

TDK-dolgozat

Marcell Balázs Tóth
2024.



Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Artificial Intelligence

Efficient Clustering of Time Series Segments and Robustness Analysis of the Results

Scientific Students' Association Report

Author:

Marcell Balázs Tóth
5th grade student

Advisors:

dr. Gábor Szűcs,
Associate professor

Marcell Németh,
PhD student

2024.

Contents

1	Introduction	2
2	Theoretical overview	4
2.1	Kolmogorov-Smirnov test	4
2.2	Dynamic Time Warping (DTW)	4
2.3	COP-KMeans	5
2.4	Key Metrics for Clustering Evaluation	5
2.5	Basepoints in Machine Learning	6
2.6	Determining Optimal Clusters: Elbow vs. Silhouette Methods	6
3	Proposed Clustering Method With Full Embedding	7
3.1	Problem Formulation	7
3.2	Embedding Construction Using Kolmogorov-Smirnov Test	7
3.3	Clustering with COP-KMeans	8
4	Reduced-Embedding Based Clustering Method With Basepoints	9
4.1	Motivation for Basepoint Selection	9
4.2	Proposed Basepoint Selection Strategy	9
4.3	Complexity of the method	10
4.4	Confidence score calculation	11
4.5	Pseudo-code Implementation	11
5	Experimental Results	13
5.1	Dataset Description	13
5.2	Evaluation Metrics	14
5.3	Results of the Clustering Performance	14
5.3.1	Clustering Performance with Ground Truth Number of Clusters . .	14
5.3.2	Cluster Number Estimation Using Elbow and Silhouette Methods .	17
5.3.3	Effectiveness of Confidence Score in Enhancing Clustering Performance	17

5.4	Sensitivity to KS-test	18
5.5	Runtime Analysis	19
5.6	Comparison with TimeSeriesKMeans	21
5.6.1	Overview of TimeSeriesKMeans	22
5.6.2	Methodological Differences	22
5.6.3	Complexity Comparison	23
5.6.4	Empirical Performance Comparison	23
6	Robustness Analysis with Noise	25
6.1	Noise Injection Strategy	25
6.2	Results	26
6.3	Confidence Score Analysis	27
6.3.1	Confidence Score Distribution with Increasing Noise	27
6.3.2	Impact of Confidence Threshold on Clustering Metrics	28
6.4	Interpretation of Results	29
7	Conclusion	31
	Bibliography	33

Chapter 1

Introduction

Time series clustering is a crucial task in machine learning with applications across various domains, including finance, healthcare, and industrial monitoring. It involves grouping similar sequences of data points that evolve over time. However, this process poses challenges due to temporal dependencies, varying segment lengths, and misalignment of time points. Traditional methods, such as whole time series and subsequence clustering [1, 30, 3, 7], either struggle with large datasets or fail to account for such complexities.

Subsequence clustering, which partitions a long time series into segments for clustering, is particularly relevant for identifying recurring patterns in complex datasets. However, existing approaches often lack the robustness to handle temporal misalignment and variability in segment length. Additionally, in industrial applications, where datasets are large, the computational inefficiency of traditional clustering algorithms is a significant limitation.

In this paper, we propose a novel clustering framework that integrates the Kolmogorov-Smirnov (KS) test [11] for statistical comparison of segments to address temporal misalignment. To further enhance clustering, we apply COP-KMeans [28], incorporating pairwise constraints to ensure that consecutive time series segments are not assigned to the same cluster, thereby preserving their temporal independence. Our method effectively handles large datasets while maintaining accuracy.

The key contributions of this paper are:

- A new subsequence clustering method combining KS test and adapted constrained clustering algorithm (COP-KMeans) for robust and flexible time series segment clustering.
- A reduced-embedding-based clustering method with base point selection to decrease computational complexity.
- Comprehensive experimental results demonstrating the method’s efficiency and accuracy on diverse datasets.
- A confidence score to quantify the reliability of cluster assignments, where higher accuracy achieved by filtering low-confidence segments.

The remainder of this paper is organized as follows: Chapter 2 provides background knowledge, Chapter 3 describes the proposed clustering method using the full embedding, Chapter 4 details the reduced-embedding-based clustering method with basepoints, and

presents the pseudo-code implementation. Chapter 5 presents the experimental results, followed by Chapter 6 on robustness analysis with noise. Finally, in Chapter 7, we draw the conclusions.

Chapter 2

Theoretical overview

In this section, we explore the key concepts, algorithms, and methodologies that formed the foundation of our research. A crucial aspect of analyzing time series data is comparing segments to determine their degree of similarity or dissimilarity. This requires the application of statistical tests and distance measures that can capture variations across time-dependent data.

2.1 Kolmogorov-Smirnov test

The Kolmogorov-Smirnov (K-S) [11] test compares the distributions of two samples to determine if they come from the same distribution. It calculates the maximum difference between their empirical cumulative distribution functions (ECDFs):

$$F_X(x) = \frac{1}{n_1} \sum_{i=1}^{n_1} \mathbf{1}_{[x_i \leq x]}, \quad F_Y(y) = \frac{1}{n_2} \sum_{i=1}^{n_2} \mathbf{1}_{[y_i \leq y]} \quad (2.1)$$

The K-S statistic D is the maximum absolute difference between the ECDFs:

$$D = \sup |F_X(x) - F_Y(x)| \quad (2.2)$$

The p-value measures the probability of observing D under the null hypothesis. The complexity of the K-S test is $O(n \log n)$, where $n = n_1 + n_2$.

2.2 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) measures the similarity between two temporal sequences by aligning them optimally, often used in time series analysis [17, 18]. Given two sequences A and B , a cost matrix $C(i, j)$ is constructed with the distance between elements $A[i]$ and $B[j]$:

$$C(i, j) = |A[i] - B[j]| \quad (2.3)$$

The accumulated cost matrix $D(i, j)$ is then computed using dynamic programming:

$$D(i, j) = C(i, j) + \min(D(i-1, j), D(i, j-1), D(i-1, j-1)) \quad (2.4)$$

The DTW distance is the value $D(n, m)$, where n and m are the sequence lengths [20]. The complexity is $O(n \cdot m)$ [22].

2.3 COP-KMeans

COP-KMeans [28] extends K-means by incorporating pairwise constraints:

- Must-link: Points must be in the same cluster.
- Cannot-link: Points cannot be in the same cluster.

During assignment, points are assigned to the nearest centroid while respecting these constraints. The complexity is similar to K-means, $O(t \cdot n \cdot k)$, but additional constraint-checking steps increase the computational cost. COP-KMeans is effective when prior relationships between data points are known.

2.4 Key Metrics for Clustering Evaluation

Several metrics are used to evaluate clustering performance, especially when multiple labels are present. Key metrics include the Rand Index (RI) [14], Adjusted Rand Index (ARI) [14], and Normalized Mutual Information (NMI) [27].

Rand Index (RI) measures the proportion of correctly classified pairs of samples. It ranges from 0 to 1, where 1 indicates perfect agreement.

$$RI = \frac{a + d}{a + b + c + d} \quad (2.5)$$

where:

- a : Number of pairs correctly clustered together
- b : Number of pairs incorrectly clustered together
- c : Number of pairs incorrectly clustered apart
- d : Number of pairs correctly clustered apart

Adjusted Rand Index (ARI) adjusts the RI by accounting for chance. It ranges from -1 to 1, with 1 indicating perfect agreement.

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]} \quad (2.6)$$

where $E[RI]$ is the expected RI under random clustering.

Normalized Mutual Information (NMI) measures the mutual information between two clusterings, normalized to account for different cluster sizes. It ranges from 0 to 1, where 1 indicates perfect correlation.

$$NMI(U, V) = \frac{2 \cdot I(U; V)}{H(U) + H(V)} \quad (2.7)$$

where $I(U; V)$ is the mutual information, and $H(U)$ and $H(V)$ are the entropies of the clusterings U and V .

These metrics provide insights into clustering performance, with ARI and NMI particularly useful for comparing results against ground truth or when cluster sizes vary.

2.5 Basepoints in Machine Learning

Basepoints are essential in machine learning, acting as reference vectors for comparison against data point embeddings. These vector embeddings are numerical representations in high-dimensional space, capturing the semantic meaning and relationships within the data [15]. The primary function of basepoints is to measure the distance to other vectors, utilizing metrics such as Euclidean distance, which computes the straight-line distance between points, and cosine similarity, which assesses directional similarity [23].

Basepoints find applications in various domains, including anomaly detection, where new data points are compared against established baselines to identify outliers, and classification tasks, where they represent class centroids, aiding in categorizing new inputs based on proximity [9]. The general workflow involves creating embeddings from data, establishing basepoints from training data, comparing new vectors to these basepoints, and making decisions based on the results.

In conclusion, basepoints are pivotal in vector-based machine learning systems, providing the necessary framework for tasks such as classification, clustering, and anomaly detection, thereby enhancing the predictive capabilities of models operating in high-dimensional spaces [2].

2.6 Determining Optimal Clusters: Elbow vs. Silhouette Methods

The Elbow Method and the Silhouette Method are popular techniques for determining the optimal number of clusters in K-means clustering. The Elbow Method involves plotting the explained variance (or within-cluster sum of squares, WCSS) against the number of clusters K to identify an "elbow" point, where adding more clusters yields diminishing returns [26]. This method is simple and visually intuitive, though it can be ambiguous if the elbow is not distinctly visible.

