The CSI Journal on Computer Science and Engineering

Journal homepage: www.csionjcse.ir

A Novel Method for Whole Time Series Clustering via Sub-Pattern Recognition

Saeed Kamro[1], Majid Rafiee[1]*

[1] Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

*** Corresponding author email address**: rafiee@sharif.edu

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Kamro, S., & Rafiee, M. (2026). A Novel Method for Whole Time Series Clustering via Sub-Pattern Recognition. *The CSI Journal on Computer Science and Engineering, 20*(2), 50-75.
http://dx.doi.org/10.22034/jcse.2026.577699.1077

A B S T R A C T

Whole time-series clustering (WTSC) methods often fail to treat sub-patterns as independent entities, limiting their ability to capture diverse behavioral characteristics in time series data. To address this limitation, this study proposes a sliding window-based whole time series clustering method (SW-WTSC) that enhances clustering performance through sub-pattern recognition. The method extracts sub-samples using a sliding window over training data, applies normalization and feature selection, and clusters the resulting sub-samples via k-means. Cluster labels are then aggregated at the sample level, enabling secondary clustering based on the highest similarity among sub-patterns. The proposed approach was evaluated on 85 datasets from the UCR repository. Statistical analysis using the non-parametric Friedman test demonstrated that SW-WTSC significantly outperforms state-of-the-art WTSC methods. These results indicate that incorporating subsequence-based analysis can substantially improve whole time-series clustering accuracy.

Keywords: *Time Series; Clustering; Sliding Window; Feature Selection; Pattern recognition*

Article history:
Published online 07 August 2026

1 Introduction

Time plays a crucial role in data generation, as every change in the characteristics of a phenomenon can be recorded over time and analyzed to uncover hidden patterns [1]. These data types, commonly referred to as time series, are increasingly prevalent and collected across a wide range of domains, including finance [2], manufacturing [3], healthcare [4], and the Internet of Things (IoT) [5], underscoring their growing significance. Due to their real-world origins, time series data often exhibit complex characteristics such as non-linearity, missing values, noise, dynamism, and high specificity to the problem domain. These challenges necessitate the use of diverse machine learning techniques, including classification and clustering, to effectively analyze and address them [6-8].

The application of clustering extends beyond time series data, aiming to group data points into clusters such that instances within the same cluster exhibit high similarity. In contrast, clusters themselves remain as distinct as possible. However, in the presence of noise, high dimensionality, and strong correlations among data points in time series, the performance of conventional clustering methods may degrade significantly. under such challenging conditions, the importance of clustering becomes even more pronounced as it distills complex information into meaningful patterns, making the data more interpretable. Time series clustering, therefore, emerges as a specialized branch of clustering designed to adapt traditional clustering processes to accommodate the unique characteristics of time series data [9-11].

Time series clustering problems can be broadly divided into three main categories: (i) Whole Time Series Clustering (WTSC), which focuses on clustering independent time series; (ii) Subsequence Time Series Clustering (STSC), which clusters subsequences extracted from a single time series using a sliding window, and (iii) Time Point Clustering, which assumes each point in a time series is independent and performs clustering based on values and temporal proximity [12]. Among these categories, the first two primarily cluster time series based on the shape of patterns. Due to the shifting cluster centers in STSC with each iteration, the resulting outcomes are often meaningless, leading to limited research in this area. Consequently, most studies in the field have concentrated on WTSC [13].

WTSC has extensive applications in real-world problems, such as pattern recognition [14] and anomaly detection [3, 15]. It is one of the most prominent tools for analyzing time series data. Various modifications have been introduced to clustering algorithms to address the diverse requirements of real-world problems. Based on these modifications, time series clustering methods can be classified into five main categories, including (i) representation-based methods [16], (ii) feature-based methods [17], (iii) distance-based methods [18], (iv) fuzzy-based methods [19], and (v) model-based methods [20].

Since clustering algorithms are unsupervised and are unaware of the domain-specific similarity measures required for a given problem, the resulting clusters may not always align with practical expectations. This occurs because the algorithm optimizes its objective function based on the homogeneity and heterogeneity of all data features without distinguishing features critical to the specific problem. Consequently, domain knowledge is crucial in guiding the clustering process during preprocessing to ensure meaningful results [21].

A comprehensive review of the literature found that most research efforts assume a time series represents a single pattern, with no emphasis on clustering based on sub-patterns. Consequently, more sophisticated methods have been developed to handle complex time series, while promising research highlights the potential of achieving good performance by simplifying the problem by dividing the data space into subspaces [22]. Furthermore, due to the proven limitations of STSC, the potential capabilities of this approach in the context of WTSC remain unexplored. Building on these insights, this study presents the following key contributions:

Proposing a Novel SW-WTSC Method: Introducing a new approach for WTSC based on identifying sub-patterns generated by a sliding window mechanism.

Verification of SW-WTSC: Evaluating the proposed SW-WTSC method by applying it to UCR datasets and benchmarking its results against competitive methods.

Developing a New Feature Selection Method: Presenting a novel feature selection approach for time series data based on the similarity of data points to a predefined set of specific patterns.

Leveraging Subsequence Clustering Capabilities: Identifying and demonstrating the usable aspects of subsequence clustering within the context of whole clustering.

Introducing Flexibility in Clustering: Developing a clustering framework that focuses on specific patterns within time series data based on the problem's requirements.

The structure of the subsequent sections of the paper is as follows: **Section 2** reviews the fundamental concepts and the existing literature related to the topic under study, While **Section 3** introduces the concepts, materials, and elements influencing the proposed method, along with a detailed explanation of the technique itself. **Section 4** presents numerical results on the performance of the proposed method across various experiments and discusses the findings in detail. Finally, **Section 5** summarizes our findings, concludes the study, and discusses potential directions for future research.

2 Literature review

This section thoroughly analyzes the problem, examines various solution categories, and offers a detailed review of the related works within each category, highlighting key contributions and methodologies in this research domain. Finally, the research gap will be identified to demonstrate the areas that our new method aims to address.

Solving a problem through clustering consists of several phases, where the accurate execution of each phase is essential to achieving an appropriate outcome. A general overview of this process is illustrated in Figure 1. Time series clustering follows the same framework, and studies addressing such problems can be categorized according to the phases of this process. Based on this framework, after precisely defining the issue, a preprocessing step is required to prepare the time series data for the clustering model's training phase. This preprocessing step typically involves either extracting suitable features or representing the data in an appropriate format. The clustering model is then developed based on factors such as the proximity measure, the cluster membership style, and the algorithm employed. After the model is trained using the preprocessed data, its success in achieving the intended objectives is evaluated by analyzing the results. If the desired outcomes are not achieved, parameter adjustments or modifications to the clustering model's structure are made to improve the results [23].

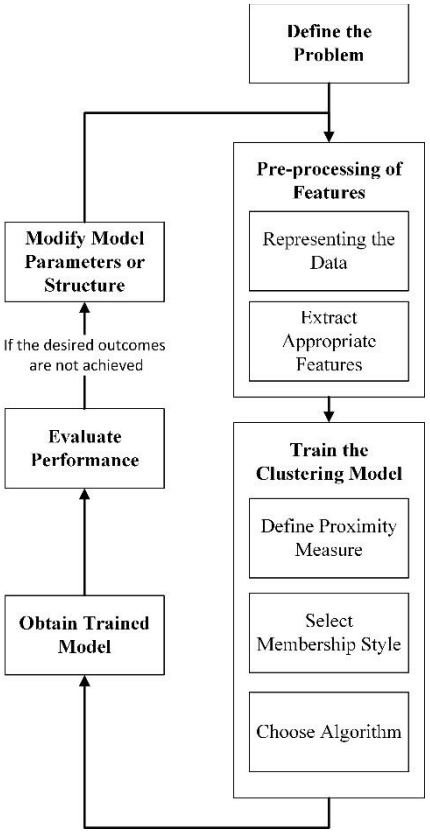

Figure 1 - general overview of clustering phases

2.1 Problem definition

From the perspective of problem definition, three main types of problems are considered in the field of clustering: 1) **whole clustering**, which focuses on clustering a set of independent time series; 2) **subsequence clustering**, which attempts to cluster subsequences extracted from a time series using a sliding window; and **3) time point clustering**, which treats individual time points independently rather than as a connected sequence forming a time series, clustering them based on their values and temporal proximity to one another [12]. According to the findings presented in [13], subsequence clustering often produces Pseudo-sine waves and unstable cluster centroids, making the clustering process meaningless in most cases. However, under specific conditions, the resulting clusters can be meaningful. This observation significantly influenced subsequent research, leading most studies to focus on whole clustering. Nevertheless, this shift in focus did not halt investigations into subsequence clustering, as research continued to generate meaningful clusters [24, 25]. It was also shown that despite the often meaningless results of subsequence clustering, the method could still be used to identify actual patterns. Moreover, increasing the number of clusters could

somewhat address the meaningless results [26]. The problem addressed in this study is the whole clustering and the methods for solving it are examined in the following section.

2.2 *Related works*

Although various classifications have been proposed for WTSC methods, they can be categorized into five main groups in line with the phases of clustering model development: (i) representation-based methods, (ii) feature-based methods, (iii) distance-based methods, (iv) fuzzy-based methods, and (v) model-based methods. The first category focuses on representing time series data. These methods aim to reduce the complexity of time series and enhance the performance of clustering models by transforming time series into new representations, often with lower dimensions. Traditional methods such as Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Principal Component Analysis (PCA), Piecewise Aggregation Approximation (PAA), and Symbolic Aggregate Approximation (SAX) are examples of this category [16]. Some newer methods are also derived from these traditional techniques. For instance, [27] utilized the SAX algorithm to convert long time series into a set of codewords, enabling the analysis of these codewords to propose probabilistic models for time series clustering. To enhance clustering models that cannot typically cluster time series, [28] proposed a representation learning framework to transform time series into an instance-feature matrix called SPIRAL.

