves time to find swaps
3. **pamsil** [7]: uses silhouette score rather than TD.
4. **pammedsil**, **fastermisc** [7]: use medoids silhouette score.

Figure 2 shows the average ranks of these six clusterers in terms of CL-ACC and NMI. PAM is significantly better than all variants except for fasterpam. Figure 3

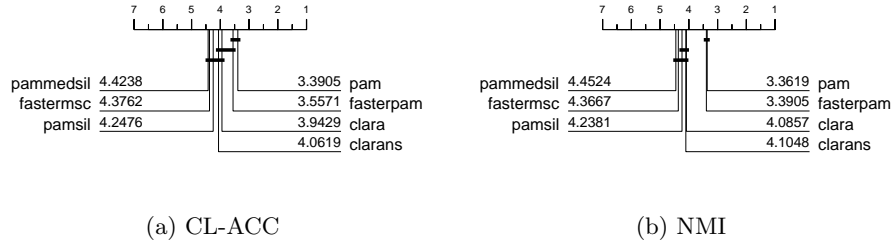


Fig. 2. Average ranks for PAM and six variants, all of which use MSM distance.

shows the distribution of the differences between PAM-MSM and the variants. As can be observed, for fasterpam most of the values are exactly 0, meaning that there is no difference to PAM for most of the datasets. Nevertheless, for the remaining five variants, boxplots generally are over the 0 value, indicating that PAM is better in average.

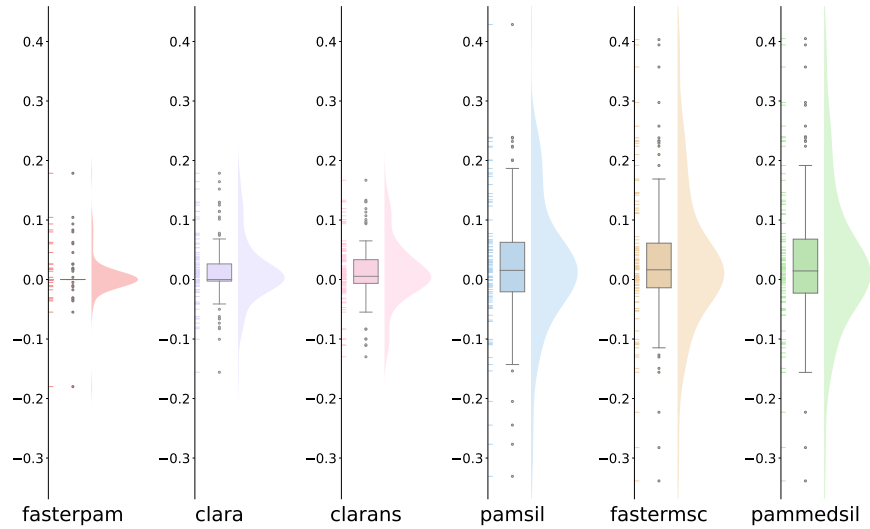


Fig. 3. Distributions of differences between PAM and the six variants. Positive values indicate PAM is better than the variant.

5.4 Elastic PAM vs alternative TSCL methods

We switch from considering relative performance of variants of the same algorithm to assess the absolute performance of PAM based clustering against popular TSCL alternatives. We compare performance of the following 10 clustering algorithms.

1. ***k*-means-DBA**: *k*-means clustering with DTW barycentre averaging and DTW distance assignment [27].
2. ***k*-means-ED**, ***k*-means-MSM** and ***k*-means-TWE**: *k*-means clustering with arithmetic mean for centres, and ED, MSM and TWE distance assignment, respectively.
3. ***k*-shapes** clustering [26].
4. Two-step Time series Clustering (**TTC**) [1].
5. **alternate-MSM** and **alternate-TWE**: alternate *k*-medoids clustering with MSM and TWE distances (Lloyds algorithm).
6. **PAM-MSM** and **PAM-TWE**: PAM *k*-medoids clustering with MSM and TWE distances.