Conversely, the Silhouette Method quantitatively evaluates clustering quality by calculating a silhouette coefficient for each data point. This coefficient measures how well a point fits into its cluster compared to others, with values ranging from -1 to +1 [21]. The best K is determined by identifying the value that maximizes the average silhouette coefficient, indicating well-defined clusters.

While both methods serve to estimate the optimal number of clusters, they complement each other. The Elbow Method offers a straightforward initial estimate, while the Silhouette Method provides a rigorous assessment of cluster quality, making their combined use beneficial for more informed decision-making [10].

Chapter 3

Proposed Clustering Method With Full Embedding

In this section, we present a novel clustering method designed for time-varying segments characterized by inhomogeneous lengths and temporal misalignment. The proposed approach integrates a robust feature extraction process using the Kolmogorov-Smirnov (KS) test [11] to quantify statistical differences between segment distributions, followed by embedding construction and clustering using the Constrained K-means (COP-KMeans) algorithm [28]. Our approach ensures that consecutive segments are assigned to distinct clusters, respecting their temporal dependencies.

3.1 Problem Formulation

Clustering time-varying segments introduces significant challenges due to two key factors:

- **Temporal Misalignment:** Segments may exhibit significant shifts along the time axis, making direct comparison difficult [8].
- **Inhomogeneous Lengths:** The segments may have varying lengths, further complicating distance-based clustering methods [13].

The core of our proposed method addresses these challenges by extracting features that capture distributional differences, independent of temporal alignment or length variation. Given a set of time-varying segments $\{x_1, x_2, \dots, x_n\}$, each segment x_i has an associated length l_i . Our goal is to cluster these segments into K distinct clusters by constructing meaningful embeddings for each segment based on distributional similarity.

3.2 Embedding Construction Using Kolmogorov-Smirnov Test

To generate robust embeddings, we begin by quantifying the statistical differences between each pair of segments. For this, we employ the Kolmogorov-Smirnov (KS) test [11], a non-parametric test that compares two sample distributions. The KS test statistic for two segments x_a and x_b , with lengths l_a and l_b , is defined as:

$$D_{KS}(x_a, x_b) = \sup |F_a(t) - F_b(t)| \quad (3.1)$$

where $F_a(t)$ and $F_b(t)$ represent the empirical cumulative distribution functions (ECDFs) of segments x_a and x_b , respectively. The KS statistic $D_{KS}(x_a, x_b)$ measures the maximum difference between the two ECDFs, offering a robust means of comparing the statistical distributions of the segments, irrespective of their length or alignment [19].

Once the KS test has been computed for all pairs of segments, we construct an embedding vector x_i for each segment x_i as follows:

$$x_i = [D_{KS}(x_1, x_i), D_{KS}(x_2, x_i), \dots, D_{KS}(x_n, x_i)] \quad (3.2)$$

This embedding captures the statistical relationship of segment x_i with all other segments in the dataset. The dimensionality of each embedding is n , the total number of segments. This process transforms the problem from clustering raw time series data to clustering embeddings that represent pairwise distributional differences.

3.3 Clustering with COP-KMeans

Although traditional clustering algorithms such as KMeans can be applied to these embeddings, they do not account for the temporal structure of the data, where consecutive segments are known to belong to distinct clusters [1]. To incorporate this domain-specific knowledge, we utilize the COP-KMeans algorithm, which introduces must-link and cannot-link constraints during the clustering process [28]. In our case, the cannot-link constraints enforce that temporally adjacent segments are assigned to different clusters, reflecting the inherent sequential structure of the data. By integrating these constraints, COP-KMeans produces a clustering that respects both the statistical differences between segments and their temporal order. The resulting labels correspond to the final clusters, with each segment assigned to its respective group based on the learned embedding space and the applied constraints.

Chapter 4

Reduced-Embedding Based Clustering Method With Basepoints

In this section, we introduce an updated approach that addresses the computational challenges associated with embedding vector construction. Specifically, the dimensionality of the embedding vectors grows quadratically with the number of segments. This quadratic growth can lead to the inclusion of redundant information beyond a certain threshold of segments. To mitigate this issue, we propose a method that replaces exhaustive pairwise comparisons of all segments with a selective approach that utilizes a predefined set of n basepoints.

4.1 Motivation for Basepoint Selection

The motivation behind the selection of basepoints lies in the desire to enhance the efficiency of the embedding vector generation process while maintaining the integrity of the clustering representation. Instead of comparing each segment with every other segment, we limit our comparisons to these b basepoints (that is, the length of the embedding vector will be reduced to b). This strategic selection significantly reduces the complexity of the algorithm from $O(n^2)$ to $O(n \cdot b)$, where b is substantially smaller than n . By focusing on representative basepoints, we can capture the essential characteristics of the dataset without incurring the computational burden associated with a full pairwise comparison, thus improving the overall efficiency of our method.

4.2 Proposed Basepoint Selection Strategy

The basepoint selection process begins by randomly selecting one segment to serve as the first basepoint. The embedding vector's initial value is then calculated for each segment based on this basepoint. Subsequent basepoints are selected iteratively by maximizing the Euclidean distance from the previously selected basepoints. This ensures that each new basepoint adds diversity and enhances the representation of the data [24].

The steps of the basepoint selection process are as follows:

1. Initialization: Select the first basepoint $B[g]$ randomly from the set of segments $X = \{x_1, x_2, \dots, x_n\}$, where $g=1$.
2. Intermediate Embedding Vector: For each segment $x_i \in X$, compute the embedding vector's first value, $v_i[g] = f(x_{B[g]}, x_i)$, where $f(x_a, x_b)$ represents the distance metric (i.e. KS statistic) and $g=1$.
3. Iterative Selection: For each subsequent basepoint $B[j]$ ($j=2,3,\dots,b$), calculate the Euclidean distance between the current basepoints and all remaining segments. Select the segment that maximizes the minimum distance from the current basepoints:

$$B[j] = \arg \max_{x_i} \min_{x_{B[1]}, \dots, x_{B[j-1]}} \|x_i - x_{B[k]}\| \quad (4.1)$$

This iterative selection continues until b basepoints are chosen.

4. Embedding Vector Construction: For each segment $x_i \in X$, compute the embedding vector based on the selected basepoints:

$$x_i = [f(x_{B[1]}, x_i), f(x_{B[2]}, x_i), \dots, f(x_{B[b]}, x_i)] \quad (4.2)$$

Since the distance of all segments to the current basepoints is calculated after each basepoint selection, by the time we reach the end of the basepoint selection process, we already possess the distances of all segments to the respective basepoints. Consequently, no further calculations are needed to construct the embedding vectors.

By strategically selecting basepoints in this manner, we ensure that each new basepoint captures diverse characteristics of the dataset, thereby enhancing the robustness and representativeness of the embedding vectors while significantly reducing computational complexity.

4.3 Complexity of the method

The time complexity of our algorithm can be analyzed in terms of several key parameters:

- b : the number of basepoints selected,
- n : the total number of segments,
- k : the number of clusters formed,
- L : the sum of the number of samples in the two segments being compared.

The overall time complexity of the proposed method can be analyzed as follows:

- KS Test Cost: The computational cost of performing the Kolmogorov-Smirnov test between two segments of combined length L is $O(L \log L)$. Since this test is performed between the basepoints and all segments, the total cost for this step is $O(b \cdot n \cdot L \log L)$, where b is the number of basepoints and n is the number of segments [11].
- Euclidean Distance Calculation: The cost of calculating the Euclidean distance for each basepoint is $O(b)$, where b is the number of dimensions in the embedding vector [17].

- **Embedding Vector Construction:** The cost of constructing the embedding vector is:

$$O(b \cdot (n \cdot L \log L + b \cdot n \cdot b)) = O(b \cdot n \cdot L \log L + b^3 \cdot n) \quad (4.3)$$

This accounts for the KS test comparisons and the embedding vector construction based on basepoints.

- **Clustering Step:** The complexity of the COP-KMeans clustering algorithm is:

$$O(n \cdot b \cdot k \cdot I) \quad (4.4)$$

where k is the number of clusters, and I is the number of iterations required for convergence [28].

Thus, the total complexity of the method is:

$$O(b \cdot n \cdot L \log L + b^3 \cdot n + n \cdot b \cdot k \cdot I) \quad (4.5)$$

This formulation provides a complete understanding of the computational demands of our method, highlighting the influence of basepoints on reducing complexity while maintaining efficiency in clustering segments varying in time.

4.4 Confidence score calculation

To further enhance the interpretability of the clustering results, we introduce a confidence score for each segment's assigned label. The confidence score is based on the relative distances between the segment and its two closest cluster centers.

The confidence score $C(x_i)$ for segment x_i is defined as follows:

$$C(x_i) = \begin{cases} 1 & \text{if } d_2(x_i) > 2 \cdot d_1(x_i) \\ \frac{d_2(x_i) - d_1(x_i)}{d_1(x_i)} & \text{otherwise} \end{cases} \quad (4.6)$$

where $d_1(x_i)$ is the distance between x_i and the closest cluster center, and $d_2(x_i)$ is the distance to the second closest cluster center. The confidence score ranges between 0 and 1:

- $C(x_i) = 1$ if the second closest cluster center is at least twice as far from the closest one, indicating high confidence.
- $C(x_i) = 0$ if the second closest center is as close as the closest, indicating low confidence.