The second category is focused on feature extraction. These methods aim to capture rich features from the complex and dynamic patterns in time series data by generating new features or transforming the feature space to uncover dynamics [17, 29]. For instance, [30] proposed a feature extraction model based on Orthogonal Wavelet Transform. Another approach, introduced by [14], is the Temporal Multi-Features Representation Clustering (TMRC) framework, which extracts appropriate features from time series using a k-LSTM autoencoder. Similarly, [17] proposed a kernel-based algorithm that reduces data dimensions while maximizing the use of knowledge derived from local data distributions during feature extraction. Additionally, [31] introduced the Two-Level Time Series Clustering method to prevent information loss during dimensionality reduction. This method simultaneously utilizes information at two levels of abstraction, whole time series and subsequence time series, during clustering. [29] introduced DeLTa, a technique that transforms time series into 2D images and utilizes pre-trained CNNs for feature extraction. Subsequently, two extended methods, DeLTa-LA (Layout-Aligned) and DeLTa-LI (Layout-Independent), were proposed. DeLTa-LA emphasizes preserving spatial patterns, effectively capturing both local and global features. In contrast, DeLTa-LI focuses on shift and translation-invariant representations, achieved by applying feature pooling techniques to CNN-activated features [32].

Distance-based methods aim to provide suitable distance measures for time series, addressing specific characteristics such as noise, amplitude scaling, and longitudinal scaling to mitigate their adverse effects on clustering [33]. Some of the most well-known traditional methods for calculating distances include Euclidean Distance (ED), Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), Edit Distance on Real Sequences (EDR), and Edit Distance with Real Penalty [21]. Numerous distance metrics have been developed based on these methods, with DTW receiving the most attention due to its shift-invariance property [34]. For example, to address the issues of overstretching and over-compression in DTW, [35] introduced Adaptive Constrained DTW. Additionally, [36] proposed the DTW+S method, which extends DTW by capturing local trends through an interpretable matrix representation. Regarding LCSS, [37] proposed the DLCSS method to address the reliance on a single similarity threshold. DLCSS demonstrated significantly improved performance by incorporating two thresholds compared to the original LCSS.

Fuzzy-based methods, instead of rigidly assigning each sample to a single cluster, allow for the possibility of assigning samples to multiple clusters. For instance, [38] utilized fuzzy cognitive maps to identify and extract temporal features. Similarly, [19] effectively performed clustering in the presence of outliers by employing Partitioning Around Medoids (PAM) within a robust fuzzy framework. Additionally, [39] proposed a robust fuzzy C-medoids method based on entropy regularization and Dynamic Time Warping (DTW), which facilitates clustering under challenging conditions, such as noise, outliers, and varying time series lengths.

Model-based methods aim to adapt the clustering process for time series by improving traditional clustering algorithms or introducing innovative approaches. For example, [20] performed clustering using mixtures of linear Gaussian and state-space models based on probabilistic

frameworks. Similarly, [40] enhanced the robustness of the mixture of ARCH models for clustering by incorporating the normal mean-variance mixture. These represent examples of clustering methods rooted in probabilistic models. Other methods are based on machine learning, leveraging adaptations through the combination of various tools or improvements to existing algorithms. For instance, TCGAN, proposed in [6], is a method for classification and clustering that combines Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), demonstrating excellent performance. TSK-Means, introduced by [41], was designed and developed by defining a time-series-specific objective function. Additionally, [42] extended the K-means algorithm to enable the use of any dissimilarity measure, such as DTW, within the algorithm. This enhancement also allowed the application of K-means to datasets with missing data.

Some methods influence multiple phases of the clustering process, making it difficult to assign them to a single category. One of these methods is K-Shape, which is developed using normalized cross-correlation as a distance measure and centroid calculation in K-means [43]. Mapping time series data and their similarities into nodes and edges of a graph, followed by using graph-based tools for clustering, represents such hybrid methods. These approaches can be categorized as both feature-based and model-based methods. For example, [8] mapped time series clustering (TSC) into a complex network and performed community detection for clustering. Similarly, [44] proposed a five-step method based on complex networks and the normal cloud model, incorporating local features while reducing the impact of missing values. Another type of hybrid method is the shapelet-based approach. Shapelets are subsequences of time series that capture local patterns, representing specific classes [45]. These methods can primarily be associated with a combination of distance-based and model-based approaches. One such example is SE-Shapelets, introduced by [46], which performs semi-supervised time series clustering in two stages: (1) extracting candidate shapelets and (2) selecting from these candidates using the linear discriminant selection algorithm.

Following an in-depth review of existing research in the field of time series clustering, it is clear that although various dimensions influencing clustering outcomes have been explored and areas for improvement have been identified, notable research gaps persist. Addressing these gaps can lead to the development of more effective models. The key research gaps identified are as follows:

1. Existing studies often treat all fluctuations within a time series as part of a single unified pattern. However, independently analyzing the similarity of sub-patterns within a time series can potentially yield more accurate and insightful clustering results.
2. The majority of research focuses on addressing problem complexities through increasingly sophisticated solutions, while the strategy of simplifying the problem by systematically breaking down its complexity has been largely overlooked.
3. Limited attention has been given to exploring practical applications for STSC.
4. There is currently no established method for feature extraction that evaluates the similarity of time series to an exhaustive set of potential patterns.

To address these research gaps, we propose a novel approach to solving the WTSC problem as follows:

- A two-stage time series clustering method leveraging STSC, which reduces the complexity of existing patterns by decomposing a time series into a set of sub-patterns. This decomposition allows for the independent sub-pattern analysis, ultimately improving clustering performance.
- A feature extraction method that quantifies the similarity of a time series to a predefined set of base patterns, producing a set of shape-based features to enhance the clustering process.

This section comprehensively analyzes the problem, explores various solution categories, and presents a detailed review of the related works within each category, emphasizing key contributions and methodologies in this research domain. Finally, the identified research gaps are discussed, and the proposed solutions are introduced.

3 Method Description

This section provides an overview of the materials and the proposed method. To achieve this, given the application of hierarchical clustering on set-type data in the proposed

method, a brief explanation of its mechanism is first presented. Subsequently, each step of the SW-WTSC method is detailed to demonstrate how this novel approach effectively addresses the WTSC problem by maximizing the knowledge extracted from sub-patterns.

3.1 Hierarchical clustering

Hierarchical clustering organizes data into a tree-like structure called a dendrogram. This type of clustering is categorized into two approaches: agglomerative and divisive. In agglomerative methods, each data point is initially treated as an individual cluster, and clusters are gradually merged. In divisive methods, all data points start as part of a single cluster and are progressively split into smaller clusters. The merging or splitting decisions are based on a dissimilarity measure and a linkage strategy. Various linkage methods have been proposed, such as single, complete, and average. Single linkage considers the shortest distance between points in two clusters but is sensitive to noise and outliers. In contrast, complete linkage, which considers the farthest distance, tends to create compact clusters. Average linkage, on the other hand, calculates the mean distance between clusters and balances intra-cluster variance reduction with inter-cluster variance increase. Additionally, hierarchical clustering can be applied to sets using measures like Weighted Jaccard Distance (WJD), as defined in Equation (1). WJD quantifies dissimilarity between sets based on the frequency of shared elements [47].

$$WJD(S_i, S_j) = 1 - \frac{\sum_k Min(Count_k(S_i), Count_k(S_j))}{\sum_k Max(Count_k(S_i), Count_k(S_j))}$$

3.2 SW-WTSC Model

This section provides a comprehensive overview of the proposed method, the Sliding Window-based Whole Time Series Clustering (SW-WTSC) Model. The approach begins by extracting multiple sub-samples (sub-sequences) using a sliding window mechanism over the time series data (samples) to be clustered. These sub-samples are unified and fed into a K-Means clustering model. Subsequently, the labels of sub-samples corresponding to each original sample are aggregated into a label dataset. This dataset is then clustered using a set clustering model, ensuring that time series with the highest similarity in their sub-samples are grouped into the same cluster. An abstract representation of this method is illustrated in Figure 2, and the detailed implementation steps are provided in the following sections.