Figure 4 shows the average ranks for three performance measures: CL-ACC, ARI and NMI. Note that PAM-MSM and PAM-TWE form a top clique and are significantly better than the other eight algorithms. Figure 5 summarises the relative performance using a heatmap tool described in [13].

Figure 4 shows TWE and MSM outperform the other elastic distances over a range of clustering metrics. [12] conducted a similar experiment over the same elastic distances for *k*-means and alternating *k*-medoids models and found similar

results. The reason TWE and MSM outperform other elastic distances was TWE and MSM constrain the diagonal warping with a constant cost penalty [12].

In addition from Figure 4 k -means is outperformed by both PAM and alternating k -medoids. The reason for this is during the averaging stage of k -means the average time series computation ignores alignment of the time series and thus a poor average (centre) is obtained [11].

Finally Figure 4 shows PAM across all our clustering metrics outperformed every other approach. The reason for this is as Section 2 outlines alternating only optimises the medoid from the current cluster assignment whereas PAM considers all instances as potential new medoid for a cluster which leads to better medoids being found.

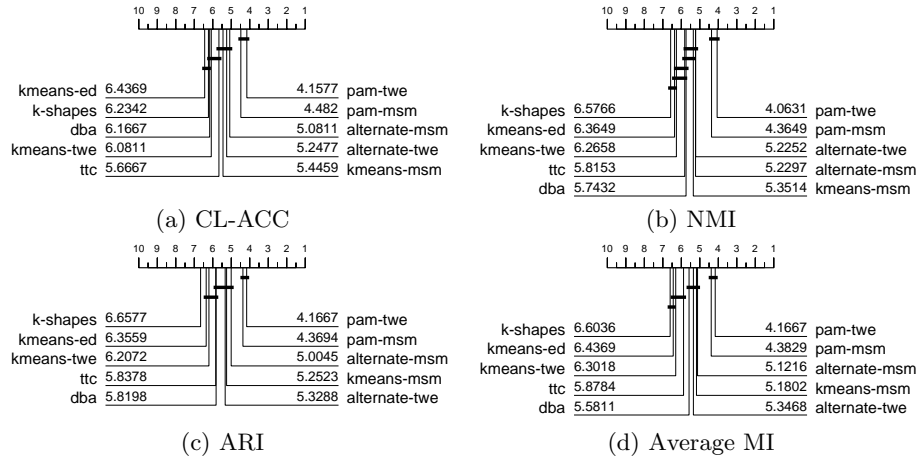


Fig. 4. Average ranks for 10 clustering algorithms using four performance measures.

Time series deep learning results are available the website associated with [17]. They provide NMI results for over 300 different clustering algorithms on the same UCR datasets we use. These are not directly comparable, since they are averaged over five runs and there may be other experimental differences. However, they can give some indication of relative performance. The best deep learning approach of the more than 300 assessed, a convolutional neural network with joint pretext loss and without clustering loss (key in their results is `res_cnn_joint_None`) achieved an average NMI of 0.3292. Average NMIs for PAM-TWE and PAM-MSM are 0.3366 and 0.3316, respectively.

In the context of other popular TSCL algorithms shown in Figure 4, k -medoids based approaches perform better than other approaches. In addition, Figure 4 highlights the strength of TWE and MSM across multiple approaches. Finally it is clear that using PAM based approaches with elastic distances yields significantly better results.