This method provides an effective way to quantify the reliability of clustering labels, offering insights into ambiguous or uncertain assignments. While similar to the Silhouette Index, which measures intra-cluster density, this approach quantifies reliability based on the proximity to the nearest centroids.

4.5 Pseudo-code Implementation

We provide the pseudo-code implementation of the proposed method in the Algorithm 1:

Data: Segments $X = \{x_1, x_2, \dots, x_n\}$, Number of basepoints b , Number of clusters k , Distance function $f(x_a, x_b)$ (i.e. KS-test)

Result: Cluster labels $L = \{l_1, l_2, \dots, l_n\}$

Initialize empty array B to store basepoints;

Select the first basepoint as the first element: $B[1] = 1$;

for *each segment* x_i *in* X **do**

 | Calculate the embedding vector's first value: $v_i[1] = f(x_{B[1]}, x_i)$;

end

for *basepoint index* $j = 2$ *to* b **do**

 | Find the next basepoint;

for *each segment* x_i *in* X *(including already selected basepoints)* **do**

 | Calculate Euclidean distance from the vectors of the current basepoints $\{x_{B[1]}, \dots, x_{B[j-1]}\}$;

end

 | Select the segment with the maximum distance as the new basepoint:

$B[j] = \arg \max d(x_{B[j-1]}, x_i)$;

end

for *each segment* x_i *in* X **do**

 | Update the embedding vector: $v_i[j] = f(x_{B[j]}, x_i)$;

end

Perform adapted COP-KMeans clustering on the embedding vectors

$V = \{v_1, v_2, \dots, v_n\}$;

Use constraints to ensure consecutive segments are assigned to different clusters;

return *Cluster labels* $L = \{l_1, l_2, \dots, l_n\}$

Algorithm 1: Basepoint Selection and Clustering with Adapted COP-KMeans

Chapter 5

Experimental Results

In this chapter, we present a comprehensive evaluation of our proposed clustering method. We begin by detailing the datasets used in the experiments, covering their characteristics and structure. Next, we outline the evaluation metrics employed to assess clustering performance, with a focus on metrics such as Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI).

Following this, we present the results of our method’s clustering performance on the selected dataset, analyzing the accuracy and consistency of the method across multiple runs. We also examine the sensitivity of the clustering results to the Kolmogorov–Smirnov test, providing insights into how variations in the KS-test affect clustering outcomes [11].

Subsequently, we offer a comparative analysis between our approach and the TimeSeriesKMeans algorithm [5] from the tslearn library [25]. This comparison includes an overview of the TimeSeriesKMeans method, a discussion of key methodological differences, and an analysis of computational complexities. Additionally, we conduct an empirical performance comparison, emphasizing clustering accuracy and runtime efficiency.

This evaluation illustrates the effectiveness and advantages of our proposed method, particularly in comparison to TimeSeriesKMeans, across a range of metrics and scenarios.

5.1 Dataset Description

For experimental evaluation, we selected ten diverse datasets from the Time Series Segmentation Benchmark (TSSB) [6]. These datasets vary in domain, complexity, and length, providing a robust basis for testing our clustering method. The datasets include:

- Plane (PLN): Time series representing plane shapes, often used for shape classification.
- NonInvasiveFetalECGThorax1 (NIFECG): Fetal ECG signals with physiological noise.
- UWaveGestureLibraryX/All (UWGLX/All): Accelerometer data capturing hand gestures, with increasing complexity across dimensions.
- EOGHorizontalSignal (EOGHS): Noisy eye movement data from electrooculography (EOG).
- ProximalPhalanxTW (PPTW): Hand bone motion data.

- SwedishLeaf (SLF): Leaf outlines used for shape classification.
- Symbols (SYM): Symbolic shape representations.
- Car (CAR): Car dynamics data capturing driving behaviors.
- InlineSkate (INSK): Data from inline skating, exhibiting non-linear motion patterns.

We generated larger datasets of 1,000 and 10,000 segments by randomly extracting 30–80% portions of the original time series, introducing segment length variability. Clustering performance was evaluated using Rand Index (RI), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI), averaged over 10 runs. Additionally, we assessed the impact of confidence-based label removal on clustering quality by recalculating metrics after discarding low-confidence labels.

5.2 Evaluation Metrics

For evaluating the performance of our clustering algorithm, we use three well-established metrics: Rand Index (RI), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI) [14, 27]. To ensure the robustness of the results, the clustering process is repeated 10 times, and the average scores for each metric, along with their standard deviations, are computed.

In addition to the overall performance of the clustering algorithm, we assess the effectiveness of our confidence-based approach. To do this, we remove labels that have a confidence score below a predefined threshold and then recalculate the evaluation metrics. The underlying hypothesis is that labels with low confidence scores are more likely to be incorrectly assigned, and removing them could improve the quality of the clustering. If the recalculated metrics (RI, ARI, and NMI) show improvement after removing these labels, it indicates that the confidence score successfully identifies unreliable labels.

This approach allows us to quantify the impact of low-confidence labels on clustering performance and assess the reliability of our confidence estimation mechanism.

5.3 Results of the Clustering Performance

In this section, we present the evaluation metrics obtained when the clustering is performed using the ground truth number of clusters. To ensure consistency and reliability, the COP-KMeans algorithm is executed ten times, with the average scores calculated across all runs. Following this, we evaluate how accurately the elbow method and silhouette analysis estimate the optimal number of clusters. Subsequently, we evaluated the effectiveness of the confidence score in improving the clustering results.

5.3.1 Clustering Performance with Ground Truth Number of Clusters

The following table summarizes the evaluation metrics for each dataset, consisting of 1,000 segments, using the ground truth number of clusters. The results presented are the averages across ten runs of the COP-KMeans algorithm with fifty basepoints. This configuration takes into account the order of the segments while calculating the clusters,

aiming to maintain accurate clustering performance even with this reduced comparison strategy.

Table 5.1 shows the performance metrics when using the basepoint approach with COP-KMeans. The basepoints act as reference segments, reducing the computational cost by limiting direct comparisons to only a subset of segments while ensuring the time-order constraints are considered.

Dataset	RI (mean/std)	ARI (mean/std)	NMI (mean/std)
PLN	0.987/0.026	0.950/0.100	0.970/0.060
NIFECG	0.947/0.044	0.818/0.148	0.872/0.094
UWGLX	0.901/0.042	0.704/0.119	0.788/0.077
UWGLAll	0.983/0.001	0.948/0.003	0.941/0.003
EOGHS	0.680/0.020	0.202/0.047	0.259/0.028
PPTW	0.994/0.000	0.984/0.000	0.976/0.000
SLF	1.000/0.000	1.000/0.000	1.000/0.000
SYM	0.920/0.002	0.786/0.005	0.736/0.005
CAR	0.977/0.000	0.940/0.000	0.925/0.000
INSK	0.864/0.000	0.695/0.000	0.654/0.000
Average	0.925/0.014	0.803/0.042	0.812/0.027

Table 5.1: Adapted COP-KMeans with 50 Basepoints

After evaluating COP-KMeans with basepoints, we perform a similar experiment without the use of basepoints, where each segment is compared to all other segments. This configuration increases the computation but also provides insight into how the method performs without basepoint sampling.

In Table 5.2, the results show the performance when all segments are compared to each other, removing the basepoint constraint. This configuration is expected to yield higher accuracy because of the direct comparison of each segment, but this comes at the cost of significantly increased computation time. The last three columns present that the goodness indicators decrease only slightly with the use of basepoints, dropping to only 97%, 89% and 91% respectively compared to the reference.

Dataset	COP w/o Basepoints			With and w/o BP ratio (%)		
	RI	ARI	NMI	RI	ARI	NMI
PLN	0.987	0.952	0.971	100	99.8	99.9
NIFECG	0.981	0.932	0.945	96.5	87.8	92.3
UWGLX	0.974	0.925	0.931	92.5	76.1	84.6
UWGLAll	0.985	0.952	0.942	99.9	99.5	99.5
EOGHS	0.761	0.381	0.432	89.4	53	60
PPTW	0.996	0.989	0.984	99.5	99.9	99.2
SLF	1.000	1.000	1.000	100	100	100
SYM	0.932	0.818	0.771	98.7	96.1	95.5
CAR	0.966	0.909	0.887	101.1	103.4	104.3
INSK	0.956	0.9	0.859	94.0	77.2	76.1
Average	0.954	0.876	0.872	96.8	89.3	91.2

Table 5.2: Clustering performance comparison of adapted COP-KMeans without basepoints and the ratio of performance with and without basepoints.

We also explore the impact of removing the constraints of COP-KMeans and using a standard KMeans approach, both with and without basepoints. These configurations allow us to assess how much the constraint of temporal ordering influences the results.

In Table 5.3, the standard KMeans algorithm is used with 50 basepoints. This allows us to analyze the impact of time-order constraints, comparing the performance against COP-KMeans. By ignoring temporal order but retaining the basepoint method, we can see how much the sequential order of the data affects clustering quality. The last three columns indicate that the goodness-of-fit metrics predominantly improve with the application of COP-KMeans, with certain cases, such as INSK, exhibiting significant enhancements in clustering quality.