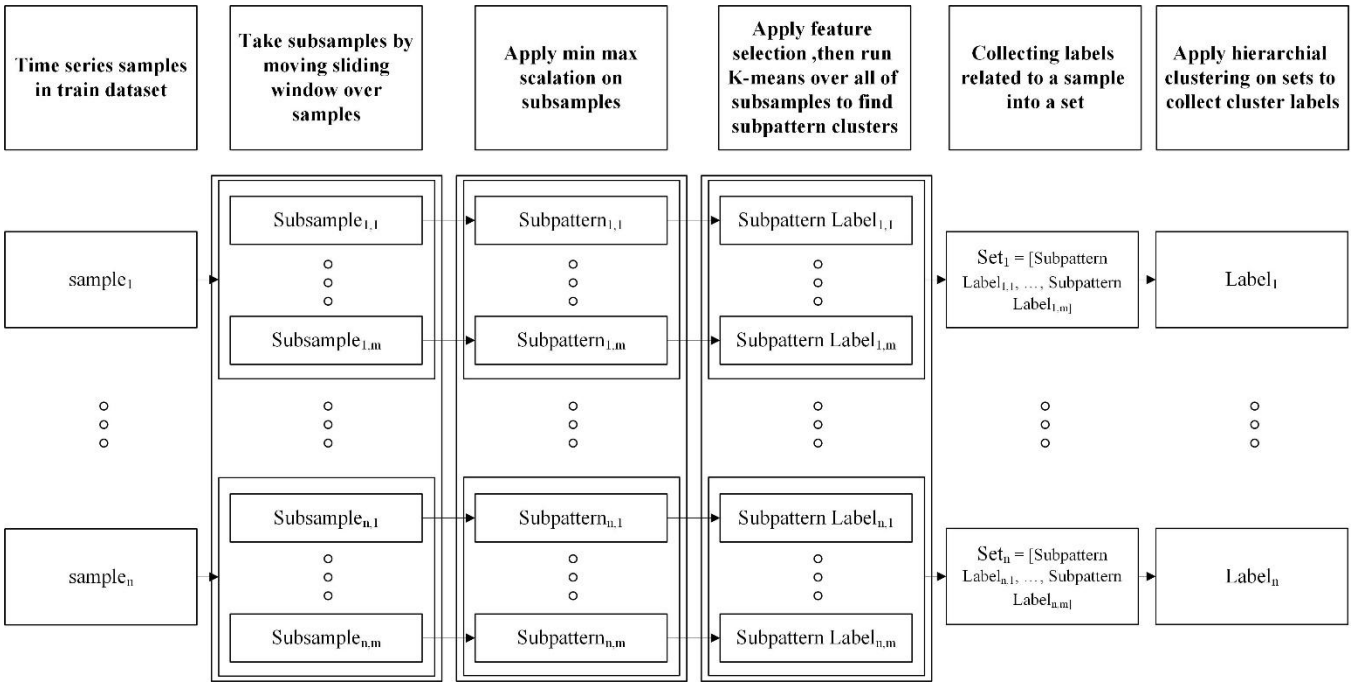

Figure 2 - Overview of the Sliding Window-based Whole Time Series Clustering (SW-WTSC) Model

3.2.1 Sample-Level Data Representation

Depending on the size of the time series and the level of detail it contains, it may be necessary to reduce the input data to minimize computational load and enable the algorithm to focus on an appropriate level of information. This step is not mandatory for all datasets and only applies to those where consecutive points predominantly consist of noise or provide minimal insight into the pattern's behavior. To achieve this, and with the goal of eliminating redundant information and

reducing computational complexity, this step transforms a time series of length L into a condensed time series of length L'.

To condense a time series, adjacent points are aggregated to generate representative points, shortening the series length. This reduction process inevitably discards some information from the original sample, which is no longer accessible in the simplified version. Figure 3 illustrates an example of this transformation.

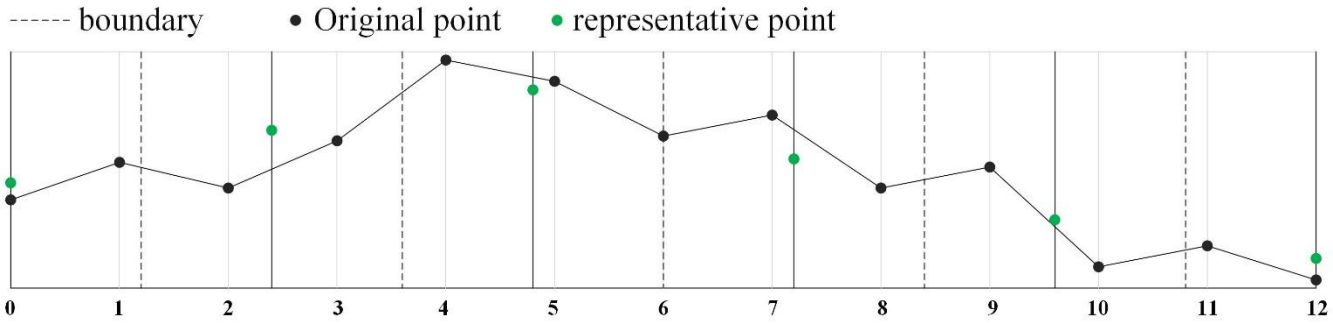

Figure 3 - representing a time series from a length of 13 to a length of 6

In the data representation process, the target length of the condensed time series is first determined. Based on this target length, the positions of the new representative points are strategically allocated among the original points. A boundary is then defined around each new point to aggregate data from the original series. These boundaries ensure equal distances from their center to their edges across all regions. To compute the values of the new representative points, the linear plot formed by the original data points is utilized to calculate the area under the curve within each defined boundary. The detailed steps for implementing this time series data representation method are outlined in Algorithm 1.

Algorithm 1: Time Series Representation (S, #O, #N)

Input: {S = Dataset of time series,
#O = Old size of time series,
#N = New size of time series}

Output: Represented time series dataset

1: Represented time series dataset = an empty table of size (#S×#N)

2: Half step = $\frac{(\#O-1)}{2\times(\#N-1)}$

3: Position of new Points = [(2×i×half step) for i in [0, …, #N]]

4: Boundaries (i) = {Lower: [max (i - half step, 0), Upper: min (i + half step, #O)] for i in Position of new Points}

5: **For** row (each sample) in S **do**:

6: PLF_{row} = create a piecewise linear function from original points and their values

7: **For** NP (each new point) in [0,…, #N], **do**:

8: NP value = $\int_{Boundaries\ (NP)[Lower]}^{Boundaries\ (NP)[Upper]} PLF_{ins}$

9: Represented time series dataset [row, NP] = NP value

10: **End for**

11: **End for**

12: **Return represented** time series dataset

3.2.2 *Sub-Sampling Using Sliding Window*

In this step, each sample is independently sub-sampled using a sliding window. The resulting sub-samples from a sample are aggregated into a matrix with dimensions $n(L-w+1)\times w$, where n is the number of training samples, L is their length, and w represents the length of the sub-sampled time series, which is known as the lookback period. Consequently, each sample is transformed into L-w+1 new sub-samples.

3.2.3 *Sub-Sample-Level Data Representation*

Similar to the process described in Section 3.2.1 for samples, the time Series Representation algorithm can also be applied to sub-samples at this stage. In that section, computational load reduction was achieved by shortening the sample length (L to L'), thereby reducing the number of sub-samples generated per sample in Section 3.2.2. However, the time series Representation algorithm at this stage contributes to computational efficiency in a different manner. Since the subsequent feature selection (which will be described in 3.2.5) has an exponential relationship with the lookback period, larger values of w can lead to

computational challenges. Additionally, a specific level of pattern abstraction within the sub-samples may be desired. Therefore, the time Series Representation algorithm is employed to address these issues in this step.

3.2.4 Sub-Sample Pattern Extraction

This stage involves standardizing the scale of sub-samples by applying the Min-Max Scaler function to each sub-sample individually. Using the formula presented in Equation (2), this function scales the values of each sub-sample to lie between 0 and 1. As a result, patterns that are structurally similar but differ in the numerical scale of their time series values become comparable. All such patterns represent the underlying shape of the time series at a standardized scale between 0 and 1.

$$Z_i = \frac{X_i - Min(X_1,...,X_w)}{Max(X_1,...,X_w) - Min(X_1,...,X_w)}$$

In this equation, X_i represents the i-th value in time series, which is transformed into its standardized value, denoted by Z_i, using the corresponding scaling function.

This standardization enhances the algorithm's ability to detect similar patterns, ensuring that clustering is based on the shape of the patterns rather than the magnitude of the time series values. Moreover, using Min-Max scaling instead of logarithmic scaling helps preserve the original shape of the sub-patterns. Consequently, clustering models focus on structural similarities, improving their overall performance in pattern recognition.

3.2.5 Applying Pattern-Based Feature Selection

This step is designed to transform the data space so that dimensions no longer represent the position of points in a pattern but instead indicate the similarity to predefined patterns. Center-based clustering models assign clusters based on the proximity of points to candidate cluster centers. This means that when time series data are used as inputs, the cluster centers are time series themselves. The type of input data dictates the nature of cluster centers.

The steps performed up to this stage transformed the data dimensions into forms that show the relative positions of the pattern's variables. This step aims to enhance clustering performance by converting these dimensions into ones that reflect a pattern's similarity to a set of predefined patterns. By doing so, the clustering model is better equipped to allocate patterns to appropriate clusters. At this stage, a series of transformation functions can be applied to define new dimensions that measure the similarity of a time series to specific patterns.

Two key considerations are necessary to construct these transformation functions: **I)** Selection of Base Patterns and **II)** Similarity Measurement. A structured framework is required to define base patterns. Without such a framework, minor variations in the parameters defining patterns could yield infinitely many patterns. Standardizing and constraining the arrangement of variables can generate a finite number of patterns. This ensures a balance between capturing adequate detail in patterns and limiting the number of possibilities.

Each pattern is defined by two components: I) Pattern Length and II) Number of candidate positions, which is the possible value for each point in the pattern. An illustrative example of this Framework is shown in Figure 4.