	pam-msm 0.5738	alternate-msm 0.5568	kmeans-msm 0.5406	dba 0.5396	ttc 0.5370	k-shapes 0.4781
Mean-Accuracy						
pam-msm 0.5738	Mean-Difference $r>c / r=c / r<c$ Wilcoxon p-value	0.0170 38 / 51 / 22 0.0149	0.0332 67 / 6 / 38 0.0003	0.0342 73 / 4 / 34 0.0009	0.0368 67 / 4 / 40 0.0029	0.0957 69 / 3 / 39 $\leq 1e-04$
alternate-msm 0.5568	-0.0170 22 / 51 / 38 0.0149	-	0.0162 59 / 6 / 46 0.0907	0.0173 70 / 6 / 35 0.0153	0.0198 63 / 4 / 44 0.0759	0.0787 61 / 4 / 46 0.0002
kmeans-msm 0.5406	-0.0332 38 / 6 / 67 0.0003	-0.0162 46 / 6 / 59 0.0907	-	0.0011 63 / 5 / 43 0.3578	0.0036 54 / 2 / 55 0.7262	0.0625 62 / 3 / 46 0.0012
dba 0.5396	-0.0342 34 / 4 / 73 0.0009	-0.0173 35 / 6 / 70 0.0153	-0.0011 43 / 5 / 63 0.3578	-	0.0026 53 / 8 / 50 0.8958	0.0614 59 / 2 / 50 0.0035
ttc 0.5370	-0.0368 40 / 4 / 67 0.0029	-0.0198 44 / 4 / 63 0.0759	-0.0036 55 / 2 / 54 0.7262	-0.0026 50 / 8 / 53 0.8958	-	0.0589 68 / 2 / 41 0.0010
k-shapes 0.4781	-0.0957 39 / 3 / 69 $\leq 1e-04$	-0.0787 46 / 4 / 61 0.0002	-0.0625 46 / 3 / 62 0.0012	-0.0614 50 / 2 / 59 0.0035	-0.0589 41 / 2 / 68 0.0010	If in bold, then p-value < 0.05

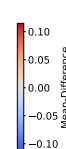


Fig. 5. Summary of performance measures for six of the clusterers described in Figure 4. Each cell shows the mean difference between algorithms, the W/D/L counts and the unadjusted p-value for a Wilcoxon sign-rank test.

6 Conclusions

Time Series Clustering (TSCL) with k -medoids has fallen out of favour in time series machine learning research in recent years. We demonstrate that k -medoids with elastic distance measures is highly effective, particularly PAM with TWE or MSM distances. We have also demonstrated that PAM is more effective than alternate k -medoids algorithm for most elastic distances and showed that PAM-TWE and PAM-MSM are significantly better than popular TSCL alternatives, and at least as good as the best known deep learning approach. Finally, we have explored the variants of PAM that hope to address the run time and memory complexity the traditional PAM algorithm suffered from. We found that the recent FasterPAM [32] yields very similar results as PAM but achieves an $O(k)$ -fold speedup in the swap phase, making FasterPAM a much more attractive alternative for TSCL.

In the future, we would like to further quantify the run time complexity of these clusterers. We will then investigate the possibility of creating an ensemble of elastic distances using k -medoids clusterer, similar to the elastic ensemble classifier proposed in [22]. Furthermore, we would like to perform a similar experiment with other clustering algorithms to if using TWE and MSM yields significantly better results than traditional euclidean and DTW distances.

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) (grant ref.: EP/W030756/1), and by “Agencia Española de Investigación (España)” (grant ref.: PID2020-115454GB-C22 / AEI / 10.13039 / 501100011033). David Guijo-Rubio’s research has been subsidised by the University of Córdoba and financed by the European Union - NextGenerationEU (grant ref.: UCOR01MS). The experiments were carried out on the High Performance Computing Cluster supported by the Research and Specialist Computing

Support service at the University of East Anglia. We would like to thank all those responsible for helping maintain the time series dataset archives and those contributing to open source implementations of the algorithms.