Dataset	KMeans with Basepoints			COP and KMeans ratio (%)		
	RI	ARI	NMI	RI	ARI	NMI
PLN	1.000	1.000	1.000	98.7	95.0	97.0
NIFECG	0.969	0.891	0.933	97.7	91.8	93.5
UWGLX	0.907	0.734	0.869	99.3	95.9	90.7
UWGLAll	0.982	0.945	0.946	100.1	100.3	99.5
EOGHS	0.607	0.193	0.298	112.0	104.7	86.9
PPTW	0.995	0.986	0.980	99.9	99.8	99.6
SLF	1.000	1.000	1.000	100	100	100
SYM	0.842	0.577	0.610	109.3	136.2	120.7
CAR	0.963	0.903	0.902	101.5	104.1	102.5
INSK	0.668	0.348	0.521	129.3	199.7	125.5
Average	0.893	0.758	0.806	104.8	112.8	101.6

Table 5.3: Comparison of clustering performance using KMeans with basepoints, along with the ratio of clustering quality between COP-KMeans and KMeans.

Lastly, Table 5.4 presents the results when using standard KMeans without basepoints, performing clustering with a full comparison of all segments.

Dataset	COP w/o Basepoints			COP with and KMeans w/o BP ratio (%)		
	RI	ARI	NMI	RI	ARI	NMI
PLN	1.000	1.000	1.000	98.7	95.0	97.0
NIFECG	0.964	0.874	0.926	98.2	93.6	94.2
UWGLX	0.915	0.758	0.879	98.5	92.9	89.6
UWGLAll	0.987	0.960	0.960	99.6	98.8	98.0
EOGHS	0.760	0.394	0.542	89.5	51.3	47.8
PPTW	0.995	0.986	0.980	99.9	99.8	99.6
SLF	1.000	1.000	1.000	100	100	100
SYM	0.930	0.812	0.774	98.9	96.8	95.1
CAR	0.949	0.866	0.873	103.0	108.5	106.0
INSK	0.956	0.902	0.880	90.4	77.1	74.3
Average	0.946	0.855	0.881	97.7	91.4	90.2

Table 5.4: Clustering performance comparison of adapted KMeans without basepoints, along with the ratio of clustering quality between COP-KMeans with basepoints and KMeans without basepoints.

We observe that the accuracy loss is more pronounced when using basepoints than when using KMeans instead of COP-KMeans. However, as detailed in Section 5.5, COP-KMeans achieves competitive performance without significantly impacting runtime, while using basepoints greatly improves runtime efficiency.

In summary, these different configurations allow us to assess the trade-offs between computational efficiency and clustering accuracy, as well as the importance of temporal ordering in the segmentation and clustering of time series data.

5.3.2 Cluster Number Estimation Using Elbow and Silhouette Methods

This section summarizes the performance of the elbow and silhouette methods in estimating the correct number of clusters across various datasets. For simpler datasets like "Plane" and "NonInvasiveFetalECGThorax1," both methods accurately predicted the correct cluster numbers. However, for more complex datasets like "UWaveGestureLibraryX," the methods showed discrepancies, particularly with the silhouette analysis, which tended to overestimate the number of clusters. In contrast, the elbow method provided more consistent results but also struggled with certain datasets, such as "EOGHorizontalSignal," where both methods had difficulty matching the ground truth cluster number (see Table 5.5).

Dataset	Elbow method (Result/Difference)	Silhouette analysis (Result/Difference)	Ground truth cluster number
PLN	7/0	7/0	7
NIFECG	6/0	6/0	6
UWGLX	5/0	8/+3	5
UWGLAll	5/0	5/0	5
EOGHS	3/-1	8/+4	4
PPTW	4/0	4/0	4
SLF	4/0	4/0	4
SYM	4/0	6/+2	4
CAR	4/0	4/0	4
INSK	3/0	9/+6	3

Table 5.5: Performance of elbow and silhouette methods in estimating the number of clusters.

5.3.3 Effectiveness of Confidence Score in Enhancing Clustering Performance

In this section, we examine the distribution of confidence scores within the "Symbols" dataset to further understand the relationship between confidence and clustering accuracy. The following plot (Figure 5.1) illustrates how the confidence scores of segments are distributed and highlights the impact of these scores on the overall clustering performance. By analyzing this distribution, we aim to identify thresholds that can effectively distinguish between reliable and unreliable segments, thereby enhancing the effectiveness of the confidence score in improving clustering outcomes. This analysis is crucial for validating the hypothesis that higher confidence correlates with more accurate clustering results.

The scatter chart in Figure 5.2 shows how filtering segments based on confidence scores affects the clustering metrics (RI, ARI, and NMI) for the "Symbols" dataset. As segments with low confidence are progressively removed, the metrics improve significantly. This indicates that segments with lower confidence scores are more likely to be incorrectly labeled. By filtering out these less reliable segments, the overall quality of the clustering improves, as reflected by the rise in all three metrics. This suggests that the confidence score effectively identifies mislabeled or ambiguous segments, leading to more accurate clustering when applied.

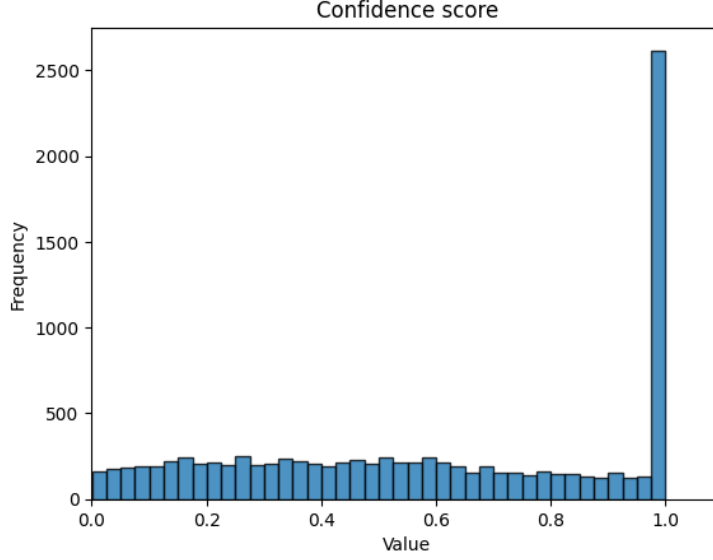


Figure 5.1: Distribution of confidence scores for segments in the "Symbols" dataset.

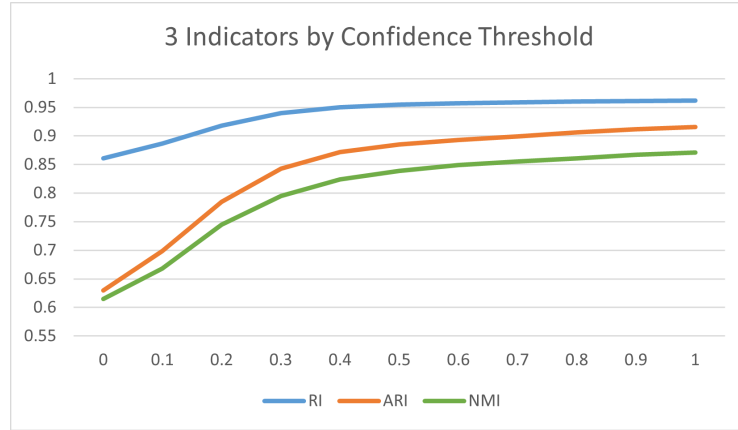


Figure 5.2: Impact of confidence score filtering on clustering performance metrics for the "Symbols" dataset.

5.4 Sensitivity to KS-test

One of the limitations of the KS-test is its inability to differentiate between segments that belong to different clusters but share the same underlying distribution. This is particularly problematic in cases where segments may have distinctive transformations or shifts but maintain identical statistical distributions. In such scenarios, the KS-test fails to capture the differences between the segments, leading to reduced clustering accuracy.

To demonstrate this limitation, we present a series of sinusoidal signals, each belonging to a different cluster. While the clusters are distinct, each segment has the same distribution, and the only transformation applied is a shift along the x-axis.

Next, we applied the KS-test-based method with fifty basepoints on a dataset consisting of 1000 segments, with three distinct clusters, all generated using sinusoidal signals. The results indicate that, due to the similar distributions between clusters, the clustering metrics remain significantly low, reflecting the KS-test's limitations in distinguishing between

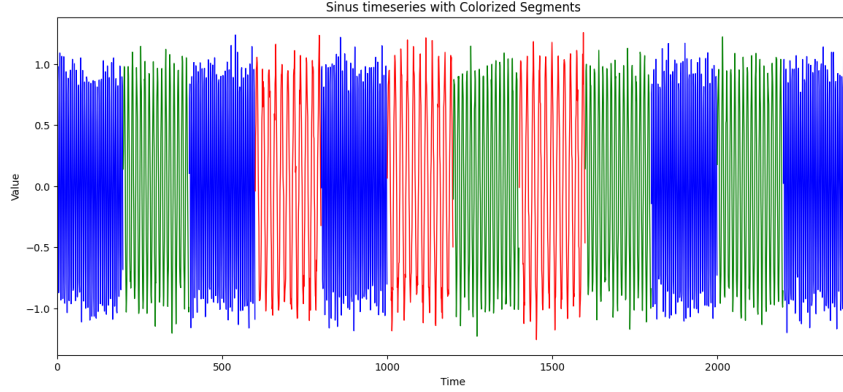


Figure 5.3: Generated sinusoidal signals example.

these transformed signals. Specifically, the Rand Index is 0.579, the Adjusted Rand Index is 0.055, and the Normalized Mutual Information is 0.057. Although these values are slightly better than random, this improvement is mainly due to COP-KMeans imposing a constraint that prevents segments near each other from being in the same cluster, which is the only useful information extracted from the dataset in this case.