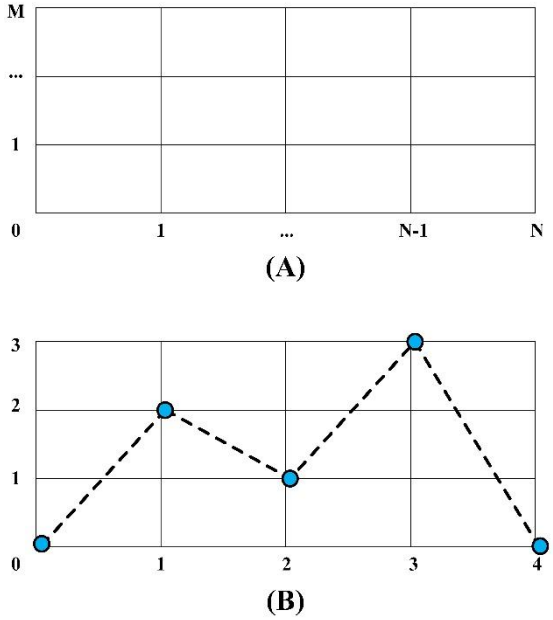

Figure 4 - A Structure for Defining Time Series Patterns

In Figure 4, N represents the pattern length, and M indicates the number of candidate values for each point. By limiting the positions of points, the total number of possible patterns can be computed using the Equation (3):

$$\#Possible\ Base\ Patterns = M^N - \binom{2}{1}(M-1)^N + (M-2)^N$$

Also here, N is the pattern length, and M indicates the number of candidate values for each point. Given the exponential nature of this relationship, selecting large values for N and M results in an unmanageable number of patterns. So, in this algorithm, M is set to 2, limiting each variable of the pattern to values of 0 or 1. Consequently, the total number of possible patterns is as Equation (4):

$$\#Possible\ Base\ Patterns = 2^N - 2 \quad (4)$$

To compute the dissimilarity, the deviation of the pattern from a base pattern is evaluated, So the Manhattan distance between a pattern like $X = (x_1, \dots, x_w)$ and a base pattern like $A_j = (a_{1,j}, \dots, a_{w,j})$ can be used, as shown in Equation (5):

$$D(X, A_j) = \sum_{i=0}^{w} \left| (x_i - a_{i,j}) \right| \quad (5)$$

Here, A_j is a base pattern of length w and $D(X, A_j)$ represents the deviation of pattern X from A_j, which can serve as a new dimension. Since the constructor variables of base patterns in this algorithm are limited to 0 and 1, the Equation (6) can be reformulated as follows:

$$D(X, A_j) = \sum_{i}^{a_{i,j}=0} x_i + \sum_{i} a_{i,j} - \sum_{i}^{a_{i,j}=1} x_i \quad (6)$$

Euclidean distance is often used to assign points to candidate clusters in the training process of center-based clustering models. This type of distance is computed by assessing the difference between the values of two points across each dimension, as shown in Equation (7):

$$Euclidean\ Distance = \sqrt{(\Delta_1)^2 + \dots + (\Delta_w)^2} \quad (7)$$

Where Δ_i represents the distance between two points in dimension *i*. Since the term $\sum_i a_{i,j}$ of Equation (6) remains constant across all samples within a specific dimension, it does not influence Δ_i, leading to the simplification of Equation (6) to Equation (8).

$$D(X, A_j) = \sum_{i}^{a_{i,j}=0} x_i - \sum_{i}^{a_{i,j}=1} x_i \quad (8)$$

If, according to Equation (4), a total of $2^N - 2$ possible patterns are generated, and the corresponding feature selection process is performed using Equation (8), half of the resulting values will be symmetric to each other. This symmetry arises because their corresponding base patterns are mirrored across the horizontal axis. Consequently, according to Equation (8), the values of the newly derived features will be pairwise symmetric.

Given this linear dependency between features, it is necessary to eliminate half of them to avoid redundancy. Therefore, it can be concluded that the base patterns can be constructed according to Equation (9), ensuring a balanced and optimized feature extraction.

$$A_{j \in \{1,\dots,\frac{\#PBP}{2}\}} = (a_{1,j}, \dots, a_{w,j}) = \begin{cases} a_{i,j} = 0 & i = w \\ a_{i,j} = ith\ value\ of\ \ j\ in\ binary\ form & i \le w \end{cases} \quad (9)$$

Where *#PBP* represents the total number of possible base patterns, *w* denotes the length of the base patterns, *j* is the index of the pattern, and $a_{i,j}$ indicates the value of the *i*-th component of the *j*-th base pattern. After generating the base patterns, the corresponding features can be calculated using Equation (9) to be considered as dimensions of the newly created data space.

3.2.6 Training the K-Means Clustering Model

The data obtained from the previous step is utilized to cluster the sub-samples using the k-means algorithm. Since k-means do not automatically determine the number of clusters, it is necessary to specify this parameter. A more significant number of clusters enables the algorithm to make finer distinctions between patterns, identifying even the most negligible differences. Conversely, setting a smaller number of clusters results in grouping broadly similar patterns while paying less attention to finer details. This flexibility can serve as a tool for adjusting the emphasis on pattern details during clustering. So, the choice of the number of clusters in this step depends on the desired level of detail to be captured in forming clusters. At this stage, all sub-samples are clustered together in a single process to ensure that sub-samples with similar patterns, even if they originate from different samples, are assigned the same cluster label.

3.2.7 Aggregating Sub-Pattern Clusters' Labels into Sets

Two time series are considered similar when they exhibit comparable patterns. This means that if the overall patterns of the two samples are identical, their constituent sub-patterns are also expected to share significant similarities. These similarities were identified through the clustering operation in section 3.2.6, where similar sub-patterns were grouped. After assigning cluster labels to the sub-patterns, the cluster labels of the sub-patterns belonging to a sample can be collected into a set. Thus, each sample is represented by an equivalent set that reflects its sub-patterns. In Figure 5, a schematic example of the outcomes from Steps 3.2.6 and 3.2.7 is illustrated.

Based on Figure 5, the sets of sub-pattern cluster labels for the four patterns are as follows: $Labels_A$ = {1, 2, 3, 4, 6}, $Labels_B$ = {1, 2, 3, 4, 6}, $Labels_C$ = {1, 2, 2, 5, 7}, $Labels_D$ = {5, 8, 9, 10, 11}. Among the four patterns, patterns A and B exhibit the highest similarity. This similarity is also evident when examining the sets of sub-pattern clusters. Essentially,

the steps executed up to this point aim to encode patterns into a collection of labels. Among these patterns, A and B share five standard labels, making them the most similar. Following them, C has two labels in common with A and B, indicating a moderate level of similarity. In contrast, D shares only one sub-pattern with C and has no common sub-patterns with A and B. Therefore, it can be concluded that pattern similarities remain observable in the resulting sets.

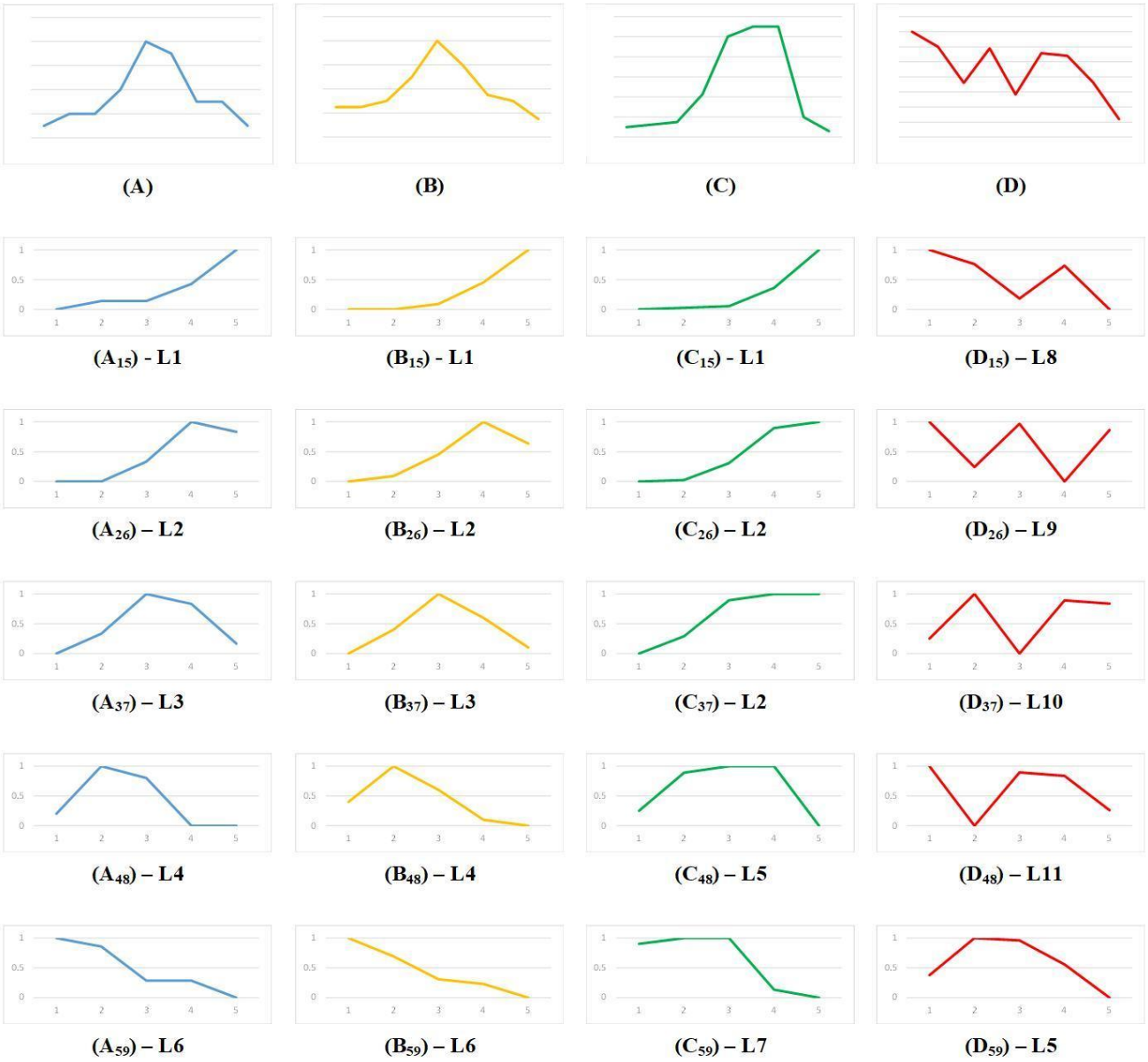

Figure 5 - Execution of Steps 3.2.6 and 3.2.7 on the four sample patterns A, B, C, and D involves generating sub-patterns, clustering them, and preparing sets containing the sub-pattern cluster labels. For instance, (A59) - L6 represents a sub-pattern extracted from sample A in the range [5, 9], which, after clustering, has been assigned the label 6.