References

1. S. Aghabozorgi and T. Y. Wah. Clustering of large time series datasets. *Intelligent Data Analysis*, 18:793–817, 2014.
2. A. Benavoli, G. Corani, and F. Mangili. Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17:1–10, 2016.
3. B. Cai, G. Huang, N. Samadiani, G. Li, and C.-H. Chi. Efficient time series clustering by minimizing dynamic time warping utilization. *IEEE Access*, 9:46589–46599, 2021.
4. J. Caiado, E. Maharaj, and P. D’Urso. Time series clustering. In *Handbook of Cluster Analysis*, pages 241–264, 2015.
5. H. Dau, A. Bagnall, K. Kamgar, M. Yeh, Y. Zhu, S. Gharghabi, C. Ratanamahatana, A. Chotirat, and E. Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
6. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
7. M. V. der Laan, K. Pollard, and J. Bryan. A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584, 2003.
8. V. Estivill-Castro. Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75, jun 2002.
9. S. García and F. Herrera. An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.
10. T. Germain, C. Truong, L. Oudre, and E. Krejci. Unsupervised study of plethysmography signals through dtw clustering. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 3396–3400. IEEE, 2022.
11. C. Holder, D. Guijo-Rubio, and A. Bagnall. Barycentre averaging for the move-split-merge time series distance measure. *15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2023.
12. C. Holder, M. Middlehurst, and A. Bagnall. A review and evaluation of elastic distance functions for time series clustering. *Knowledge and Information Systems*, 2023.
13. A. Ismail-Fawaz, A. Dempster, C. W. Tan, M. Herrmann, L. Miller, D. F. Schmidt, S. Berretti, J. Weber, M. Devanne, G. Forestier, et al. An approach to multiple comparison benchmark evaluations that is stable under manipulation of the compare set. *arXiv preprint arXiv:2305.11921*, 2023.
14. A. Javed, B. S. Lee, and D. Rizzo. A benchmark study on time series clustering. *Machine Learning with Applications*, 1, 2020.
15. O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. ii: The p -medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.
16. L. Kaufman and P. J. Rousseeuw. Clustering large data sets. In *Pattern Recognition in Practice*, pages 425–437. Elsevier, Amsterdam, 1986.
17. B. Lafabregue, J. Weber, P. Gancarski, and G. Forestier. End-to-end deep representation learning for time series clustering: a comparative study. *Data Mining and Knowledge Discovery*, 36:29–81, 2022.

18. L. Lenssen and E. Schubert. Clustering by direct optimization of the medoid silhouette. In *Similarity Search and Applications: 15th International Conference, SISAP 2022, Bologna, Italy, October 5–7, 2022, Proceedings*, pages 190–204. Springer, 2022.
19. P. J. R. Leonard Kaufman. *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley and Sons, Ltd, 1990.
20. H. Li, J. Liu, Z. Yang, R. W. Liu, K. Wu, and Y. Wan. Adaptively constrained dynamic time warping for time series classification and clustering. *Information Sciences*, 534:97–116, 2020.
21. X. Li, J. Lin, and L. Zhao. Time series clustering in linear time complexity. *Data Mining and Knowledge Discovery*, 35(3):2369–2388, 2021.
22. J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29:565–592, 2015.
23. S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.
24. P. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):306–318, 2009.
25. R. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14:1003–1016, 10 2002.
26. J. Paparrizos and L. Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870, 2015.
27. F. Petitjean, A. Ketterlin, and P. Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44:678–, 03 2011.
28. C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *proceedings of the 5th SIAM International Conference on Data Mining*, 2005.
29. P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
30. E. Schubert and L. Lenssen. Fast k-medoids clustering in rust and python. *Journal of Open Source Software*, 7(75):4183, 2022.
31. E. Schubert and P. J. Rousseeuw. Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms. In *Similarity Search and Applications*, pages 171–187, Cham, 2019. Springer International Publishing.
32. E. Schubert and P. J. Rousseeuw. Fast and eager k-medoids clustering: O(k) runtime improvement of the pam, clara, and clarans algorithms. *Information Systems*, 101:101804, 2021.
33. A. Shifaz, C. Pelletier, F. Petitjean, and G. Webb. Elastic similarity and distance measures for multivariate time series. *Knowledge and Information Systems*, 65(6), 2023.
34. A. Stefan, V. Athitsos, and G. Das. The Move-Split-Merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2013.
35. R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods. Tsllearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.