Finally, the confidence scores assigned by the method indicate the uncertainty in distinguishing between the clusters. The scores range from 0 to 0.6, with the majority of them being close to 0, further confirming the KS-test’s insensitivity in these cases.

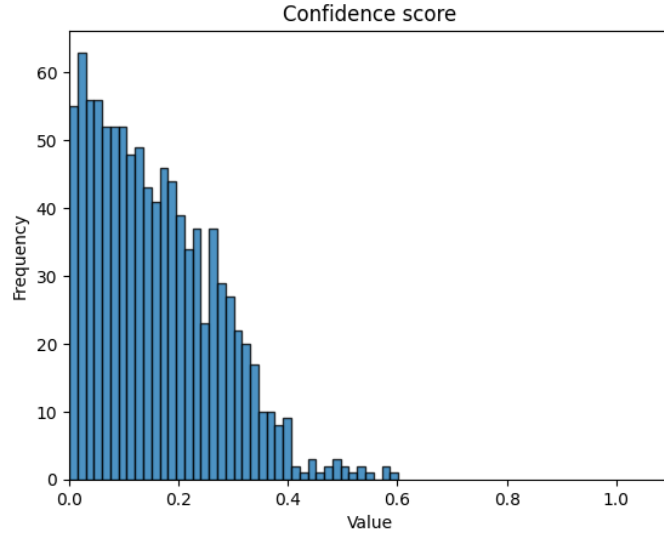


Figure 5.4: Distribution of confidence scores.

5.5 Runtime Analysis

The performance of clustering algorithms is crucial not only in terms of accuracy but also in terms of computational efficiency, especially when dealing with large datasets. In this section, we evaluate the runtime of the COP-KMeans algorithm under various conditions, presenting results based on different numbers of basepoints and segments. By comparing the algorithm’s performance with and without basepoints, as well as with only KMeans (without COP), we gain insight into how the number of segments and basepoints affect

the overall runtime. These results are important for understanding the scalability of the method and its applicability to real-world, large-scale datasets.

We start by comparing three configurations in Table 5.6. The algorithm was run with 50 basepoints, without basepoints, and with basepoints but without using the constraints imposed by COP-KMeans (i.e., only using KMeans). In each case, the dataset size was fixed at 1,000 segments, allowing us to observe the computational trade-offs between these different setups.

Dataset	N × B - COP (mean ± std)	N × N - COP (mean ± std)	N × B - Normal (mean ± std)
PLN	22.8 ± 0.6	375.4 ± 12.3	23.7 ± 0.4
NIFECG	24.1 ± 0.4	417.3 ± 11.4	25.0 ± 0.3
UWGLX	23.1 ± 0.7	370.4 ± 11.6	24.0 ± 0.4
UWGLAll	23.4 ± 0.5	403.0 ± 10.0	24.3 ± 0.5
EOGHS	23.2 ± 0.5	387.6 ± 12.2	23.7 ± 0.4
PPTW	22.3 ± 0.3	382.9 ± 7.5	23.4 ± 0.4
SLF	22.1 ± 0.7	374.9 ± 6.4	23.0 ± 0.5
SYM	23.3 ± 0.7	400.8 ± 6.4	23.9 ± 0.3
CAR	21.3 ± 0.6	363.9 ± 7.3	23.0 ± 0.2
INSK	22.9 ± 0.4	404.0 ± 7.6	24.0 ± 0.4
Average	22.8 ± 0.5	388.0 ± 9.3	23.8 ± 0.4

Table 5.6: Runtime comparison of adapted COP-KMeans configurations with 1,000 segments.

Next, Table 5.7 presents the runtime results with a fixed dataset size of 1,000 segments but varying the number of basepoints (25, 50, 100, 200, 1,000). This shows how sensitive the algorithm is to changes in the number of basepoints, which directly influences the runtime complexity.

Dataset \ B	25	50	100	200	500
PLN	11.1 ± 0.6	23.7 ± 0.4	54.5 ± 0.5	142.7 ± 0.6	615.2 ± 0.7
NIFECG	11.6 ± 0.4	24.7 ± 0.3	56.9 ± 0.4	150.6 ± 0.4	659.4 ± 0.5
UWGLX	11.4 ± 0.7	23.9 ± 0.4	55.1 ± 0.4	142.6 ± 0.5	617.0 ± 0.6
UWGLAll	11.6 ± 0.5	23.8 ± 0.5	54.9 ± 0.4	146.6 ± 0.5	642.1 ± 0.5
EOGHS	11.3 ± 0.5	23.8 ± 0.4	54.1 ± 0.4	143.4 ± 0.5	630.5 ± 0.4
PPTW	11.4 ± 0.3	22.7 ± 0.4	51.4 ± 0.4	140.6 ± 0.5	617.8 ± 0.5
SLF	10.6 ± 0.7	23.0 ± 0.5	52.4 ± 0.4	140.9 ± 0.5	623.0 ± 0.5
SYM	11.8 ± 0.7	24.2 ± 0.3	54.0 ± 0.4	144.8 ± 0.5	622.3 ± 0.6
CAR	10.6 ± 0.6	22.1 ± 0.2	51.5 ± 0.5	138.0 ± 0.4	620.4 ± 0.4
INSK	11.1 ± 0.4	23.5 ± 0.4	54.9 ± 0.4	145.2 ± 0.4	636.6 ± 0.5
Average	11.3 ± 0.5	23.5 ± 0.4	54.0 ± 0.4	143.5 ± 0.5	628.4 ± 0.5

Table 5.7: Runtime results with varying numbers of basepoints at a fixed dataset size of 1,000 segments.

In the third table, Table 5.8, the number of basepoints was fixed at 50, and the number of segments was varied (100, 500, 1,000, 5,000, 10,000) to examine the algorithm’s scalability when the dataset size increases. This helps us understand how the algorithm performs with larger datasets.

Dataset \ N	100	500	1000	5000	10000
PLN	6.4 ± 0.6	14.6 ± 0.4	23.7 ± 0.4	96.5 ± 0.5	197.3 ± 0.6
NIFECG	6.3 ± 0.4	15.6 ± 0.3	24.7 ± 0.3	106.7 ± 0.4	199.8 ± 0.5
UWGLX	6.3 ± 0.7	14.4 ± 0.4	23.9 ± 0.4	96.0 ± 0.5	190.8 ± 0.6
UWGLAll	6.4 ± 0.5	14.9 ± 0.5	23.8 ± 0.5	101.6 ± 0.5	190.6 ± 0.5
EOGHS	6.4 ± 0.5	14.7 ± 0.5	23.8 ± 0.4	100.2 ± 0.5	192.4 ± 0.5
PPTW	6.3 ± 0.3	14.2 ± 0.4	22.7 ± 0.4	94.7 ± 0.5	178.2 ± 0.5
SLF	6.4 ± 0.7	14.4 ± 0.5	23.0 ± 0.5	96.1 ± 0.5	180.6 ± 0.5
SYM	6.4 ± 0.7	14.7 ± 0.3	24.2 ± 0.4	100.5 ± 0.5	184.8 ± 0.5
CAR	6.4 ± 0.6	13.9 ± 0.2	22.1 ± 0.5	92.5 ± 0.4	178.5 ± 0.4
INSK	6.4 ± 0.4	15.2 ± 0.4	23.5 ± 0.4	97.4 ± 0.4	188.5 ± 0.5
Average	6.4 ± 0.5	14.5 ± 0.4	23.4 ± 0.4	98.7 ± 0.5	188.5 ± 0.5

Table 5.8: Runtime results with a fixed number of basepoints (50) and varying numbers of segments.

In addition, Figure 5.5 and 5.6 illustrates the average runtime of the adapted COP-KMeans algorithm as the number of segments (n) and basepoints (b) vary. The algorithm’s complexity is $O(b \cdot n \cdot L \cdot \log L + b^3 \cdot n + n \cdot b \cdot k \cdot I)$, indicating linear scaling with respect to n and power function behavior with respect to b . The plots confirm these trends, aligning with theoretical expectations for runtime behavior.

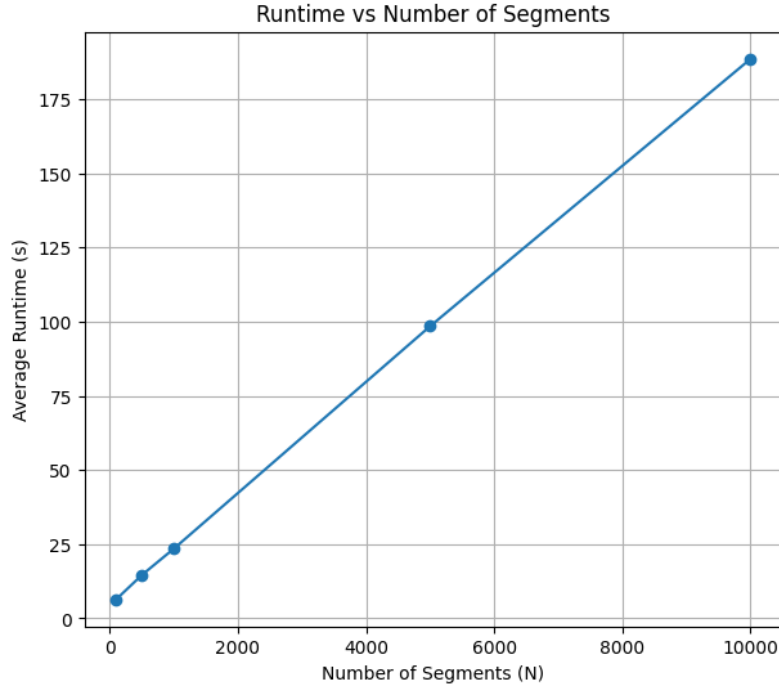


Figure 5.5: Average runtime of the adapted COP-KMeans algorithm based on varying segments (n).