3.2.8 Training the Hierarchical Clustering Model on Sets

In the final step, the sample sets are re-clustered using a hierarchical clustering algorithm described in section 3.1. This algorithm groups the sets based on the similarity of their elements while also considering the frequency of repeated elements. As a result, it can place samples with high similarity, based on their label sets, into the same cluster.

The use of hierarchical clustering in this section is motivated by the fact that the optimal number of clusters is not known in advance. Hierarchical clustering helps address this by capturing the global structure of the data and providing a dendrogram, which improves interpretability and facilitates the identification of an appropriate number of clusters. Moreover, compared to methods like K-Means, which assume spherical cluster shapes, hierarchical

clustering—by gradually merging samples—can better capture complex and non-linear cluster structures.

Complete linkage was selected because the final representation of each time series is based on aggregated sub-pattern cluster memberships. In this structured representation space, maximizing inter-cluster separation improves cluster compactness and reduces chaining effects that may arise with single linkage. Preliminary comparisons with alternative linkage strategies indicated that complete linkage provided more stable and well-separated clustering results. It helps prevent risks such as the chaining effect, which is more likely to occur with a single linkage. These reasons justify the choice of hierarchical clustering over other methods, such as DBSCAN or spectral clustering, which may perform better in handling density-based structures but lack the interpretability and flexibility offered by hierarchical clustering.

Finally, the clustering quality must be evaluated to ensure alignment with expectations. If the results do not meet the desired criteria, the algorithm should be re-executed with different parameters. This process is repeated with various parameter combinations until satisfactory results are achieved. The definition of a "satisfactory" result can vary from person to person or depend on the specific use case. Unlike classification problems, clustering lacks predefined labels, making it necessary to evaluate the appropriateness of the assigned labels post-clustering to determine their alignment with the intended objectives. The following pseudocode provides additional details on the SW-WTSC algorithm:

Algorithm 2: SW-WTSC (S, #FS, #L, #AS, #SC, #IC)

Input: {S = Dataset of time series,
#FS = Sample-Level Data representation size,
#L = Lookback period size,
#AS = Sub-Sample-Level Data Representation size,
#SC = Number of Clusters in sub-instances clustering,
#IC = Number of Clusters in instances clustering}

Output: SW-WTSC model

1: S = Condense S by Time Series Representation algorithm to size #FS (Optional step)

2: Aggregated subsequence time series table (ASTT) = an empty table

3: **For** row (each sample) in S **do**:

4: apply sliding window with size #L on each row of S and add subsequence to ASTT

5: **End for**

6: SASTT = Condense ASTT by Time Series Representation algorithm to size #AS

7: **For** each row (each sample) in SASTT **do**:

8: Normalize each row of SASTT by Min Max Scaler and add to NSASTT

9: **End for**

10: Cluster Input Table (CIT) = Extract shape features from NSASTT

11: Trained first clustering model (FCM) = Train K-means with CIT to #SC clusters

12: Aggregated Label Dictionary (ALD) = {}

13: **For** row (each sample) in S **do:**

14: ALD [row] = Aggregate the labels of the sub-instances generated by FCM, which were related to the instance row as a list

15: **End for**

16: Trained second clustering model (SCM) = Train Hierarchical clustering model with ALD to #IC clusters

17: SW-WTSC = (FCM, SCM)

18: **return** SW-WTSC model

Since the SW-WTSC algorithm employs both K-Means and hierarchical clustering with complete linkage, its computational complexity includes the complexity of both algorithms and other computational costs associated with different stages of the method. If the original samples are not represented before clustering, the algorithm will have the $O(nT^2) + O(nTK2^s) + O(n^2k') + O(n^2k)$ complexity. However, if the original samples are represented in the first stage, the computational complexity will be converted to $O(nT) + O(nT'^2) + O(nT'K2^s) + O(n^2k') + O(n^2k)$ where n is the number of training samples, T is their length, T' is the length of samples after representation, w is the sliding window length for subsequence extraction, s is the length of subsequences after representation, k is the number of clusters in the K-Means algorithm, and k' is the number of clusters in the final clustering stage.

Among these factors, n cannot be reduced since removing training samples is undesirable. Also, k and k' depend on the intrinsic structure of the. Therefore, the algorithm is designed to control the complexity of other parameters. If T is significantly larger than the other variables, the complexity without representation in the first stage would be $O(nT^2)$. To address this, a representation step is introduced at the beginning of the algorithm to reduce computational costs. Additionally, since s can exponentially increase complexity, its growth is controlled through the

representation of subsequences, ensuring that computational efficiency is maintained.

In this section, we comprehensively explained the algorithms utilized in our proposed method. The following section will delve into the experimental setup and numerical evaluations to assess the effectiveness of the proposed approach.

4 Experimental Results and Discussion

To verify the capability of SW-WTSC in performing whole time series clustering, this method has been applied to 85 datasets from the UCR Time Series Archive. Subsequently, the results obtained were compared with a set of state-of-the-art methods to clarify the impact of using this approach. Finally, this section concludes with a detailed analysis of the strengths and limitations, offering a comprehensive discussion of the findings.

4.1 Data

To address challenges arising from generalizing method results on a single time series, the UCR Time Series Archive was created by [47], which has been cited by hundreds of researchers for evaluation purposes. This archive, containing 85 datasets, provides a comprehensive diversity of behaviors in time series, such as noise, shift, longitudinal scaling, and more. This makes it an ideal tool for assessing the performance of methods under various conditions.

4.2 Evaluation metrics

A variety of measures have been introduced to evaluate the performance of methods proposed for time series clustering. When class labels for each sample are pre-existing, clustering can be performed without considering the labels, and the results can then be compared to the pre-defined labels. In such cases, measures known as External Indexes are used. Among these, Normalized Mutual Information (NMI) stands out because its normalized nature allows for comparing different methods with varying numbers of clusters, unlike other measures. The following equation (10) demonstrates how NMI is calculated.

$$NMI(A,B) = \frac{I(A,B)}{\left(\frac{H(A)+H(B)}{2}\right)} \quad (10)$$

In this equation, I represent the Mutual Information, and H denotes the Entropy.

4.3 Non-parametric Statistical Analysis Method

The Friedman test is a statistical tool used to compare the performance of multiple methods. This non-parametric test operates based on ranking the methods within each experimental block and tests the null hypothesis that all models perform equally. In this section, the Friedman test is applied to compare the performance of SW-WTSC against other state-of-the-art methods. The experimental blocks consist of datasets from the UCR archive, and the treatments are the time series clustering methods. Following the execution of the Friedman test, a post hoc analysis was conducted, adjusting the significance level for multiple comparisons. By comparing the obtained p-values with the significance level, it is determined whether the null hypothesis is rejected or not.

4.4 Algorithm parameters

In implementing the SW-WTSC algorithm, several parameters need to be configured, as their values significantly impact the final results. As discussed in Section 2, evaluating the quality of clusters obtained from the clustering process is a passive task after results are achieved. Consequently, no systematic method exists for tuning the parameters of SW-WTSC, and they must be adjusted through trial and error. Each parameter influences an aspect of clustering and can alter the results obtained.

The first parameter is the New Size of samples, representing the samples with a new, shorter size, thereby reducing their length. The smaller this parameter, the more information is considered noise and removed. The following parameter is the Look-back Period in the sliding window. A higher value for this parameter means that longer sub-patterns are analyzed, considering the relative position of points over more extended periods for clustering. This parameter determines the independence of points separated by distances greater than its value.

Another parameter is the Sub-sample Representation length, which, similar to the samples, simplifies and removes noise from sub-samples. The following parameter concerns the Number of Clusters in the Initial Clustering (STSC). Fewer clusters lead to less similar samples being grouped, while more clusters result in highly identical sub-patterns being grouped. This parameter is adjusted depending on the level of similarity required for grouping in a cluster. The final parameter is the Number of Clusters in WTSC, which,

in the experiment conducted, was provided by the UCR datasets.

The dataset samples were first analyzed to optimize the model parameters, and a set of candidate values for the sliding window size was selected to ensure the generation of meaningful sub-patterns. The goal was to facilitate the differentiation of patterns through these sub-patterns. Candidate values of 5 and 10 were considered for selecting the Subsample Representation Length. Values exceeding 10 significantly increased computational complexity, while those below five overly simplified the sub-patterns. Since the simplification ratio of 5 is twice that of 10, intermediate values were not examined to reduce the candidate parameter space. Additionally, 50, 100, and 200 candidate values were chosen for the number of K-Means clusters to establish an appropriate range for this parameter. Following the first clustering stage, the parameters were refined based on the formation of clusters containing similar sub-patterns. Further adjustments were made as needed to improve model performance. The final values are included in Appendix 1.