5.6 Comparison with TimeSeriesKMeans

In this section, we compare our proposed clustering method with the TimeSeriesKMeans algorithm implemented in the `tslearn` library [25]. TimeSeriesKMeans is a widely used method for clustering time series data and serves as a strong baseline for our evaluation.

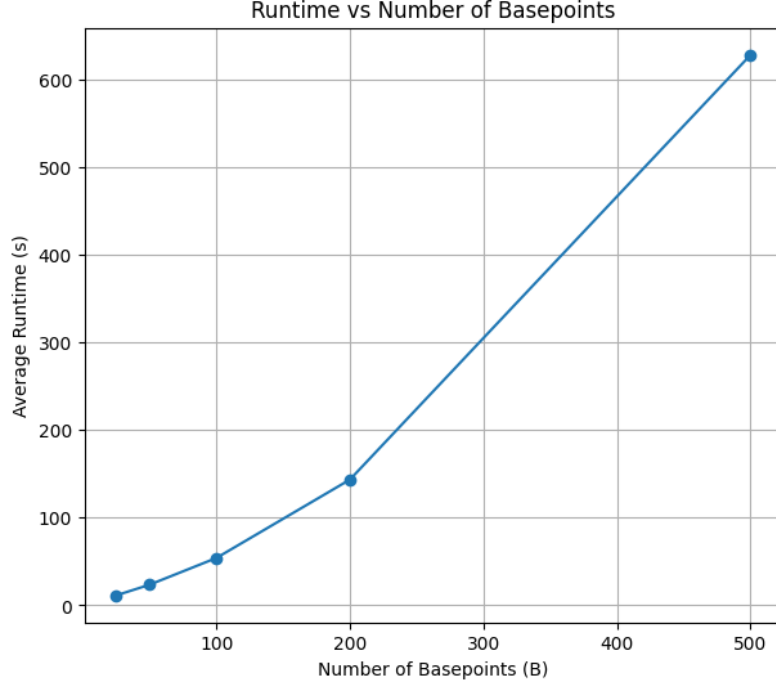


Figure 5.6: Average runtime of the adapted COP-KMeans algorithm based on varying basepoints (b).

The comparison is conducted in several aspects, including methodological differences [18, 20], computational complexity [16], and empirical performance [5]. Through this analysis, we aim to highlight the strengths and weaknesses of each approach, demonstrating the potential advantages of our method in terms of accuracy, efficiency, and scalability across various time series datasets.

5.6.1 Overview of TimeSeriesKMeans

TimeSeriesKMeans [25] is a variant of the traditional KMeans algorithm, specifically adapted for time series data. It employs a specialized distance metric, typically dynamic time warping (DTW) or soft DTW, to handle temporal dependencies within sequences [22, 4]. This metric allows the algorithm to account for potential misalignments and variations in time series data, making it suitable for a range of applications where time-dependent structure is important. TimeSeriesKMeans has been shown to perform well in many scenarios, especially when time series are of similar lengths and share comparable temporal patterns [18].

5.6.2 Methodological Differences

Our proposed method differs from TimeSeriesKMeans in several key areas. First, our method leverages statistical tests, such as the KS test, to assess the distributional similarity between segments, enabling it to capture not only shape but also distributional characteristics. Additionally, our method incorporates basepoints, which allow for a more flexible representation of clusters when dealing with irregular or heterogeneous time series data [29].

A key advantage of our approach is that it can handle segments of varying lengths, while TimeSeriesKMeans requires all segments to be of the same length. To accommodate this constraint for comparison purposes, in our research, we preprocess the data by cutting each segment to match the length of the shortest one across the dataset. This step ensures compatibility with TimeSeriesKMeans while maintaining the integrity of our method’s ability to handle diverse segment lengths. In contrast, TimeSeriesKMeans aligns sequences based on DTW, which can be computationally intensive and may not capture nuanced statistical properties of the data as effectively.

5.6.3 Complexity Comparison

The computational complexity of TimeSeriesKMeans is primarily governed by the calculation of DTW distances across all time series and the iteration over cluster assignments. For n time series of length L and k clusters, the complexity is $O(n \cdot k \cdot L^2 \cdot I)$, where I denotes the number of iterations until convergence, assuming DTW-based distance calculation [16].

Our method, by contrast, has a complexity of $O(b \cdot n \cdot L \cdot \log L + b^3 \cdot n + n \cdot b \cdot k \cdot I)$, where b represents the number of basepoints, n the number of segments, L the segment length, k the number of clusters, and I the number of iterations. This formulation allows for a potentially more scalable approach as segment lengths and the number of segments grow, with basepoints enabling a significant reduction in redundancy and complexity.

5.6.4 Empirical Performance Comparison

We conducted experiments on various time series datasets to compare the clustering quality and computational efficiency of our method and TimeSeriesKMeans [5]. The results, summarized in Table 5.9, demonstrate that our approach (with 50 basepoints) outperforms TimeSeriesKMeans in most scenarios, particularly in datasets exhibiting high heterogeneity or distributional variation. However, in the case of the EOGHS dataset, TimeSeriesKMeans achieved better performance. This outcome can be attributed to the clusters in EOGHS having similar distributions, with only the signal shapes differing. Consequently, TimeSeriesKMeans, which relies on DTW for alignment, is more effective in capturing these patterns.

The table also illustrates that runtime varies significantly, primarily influenced by the length of the shortest segments in the datasets. Our method, utilizing basepoints, generally offers enhanced computational efficiency compared to DTW-based measures while maintaining competitive clustering performance.

Dataset	TimeSeriesKMeans			Differences			TimeSeriesKMeans Runtime (s)	COP with BP Runtime (s)
	RI	ARI	NMI	RI	ARI	NMI		
PLN	0.790	0.181	0.322	0.197	0.769	0.648	27.6	22.8
NIFECG	0.763	0.148	0.275	0.184	0.670	0.597	351.1	24.1
UWGLX	0.819	0.442	0.582	0.082	0.262	0.206	42.2	23.1
UWGLAll	0.872	0.647	0.796	0.111	0.301	0.145	236.7	23.4
EOGHS	0.818	0.523	0.637	-0.138	-0.321	-0.378	43.9	23.2
PPTW	0.590	-0.001	0.003	0.404	0.985	0.973	5.5	22.3
SLF	0.706	0.282	0.444	0.294	0.718	0.556	105.1	22.1
SYM	0.686	0.167	0.253	0.234	0.718	0.556	20.6	23.3
CAR	0.607	0.005	0.012	0.370	0.935	0.913	29.7	21.3
INSK	0.570	0.042	0.337	0.294	0.653	0.604	107	22.9
Average	0.722	0.244	0.337	0.203	0.559	0.475	96.94	22.8

Table 5.9: Comparison of clustering metrics between TimeSeriesKMeans and our method across various datasets and runtime.

Chapter 6

Robustness Analysis with Noise

In real-world applications, time series data is rarely free from noise or distortion. Sensor signals, for example, often experience fluctuations due to environmental factors, measurement errors, or hardware limitations. Therefore, it is crucial to assess how well a clustering method performs under noisy conditions to ensure its reliability and adaptability. By introducing noise into the data and examining the impact on clustering results, we can evaluate the stability of the method and its ability to maintain performance despite varying levels of uncertainty. This robustness analysis not only highlights the strengths of the method but also uncovers potential weaknesses, providing insight into its limitations and guiding future improvements.

6.1 Noise Injection Strategy

In order to evaluate the robustness of the clustering method, noise is systematically injected into the segments using a Gaussian noise model. This approach allows us to simulate real-world imperfections in sensor data and assess the impact of noise on clustering performance. The noise injection strategy is controlled by two key parameters:

- **Noise Percentage:** This parameter defines the proportion of data points within a segment that will be affected by noise. For example, an noise percentage of 10% means that 10% of the data points in each segment will be modified by noise.
- **Noise Magnitude:** This parameter determines the magnitude of the noise added to the selected data points. The noise follows a Gaussian distribution with a mean of 0 and a standard deviation proportional to the maximum value of the segment. The standard deviation ranges from 0% to 100%, where 100% corresponds to the largest possible noise relative to the segment's data range.

The noise injection is applied in incremental steps, with both the noise percentage and the standard deviation varying from 0% to 100% in 10% steps. This systematic variation allows us to analyze the method's performance across different levels of noise intensity and distribution.

6.2 Results

To evaluate the impact of noise on clustering performance, we generated 11x11 ARI heatmaps, where the x-axis represents the noise magnitude, and the y-axis indicates the sample percentage affected by noise. These heatmaps provide a clear visual representation of how well the clustering method holds up as the level of noise increases. COP-KMeans was run 10 times for each configuration to ensure accuracy, and the results were averaged to smooth out any randomness in the process.

The results fall into three distinct categories based on the behavior of the ARI values under varying levels of noise:

1. **Low Performance Without Noise:** In some datasets, the clustering method already struggles to produce accurate results, even in the absence of noise. These heatmaps are generally dark throughout, indicating low ARI values across all levels of noise. The clustering accuracy remains poor, and additional noise only slightly worsens the outcome, but the method already fails to achieve meaningful clustering (see Figure 6.1).

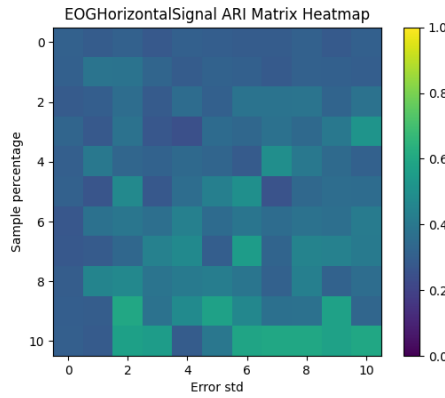


Figure 6.1: EOGHorizontalSignal dataset’s ARI heatmap.

2. **High Sensitivity to Noise:** In another set of datasets, the clustering method performs well in the absence of noise, achieving high ARI values. However, as noise levels increase, the clustering accuracy drops dramatically. The ARI values approach zero at high noise levels, indicating that the method is highly sensitive to noise and cannot maintain its performance as the data becomes distorted (see Figure 6.2).
3. **Moderate Sensitivity to Noise:** Lastly, in some datasets, the clustering method starts with high accuracy without noise but remains relatively robust as noise is introduced. Although there is a clear drop in ARI values as the noise levels increase, the accuracy does not plummet to zero. Instead, the method maintains moderate performance with ARI values in the range of 0.5-0.6, even under high noise conditions. This shows that the method has some degree of resilience to noise and can still produce useful clusters even when the data is imperfect (see Figure 6.3).

These results highlight the varying degrees of sensitivity of the clustering method to noise, depending on the dataset and the nature of the sensor signals. The robustness analysis demonstrates that while the method can perform well under ideal conditions, its resilience to noise varies significantly, suggesting areas for further optimization in noisy environments.

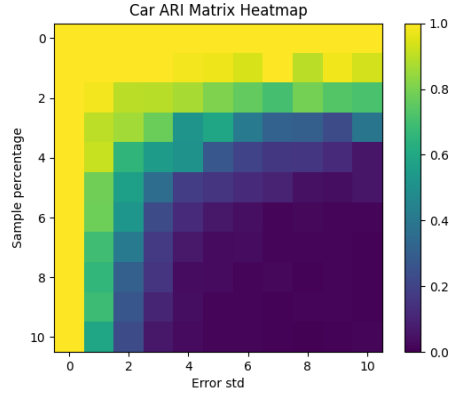


Figure 6.2: Car dataset's ARI heatmap.

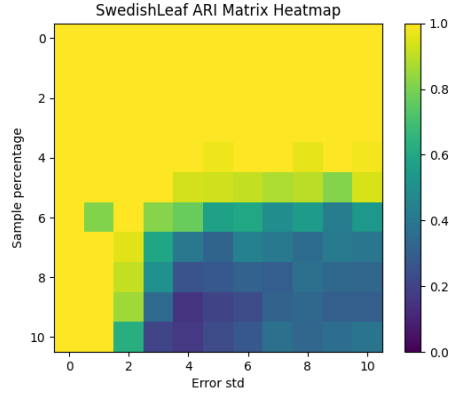


Figure 6.3: SwedishLeaf dataset's ARI heatmap.

6.3 Confidence Score Analysis

In addition to examining the ARI values across various noise levels, we also evaluated the confidence scores assigned to each segment. As noise is introduced into the dataset, we observe that the distribution of these confidence scores shifts toward zero, indicating a decrease in the reliability of the clustering process.

6.3.1 Confidence Score Distribution with Increasing Noise

To illustrate this, we selected the Car dataset and injected three different levels of noise into it. We show how the confidence score distribution changes as noise increases:

- **Low noise:** At a low level of noise (e.g., 20% noise percentage and noise magnitude), the confidence scores remain high for the majority of segments, indicating that the clustering method is still able to make reliable predictions (see Figure 6.4).
- **Moderate noise:** As we increase the noise (e.g., 40% noise percentage and noise magnitude), we observe a noticeable shift in the confidence score distribution, with more segments receiving lower confidence scores. This suggests that the clustering method begins to struggle in differentiating segments as the noise level rises (see Figure 6.5).

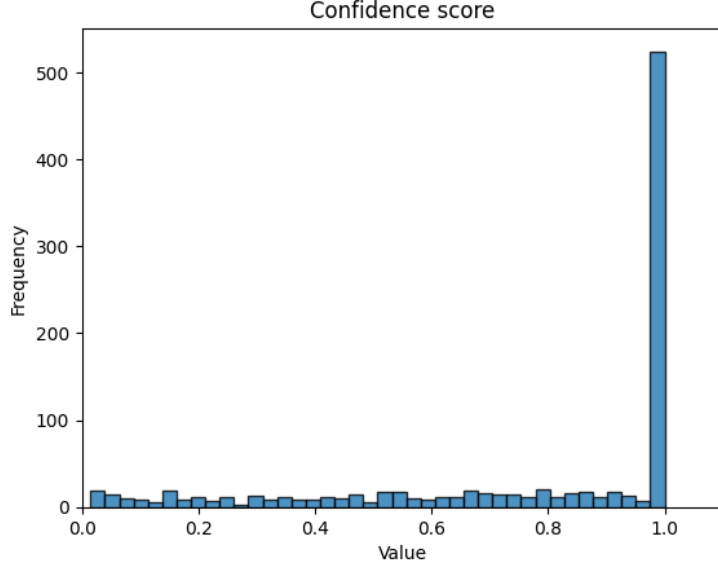


Figure 6.4: Car dataset’s confidence distribution with 20% noise.

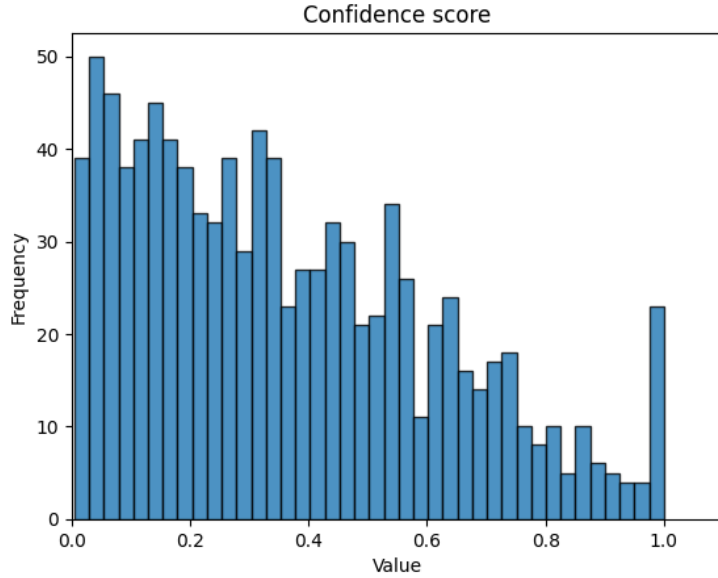


Figure 6.5: Car dataset’s confidence distribution with 40% noise.

- **High noise:** At the highest noise level (e.g., 100% noise percentage and noise magnitude), the confidence scores drop significantly, with most scores concentrated near zero. This highlights the difficulty the method has in making confident predictions under high noise conditions (see Figure 6.6).

6.3.2 Impact of Confidence Threshold on Clustering Metrics

To further analyze the effect of confidence on clustering performance, we injected noise with 40% noise percentage and noise magnitude into the Car dataset. As we increase the confidence threshold from 0 to 1, the clustering metrics (RI, ARI, and NMI) improve. However, this improvement comes at a cost: many segments are discarded, leading to

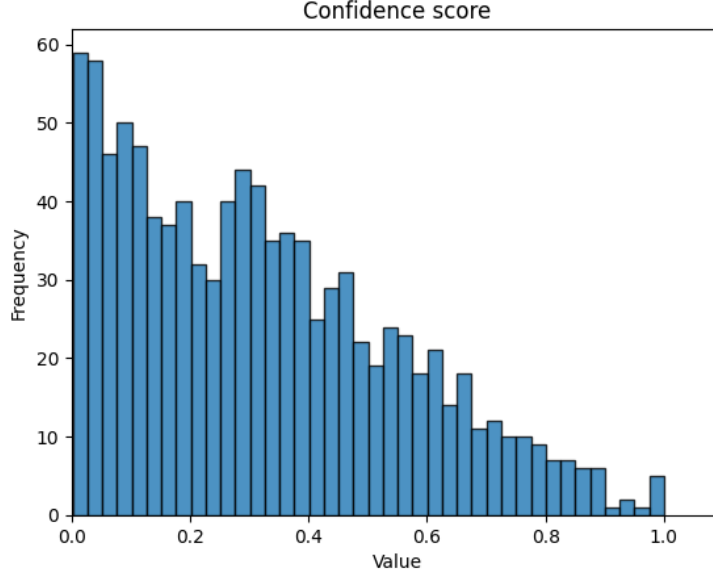


Figure 6.6: Car dataset’s confidence distribution with 100% noise.

a trade-off between maintaining higher confidence levels and retaining more data (see Figure 6.7).