4.5 Experiment Setup

To verify the effectiveness of SW-WTSC in performing Whole Time Series Clustering, the method was applied to 85 datasets from the UCR archive, and the NMI index was calculated to compare its performance with state-of-the-art methods. By introducing improvements across various stages of the clustering process, SW-WTSC can be categorized as model-based, feature-based, and representation-based. Accordingly, the comparison was conducted using a diverse set of methods, including DeLTa-LA, DeLTa-LI, SPIRAL, K-means ED, DWT, SAX, TCGAN, and K-shape. To ensure the reliability of the results, SW-WTSC was executed five times on each dataset, and the average results were considered as the final performance of the model. Subsequently, the Friedman test was performed to statistically determine the relative performance of the methods. The results obtained from applying the various techniques to the datasets are presented in Table 1. Additionally, Appendix 1 provides information on the execution time of the algorithm across different datasets examined in this study. The algorithm was developed in Python and executed on a system equipped with six threads of an AMD EPYC 7763 2.5 GHz CPU and 64 GB of RAM.

Table 1 - Performance Comparison of SW-WTSC and State-of-the-Art Methods on 85 UCR Datasets Using the NMI Index

Datasets	DeLTa LA	DeLTa LI	SPIRAL	K-means ED	K-means DWT	K-means SAX	k-means TCGAN	k-shape	SW-WTSC
50words	0.710	0.550	0.730	0.660	0.680	0.680	0.713	0.664	0.696
Adiac	0.710	0.730	0.360	0.600	0.610	0.500	0.683	0.503	0.598
ArrowHead	0.470	0.370	0.470	0.470	0.470	0.380	0.298	0.260	0.366
Beef	0.360	0.250	0.230	0.230	0.230	0.350	0.258	0.288	0.319
BeetleFly	0.030	0.030	0.010	0.000	0.000	0.010	0.067	0.024	0.439
BirdChicken	0.080	1.000	0.080	0.300	0.300	0.300	0.002	0.108	0.597
Car	0.390	0.270	0.180	0.280	0.280	0.230	0.342	0.240	0.615
CBF	0.660	0.050	0.420	0.460	0.310	0.310	0.929	0.746	1.000
ChlorineConcentration	0.010	0.010	0.010	0.010	0.010	0.010	0.074	0.000	0.068
CinC_ECG_torso	0.290	0.250	0.500	0.480	0.480	0.280	0.959	0.103	0.147
Coffee	0.100	0.160	0.700	0.810	0.810	0.210	0.736	0.177	0.780
Computers	0.020	0.070	0.060	0.000	0.000	0.000	0.018	0.037	0.106
Cricket_X	0.410	0.390	0.330	0.270	0.270	0.300	0.352	0.359	0.390
Cricket_Y	0.360	0.380	0.360	0.280	0.290	0.310	0.374	0.388	0.435
Cricket_Z	0.420	0.400	0.320	0.290	0.270	0.290	0.333	0.369	0.410
DiatomSizeReduction	0.870	0.740	0.640	1.000	1.000	0.910	0.700	0.802	0.715
DistalPhalanxOutlineAgeGroup	0.330	0.230	0.310	0.190	0.190	0.180	0.429	0.394	0.450
DistalPhalanxOutlineCorrect	0.010	0.000	0.010	0.030	0.030	0.030	0.003	0.003	0.042
DistalPhalanxTW	0.500	0.400	0.520	0.460	0.460	0.360	0.565	0.472	0.612
Earthquakes	0.050	0.070	0.060	0.000	0.010	0.000	0.042	0.005	0.053
ECG200	0.170	0.220	0.140	0.140	0.140	0.130	0.273	0.133	0.203
ECG5000	0.530	0.530	0.570	0.550	0.540	0.550	0.513	0.480	0.612
ECGFiveDays	0.160	0.090	0.180	0.000	0.000	0.090	0.024	0.705	0.534

ElectricDevices	0.310	0.280	0.370	0.210	0.190	0.210	0.306	0.260	0.337
FaceAll	0.710	0.300	0.700	0.420	0.410	0.450	0.434	0.538	0.663
FaceFour	0.450	0.540	0.600	0.490	0.490	0.480	0.520	0.475	1.000
FacesUCR	0.690	0.410	0.690	0.480	0.470	0.420	0.703	0.595	0.718
Fish	0.660	0.620	0.230	0.350	0.320	0.280	0.558	0.308	0.686
FordA	0.000	0.000	0.000	0.000	0.000	0.000	0.248	0.105	0.077
FordB	0.000	0.020	0.000	0.000	0.000	0.000	0.304	0.022	0.084
Gun_Point	0.010	0.020	0.000	0.010	0.010	0.010	0.000	0.008	0.171
Ham	0.010	0.010	0.020	0.010	0.010	0.600	0.010	0.050	0.100
HandOutlines	0.080	0.050	0.220	0.210	0.210	0.140	0.098	0.178	0.149
Haptics	0.130	0.090	0.100	0.150	0.150	0.140	0.042	0.088	0.083
Herring	0.000	0.010	0.000	0.000	0.000	0.000	0.008	0.009	0.113
InlineSkate	0.210	0.160	0.200	0.210	0.210	0.190	0.040	0.089	0.096
InsectWingbeatSound	0.560	0.320	0.580	0.550	0.570	0.570	0.510	0.453	0.493
ItalyPowerDemand	0.000	0.040	0.030	0.020	0.050	0.030	0.048	0.210	0.428
LargeKitchenAppliances	0.070	0.040	0.070	0.040	0.040	0.050	0.043	0.158	0.031
Lighting2	0.000	0.060	0.020	0.010	0.010	0.030	0.115	0.103	0.267
Lighting7	0.520	0.410	0.440	0.440	0.480	0.510	0.507	0.524	0.478
Mallat	0.840	0.780	0.920	0.880	0.880	0.880	0.691	0.830	0.899
Meat	0.670	0.610	0.620	0.670	0.670	0.680	0.550	0.567	0.708
MedicalImages	0.330	0.250	0.310	0.240	0.250	0.220	0.293	0.206	0.227
MiddlePhalanxOutlineAgeGroup	0.110	0.030	0.100	0.020	0.020	0.030	0.395	0.336	0.481
MiddlePhalanxOutlineCorrect	0.040	0.020	0.060	0.000	0.000	0.020	0.002	0.007	0.025
MiddlePhalanxTW	0.420	0.390	0.410	0.420	0.420	0.410	0.428	0.414	0.482
MoteStrain	0.510	0.120	0.280	0.130	0.190	0.220	0.006	0.301	0.297
NonInvasiveFatalECG_Thorax1	0.750	0.590	0.600	0.720	0.730	0.590	0.811	0.671	0.564
NonInvasiveFatalECG_Thorax2	0.840	0.770	0.680	0.780	0.770	0.660	0.846	0.711	0.664
OliveOil	0.400	0.340	0.810	0.750	0.750	0.470	0.734	0.503	0.716
OSULeaf	0.280	0.560	0.300	0.240	0.210	0.240	0.271	0.369	0.435
PhalangesOutlinesCorrect	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.014	0.014
Phoneme	0.520	0.530	0.520	0.510	0.440	0.400	0.322	0.221	0.222
Plane	1.000	1.000	0.920	0.900	0.900	0.900	0.916	0.851	0.978
ProximalPhalanxOutlineAgeGroup	0.570	0.550	0.580	0.500	0.500	0.430	0.532	0.437	0.575
ProximalPhalanxOutlineCorrect	0.030	0.030	0.050	0.060	0.060	0.080	0.096	0.092	0.129
ProximalPhalanxTW	0.550	0.500	0.560	0.510	0.500	0.460	0.579	0.507	0.567
RefrigerationDevices	0.110	0.050	0.140	0.010	0.000	0.010	0.024	0.005	0.027
ScreenType	0.020	0.010	0.020	0.030	0.030	0.030	0.006	0.012	0.095
ShapeletSim	0.280	0.070	0.100	0.010	0.010	0.010	0.004	0.390	0.182
ShapesAll	0.790	0.760	0.710	0.730	0.720	0.730	0.719	0.745	0.728
SmallKitchenAppliances	0.080	0.120	0.120	0.020	0.010	0.040	0.084	0.032	0.240
SonyAIBORobotSurface	0.220	0.100	0.010	0.020	0.030	0.030	0.613	0.193	0.409
SonyAIBORobotSurfaceII	0.380	0.000	0.380	0.300	0.300	0.380	0.241	0.066	0.344
StarLightCurves	0.600	0.660	0.610	0.610	0.610	0.610	0.644	0.618	0.628
Strawberry	0.080	0.130	0.100	0.090	0.090	0.110	0.145	0.102	0.163
SwedishLeaf	0.740	0.740	0.550	0.540	0.570	0.490	0.723	0.523	0.609
Symbols	0.910	0.950	0.710	0.820	0.820	0.820	0.826	0.770	0.895
synthetic_control	0.980	0.560	0.820	0.780	0.790	0.790	0.804	0.707	0.748
ToeSegmentation1	0.030	0.550	0.010	0.000	0.000	0.000	0.003	0.006	0.492
ToeSegmentation2	0.070	0.530	0.000	0.010	0.010	0.000	0.034	0.090	0.575
Trace	0.740	1.000	0.500	0.510	0.510	0.510	0.502	0.684	0.955
Two_Patterns	0.000	0.010	0.110	0.020	0.020	0.020	0.001	0.350	0.357
TwoLeadECG	0.090	0.180	0.070	0.090	0.090	0.070	0.002	0.078	0.503
uWaveGestureLibrary_X	0.460	0.300	0.460	0.450	0.440	0.440	0.489	0.466	0.241
uWaveGestureLibrary_Y	0.390	0.200	0.460	0.460	0.460	0.440	0.442	0.347	0.190
uWaveGestureLibrary_Z	0.460	0.260	0.450	0.440	0.450	0.440	0.474	0.443	0.276
UWaveGestureLibraryAll	0.790	0.260	0.690	0.680	0.680	0.650	0.702	0.656	0.348
Wafer	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.039
Wine	0.000	0.010	0.000	0.000	0.000	0.100	0.030	0.017	0.089