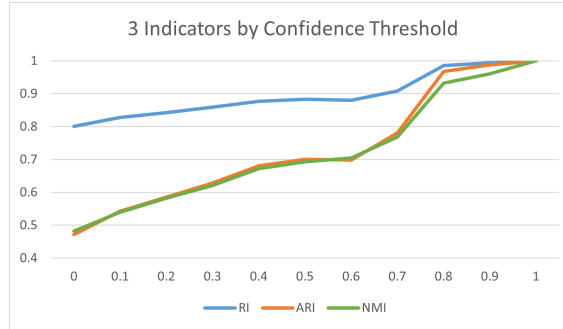


Figure 6.7: Car dataset’s clustering metrics as a function of confidence threshold with 40% noise.

When noise is increased to 100%, this trend no longer holds. The metrics no longer improve as we raise the confidence threshold due to the significant data loss caused by the high level of noise (see Figure 6.8). This demonstrates that while filtering by confidence can lead to better performance with moderate noise, it also reduces the amount of usable data, making it less effective at extreme noise levels.

6.4 Interpretation of Results

The robustness of the clustering results when noise is added is highly dependent on the characteristics of the dataset. Specifically, datasets where the distributions of segment data are significantly distinct from one another tend to be more resilient to noise. In such cases, even with moderate levels of noise, the clustering process can still differentiate between the segments effectively.

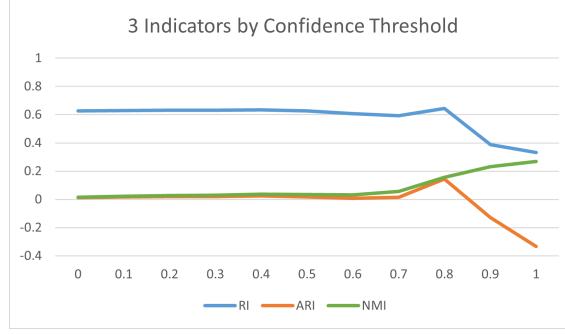


Figure 6.8: Car dataset’s clustering metrics as a function of confidence threshold with 100% noise.

However, for datasets where the distributions of segments are already close to one another, adding noise has a much greater impact. The noise causes the segment distributions to blur into each other, making it harder for the clustering algorithm to maintain clear boundaries between them. As a result, the performance metrics of the clustering degrade more rapidly in these cases.

This demonstrates that the success of the clustering process in noisy conditions depends not only on the amount of noise but also on how easily separable the original segment distributions are. As noise increases, the boundaries between the clusters become less distinct, leading to a decline in clustering accuracy.

Chapter 7

Conclusion

In this paper, we presented a novel approach for clustering time-varying segments by integrating statistical methods with constrained clustering techniques. The primary contributions of this work include the development of a hybrid clustering framework that combines the Kolmogorov–Smirnov (KS) test [11] and adapted COP-KMeans [28] for clustering with temporal constraints. This approach addresses several key challenges in time series clustering, such as segment misalignment and varying lengths.

The core innovation lies in using the KS test to generate distribution-based embeddings that capture statistical differences between segments. These embeddings provide a robust representation of the data, independent of temporal misalignment and length variation. Adapted COP-KMeans is then applied with cannot-link constraints, ensuring that consecutive segments are assigned to different clusters, preserving their temporal structure.

To improve computational efficiency, we introduced a basepoint selection strategy, reducing the dimensionality of the embedding space from $O(n^2)$ to $O(n \cdot b)$. This effectively decreases the computational complexity of the KS-based embedding construction from $O(n^2 \cdot L \log L)$ to $O(n \cdot b \cdot L \log L)$, where n is the number of segments, b is the number of selected basepoints, and L is the segment length. The experimental results demonstrate that this basepoint reduction significantly enhances runtime performance without a substantial loss in clustering accuracy (with about 10% goodness reduction, a 17-times speed-up could be achieved). This efficiency gain is crucial for large-scale industrial datasets, where the traditional exhaustive pairwise comparison would be computationally prohibitive.

In addition to the basepoint strategy, we introduced a confidence score metric that quantifies the reliability of clustering assignments based on the relative distances to the nearest and second-nearest cluster centroids. This score provides a nuanced view of the clustering results, allowing low-confidence labels to be filtered out, thus improving the overall quality of the clustering.

The performance of our method was evaluated on a diverse set of benchmark time series datasets, including motion data and physiological signals. We measured clustering performance using Rand Index (RI), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI) [14, 27]. The results demonstrate that our approach consistently outperforms traditional clustering techniques in terms of computational efficiency. Furthermore, the confidence-based filtering method provided a marked improvement in clustering quality by discarding low-confidence assignments.

Our method was evaluated against TimeSeriesKMeans [20], and the results highlighted its capacity to deliver fast and accurate clustering solutions across a variety of datasets.

While our approach generally outperformed TimeSeriesKMeans, particularly in scenarios marked by high heterogeneity and distributional variation, there were instances where TimeSeriesKMeans excelled. Notably, in the EOGHS dataset, the similar distribution of clusters, despite differing signal forms, allowed TimeSeriesKMeans to achieve superior performance due to its reliance on Dynamic Time Warping (DTW) [17] for optimal alignment. This underscores the importance of selecting the appropriate clustering technique based on the dataset characteristics, emphasizing that, while our method offers robust solutions, DTW-based approaches may be more suitable in specific contexts where the alignment of similar distributions is critical.

While our proposed method shows strong performance across multiple datasets, several avenues for future research remain open. First, the use of alternative statistical tests and distance measures could enhance the sensitivity of the embedding space, particularly in cases where segments share the same distribution but differ in temporal structure or magnitude. The incorporation of more sophisticated alignment techniques, such as shape-based clustering methods [12] or Dynamic Time Warping (DTW) [17], could further improve the accuracy of our approach in handling complex, multi-dimensional time series data.

Second, while the basepoint selection strategy significantly reduces the dimensionality of the embeddings, optimizing the selection process remains an area for further exploration. Methods such as k-medoids or greedy optimization could be employed to select a more diverse and representative set of basepoints, potentially improving clustering performance while maintaining computational efficiency.

In conclusion, the proposed framework provides a robust, scalable, and interpretable solution to time series clustering, combining statistical rigor with computational efficiency. Our method offers a significant step forward in addressing the challenges of temporal misalignment, variable segment lengths, and large dataset scalability in time series clustering. By balancing the theoretical rigor of statistical tests with the practical demands of large-scale data analysis, this work sets the stage for future advancements in time series clustering methodologies. Ultimately, the choice of method—whether our proposed technique or a DTW-based approach—should be guided by the specific characteristics and requirements of the dataset at hand, ensuring that researchers and practitioners can make informed decisions to optimize clustering performance.

Bibliography

- [1] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering—a decade review. *Information systems*, 53:16–38, 2015.
- [2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [3] J. Caiado, E. A. Maharaj, and P. D’Urso. Time series clustering. In *Handbook of Cluster Analysis*, pages 241–263. CRC Press, 2015.
- [4] Marco Cuturi and Mathieu Blondel. Soft-dtw: A differentiable loss function for time-series. *Proceedings of the 34th International Conference on Machine Learning*, 70: 894–903, 2017.
- [5] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, and et al. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
- [6] A. Ermshaus, P. Schäfer, and U. Leser. Clasp: parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37(3):1262–1300, 2023.
- [7] R. Fujimaki, S. Hirose, and T. Nakata. Theoretical analysis of subsequence time-series clustering from a frequency-analysis viewpoint. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 506–517. Society for Industrial and Applied Mathematics, 2008.
- [8] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems*, 8: 154–177, 2005.
- [9] Shahbaz M. Khan and Farhat A. Bhat. An overview of clustering techniques in data mining. *International Journal of Computer Applications*, 181(32):22–25, 2018.
- [10] J. Kodinariya and S. Makwana. Review on determining number of clusters in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1(6):90–95, 2013.
- [11] A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *Giornale dell’Istituto Italiano degli Attuari*, 4:83, 1933.
- [12] Y. Li, D. Shen, T. Nie, and Y. Kou. A new shape-based clustering algorithm for time series. *Information Sciences*, 609:411–428, 2022.
- [13] T. W. Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11): 1857–1874, 2005.

- [14] M. McCrory and S. A. Thomas. Cluster metric sensitivity to irrelevant features. *arXiv preprint*, arXiv:2402.12008, 2024.
- [15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, 2013.
- [16] Abdullah Mueen and Eamonn Keogh. Exact discovery of time series motifs. *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 473–484, 2012.
- [17] M. Müller. *Dynamic Time Warping*, pages 69–84. Springer, 2007.
- [18] Francois Petitjean, Alain Ketterlin, and Alain Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.
- [19] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Mdl-based time series clustering. *Knowledge and Information Systems*, 33:371–399, 2012.
- [20] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, and E. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):1–31, 2013.
- [21] 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.
- [22] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
- [23] Amit Singh and Anil Kumar Jain. A survey on clustering algorithms. *International Journal of Computer Applications*, 111(12):1–8, 2015.
- [24] G. Szűcs, M. B. Tóth, and M. Németh. Distribution controlled clustering of time series segments by reduced embeddings. *International Conference on Pattern Recognition Applications and Methods (ICPRAM 2025)*, 2025 (submitted).
- [25] Romain Tavenard, Johann Faouzi, and Gilles et al. Vandewiele. Tsllearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.
- [26] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.
- [27] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080, 2009.
- [28] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, volume 1, pages 577–584, 2001.
- [29] X. Wang, Abdullah Mueen, H. Ding, and et al. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26:275–309, 2012.

- [30] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh. A review of subsequence time series clustering. *The Scientific World Journal*, 2014(1):312521, 2014.