WordsSynonyms	0.590	0.440	0.530	0.500	0.500	0.520	0.549	0.464	0.478
Worms	0.160	0.410	0.170	0.100	0.080	0.170	0.264	0.086	0.189
WormsTwoClass	0.000	0.040	0.020	0.000	0.000	0.000	0.023	0.003	0.067
Yoga	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.005
overall	0.340	0.306	0.317	0.302	0.300	0.290	0.341	0.310	0.400

5 Results for Experiment

To perform the Friedman test, it is essential to define the statistical hypotheses and determine the significance level of the obtained results. The hypotheses and the significance level of 5% ($\alpha = 0.05$) are as follows:

$$\begin{cases} \mathbf{H_0}: \text{The selected TSC method does not affect the NMI index} \\ \mathbf{H_1}: \text{The selected TSC method affects the NMI index} \end{cases}$$

The results of the test are presented in Table 2. Based on the calculated Average of Ranks, SW-WTSC performs best and ranks first, followed by DeLTa-LA and TCGAN in second and third places, respectively. The calculated p-values from the comparison of each method with SW-WTSC indicate that the null hypothesis (H_0) is rejected, as these values are less than the Holm-adjusted significance level. For instance, in the first comparison between DeLTa-LA and SW-WTSC, the p-value of 0.01729 is less than 0.05, leading to the rejection of H_0. Similarly, other methods were also outperformed by SW-WTSC based on the same criteria.

Table 2 - Friedman Test Results and Performance Ranking of Clustering Methods

Model	Avg of ranks	in compare with ICSPMH (GWO-SVC)	
		P-value	holm
SW-WTSC	3.28235	1.00000	-
DeLTa -LA	4.28235	0.01729	0.05000
K-means TCGAN	4.46471	0.00488	0.02500
SPIRAL	4.81765	0.00026	0.01667
DeLTa-LI	5.21176	0.00000	0.01250
K-shape	5.34118	0.00000	0.01000
K-means DWT	5.80588	0.00000	0.00833
K-means ED	5.83529	0.00000	0.00714
K-means SAX	5.95882	0.00000	0.00625

The average rank in SW-WTSC is significantly lower than that of the second-ranked method. However, the average ranks in the second, third, and fourth methods are very close to one another, indicating that these methods exhibit varying performance across different datasets. This variability could be attributed to the dependence of these methods on specific behavioral features of time series. In contrast, SW-WTSC performs consistently well across a broader range of datasets and is less sensitive to diverse time series behaviors. This consistency allows SW-WTSC to achieve a better average ranking than its competitors.

The results of applying the methods to the datasets are illustrated in Figure 6 using box plots, which provide a clear visualization of the performance of the models. SW-WTSC demonstrates a distinct advantage over other methods in Q1, Q2, and Q3, indicating that its performance is effectively improved across many datasets. In contrast, none of the other methods achieve dominance across all three quartiles relative to their competitors. Additionally, the upper whisker for SW-WTSC reaches the maximum possible value of 1, representing the highest NMI, corresponding to perfect clustering. This further underscores SW-WTSC's superiority over the four competing methods. Another noteworthy observation is related to the interquartile range (IQR). Comparing the IQRs of the top three performing methods, it becomes evident that SW-WTSC exhibits a significantly more compact range. This compactness indicates more stability in its performance compared to TCGAN and DeLTa-LA. This observation also highlights the reduced influence of diverse time series behaviors on SW-WTSC, reinforcing its consistent performance compared to its competitors.

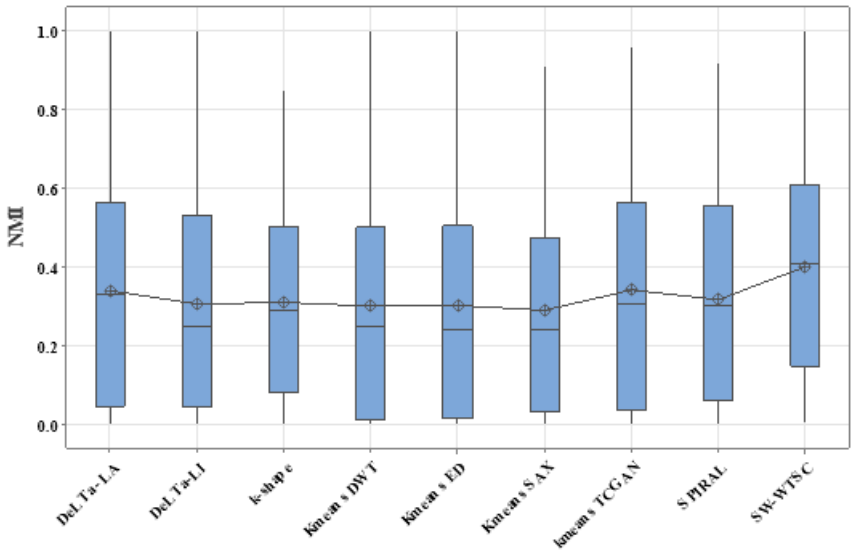

Figure 6 - Box Plot Comparison of Clustering Methods Across Datasets

Assuming that time series exhibit diverse behaviors and each method can capture some of these characteristics, Table 3 has been presented for a more detailed analysis. Each row in this table summarizes the results of datasets where a specific method has consistently outperformed others. The name of the best-performing method has been used to label the corresponding category of datasets, which is displayed in the first column. The subsequent columns show the average NMI achieved by different methods in each category, and the final column indicates the number of datasets in each category.

Table 3 - Performance Breakdown of Clustering Methods Across Dataset Categories

Category (Method with rank 1)	Method										
	DeLTa LA	DeLTa LI	SPIRAL	K-means ED	K-means DWT	K-means SAX	K-means TCGAN	K-shape	SW-WTSC	Avg Rank	Count
DeLTa-LA	0.58	0.40	0.50	0.45	0.45	0.46	0.44	0.43	0.47	4.93	15
DeLTa-LI	0.41	0.65	0.33	0.37	0.36	0.36	0.36	0.34	0.51	3.40	10
SPIRAL	0.40	0.31	0.49	0.42	0.42	0.39	0.41	0.37	0.41	4.30	10
K-means ED	0.37	0.33	0.48	0.65	0.65	0.42	0.49	0.36	0.53	6.00	3
K-means SAX	0.01	0.01	0.01	0.01	0.01	0.35	0.02	0.03	0.09	2.00	2
K-means TCGAN	0.34	0.27	0.31	0.32	0.32	0.28	0.52	0.31	0.30	5.09	11
K-shape	0.26	0.15	0.20	0.12	0.13	0.17	0.14	0.44	0.31	5.00	4
SW-WTSC	0.21	0.20	0.19	0.17	0.17	0.16	0.24	0.22	0.39	1.00	31

Since DWT did not outperform other methods in any dataset, no row in the table has been allocated to it. SW-WTSC, with 31 top rankings among 85 datasets, encompasses the most significant number of datasets in its category, followed by the DeLTa-LI category with 15 datasets. To provide deeper insights into the values presented in this table and to enhance visual interpretation, these values have been depicted in Figure 7. Each line in the figure represents the performance of the methods on a specific dataset category.

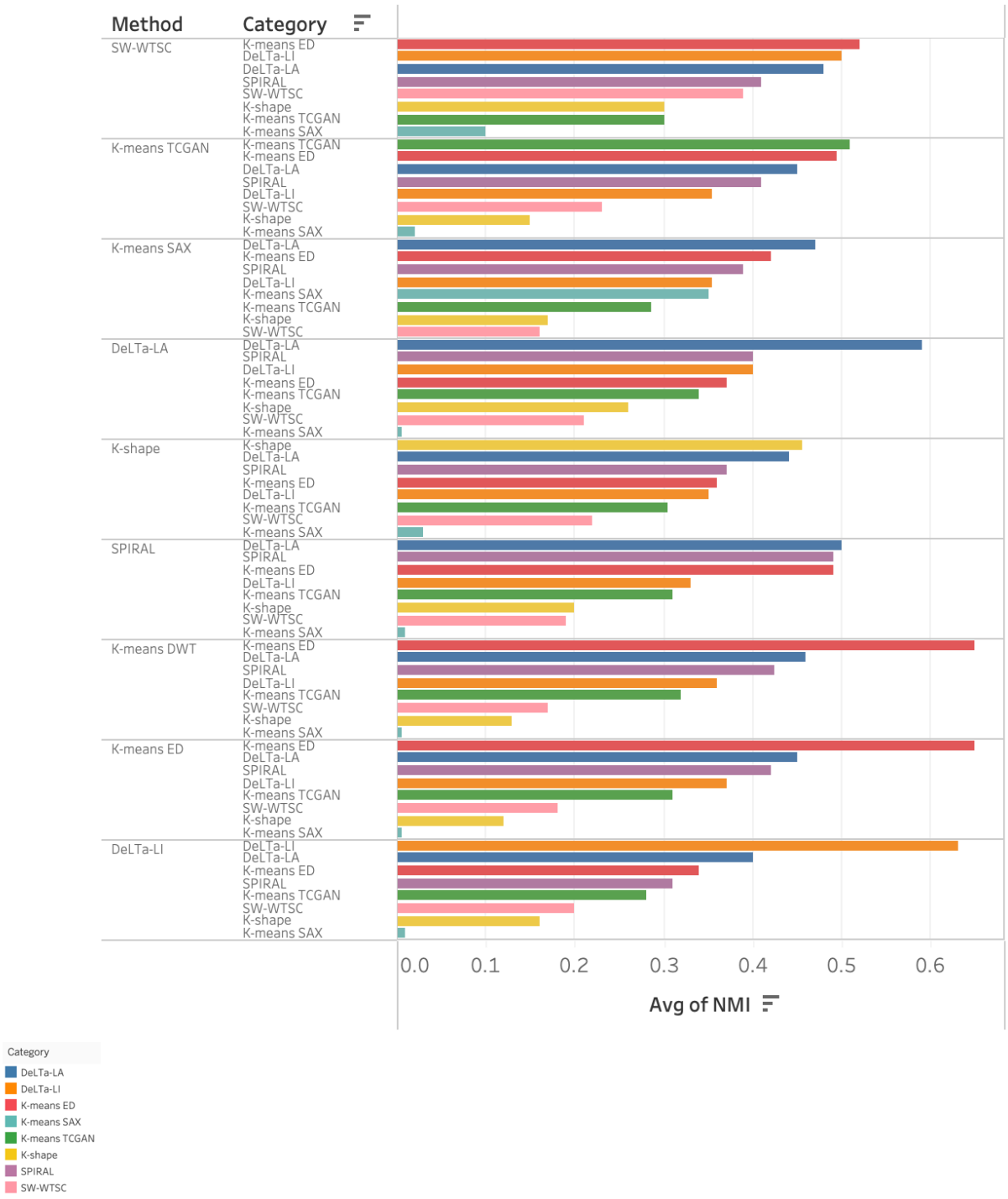

Figure 7 - Visual Comparison of Methods Across Dataset Categories

The noteworthy observation is that the various dataset categories are positioned at different levels. This supports the hypothesis that other methods have varying capabilities in capturing the diverse behaviors of time series. if there were no relationship between the datasets and the performance of the methods, the lines would not appear at different levels.

The SAX category includes two datasets where all methods except SAX perform poorly. In this category, SW-WTSC ranks second in terms of performance. Similar conditions are observed in the DeLTa-LI and K-shape categories, where all methods except the top performer exhibit similar performance, with SW-WTSC achieving results that are intermediate between the top performer and the rest. Only the top-performing method demonstrates a distinct performance advantage in the K-means ED, TCGAN, DeLTa-LA, and SPIRAL categories. At the same time, SW-WTSC shows no significant difference from the

other methods. Thus, it can be concluded that SW-WTSC does not exhibit poor performance in any category relative to its competitors.

To analyze the dependency between the NMI metric derived from different methods and two features, namely time series length and the number of actual clusters in each dataset, the correlation coefficient was calculated and presented in Table 4. By examining the correlation coefficient between the methods' performance and the NMI values, it becomes evident that all methods, except SW-WTSC, show no significant correlation. Since the overall performance of SW-WTSC is positive, and a negative correlation indicates varying results across datasets with different lengths, it can be concluded that SW-WTSC has achieved more significant improvements, specifically in shorter time series.

Similarly, examining the correlation coefficient between the methods' performance and the number of clusters reveals a positive relationship between the number of clusters and the NMI values. This can be interpreted as follows: datasets with more clusters generally exhibit more apparent separations among their clusters, making them easier to distinguish. SW-WTSC shows the least correlation among all correlation coefficients, indicating its superior capability to identify clusters with higher complexity (typically datasets with fewer clusters). This ability produces a lower correlation coefficient, signifying its robustness in challenging scenarios.

Table 4 - Correlation Analysis Between Time Series Characteristics and NMI Performance

Dataset Characteristics	Method								
	DeLTa-LA	DeLTa-LI	SPIRAL	K-means ED	K-means DWT	K-means SAX	K-means TCGAN	K-shape	SW-WTSC
Length	-0.08	-0.03	0.00	0.04	0.03	-0.01	-0.04	-0.12	-0.22
Number of clusters	0.51	0.44	0.45	0.47	0.47	0.47	0.45	0.44	0.28

6 Discussion

In the previous section, experiments confirmed that the effectiveness of SW-WTSC can be statistically verified. Also, it does not exhibit poor performance in any category relative to its competitors. In a significant number of datasets, this method has achieved an improvement of over 80% in the NMI metric. Although it does not rank first in some categories, it has outperformed the average performance of other competitors and demonstrated results comparable to the top method. Finally, in different categories, SW-WTSC has at least matched the average performance of rival methods. However, it is crucial to understand the underlying reasons behind the superior performance of SW-WTSC. This section delves deeper into the fundamental factors contributing to its exceptional results. To achieve this, we will take a closer look at some specific datasets to shed light on the key elements of the method that contribute to its success in pattern detection.

The first dataset analyzed is *Beetle Fly*, with one instance from each cluster illustrated in Figure 8. The significant margin of 0.37 in NMI between SW-WTSC and the second-best method is noteworthy. The time series in this dataset feature triangular sub-patterns with subtle differences.

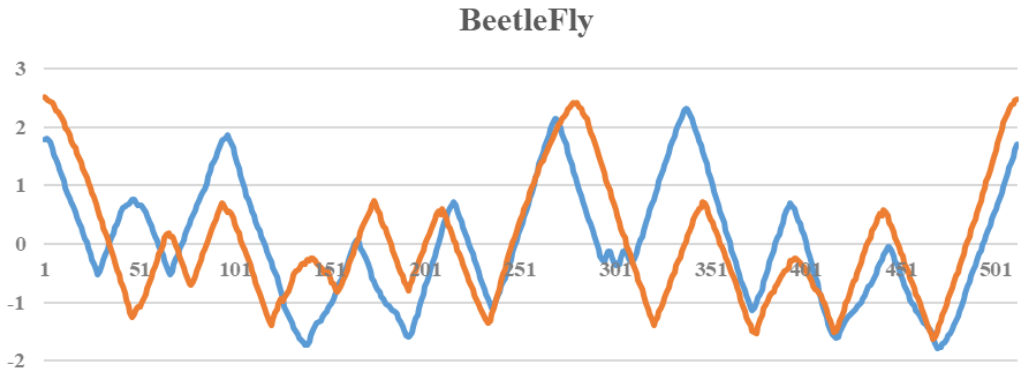

Figure 8 - Visualization of Beetle Fly Instances in Each Cluster

Distance-based methods tend to lose their effectiveness as the complexity of sub-patterns increases because they rely solely on a single overall distance measure between samples, often raising questions about the optimality of the distance calculation itself. A clearer understanding of this concept can be gained from Figure 9. Suppose we aim to cluster the

four patterns illustrated in this figure, with the desired outcome being that the blue and yellow patterns are grouped into one cluster and the green and red patterns into another. If the blue and green samples are chosen as cluster centroids, the distance of the red and yellow samples to these centroids will be equal. This happens because traditional distance metrics do not account for where the differences between the patterns occur. In contrast, SW-WTSC extracts sub-samples and evaluates the distances between these sub-samples within a more localized space, preventing the mixing of unrelated sub-patterns. This approach allows for more nuanced and compelling clustering by focusing on the structural details of the patterns rather than a single aggregated distance.

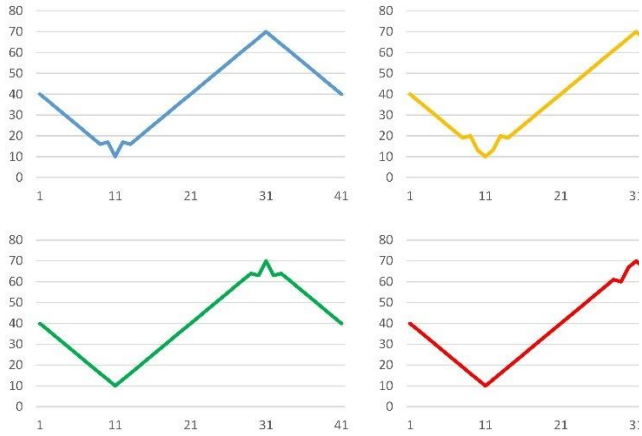

Figure 9 - Illustration of the Limitations of Distance-Based Methods

SW-WTSC, by enabling the recognition of sub-patterns, allows the use of more detailed information for clustering. This capability to discern sub-patterns provides an additional advantage: matching sub-patterns across different scales. This property stems from extracting sub-samples using the sliding window mechanism and then scaling them between 0 and 1. By doing so, the global peaks and troughs within the sub-samples are excluded from the normalization process, facilitating better alignment across varying scales. Figure 10 further illustrates the capabilities of SW-WTSC. Imagine the task is to divide the four time series into two clusters, where the series with two triangles belong to one cluster, and those with a trapezoid and a triangle belong to another. If the entire pattern is normalized globally between 0 and 1, the green and blue samples will appear close to each other, as will the yellow and red samples—resulting in an undesired clustering outcome. The ideal solution is to normalize the sub-patterns locally so that the influence of global peaks and troughs does not distort their shape. SW-WTSC achieves this effectively using a sliding window of length 4. With this approach, all sub-samples from the yellow and blue series will resemble each other, and similarly, the green and red sub-samples will align well, ensuring the desired clustering outcome.

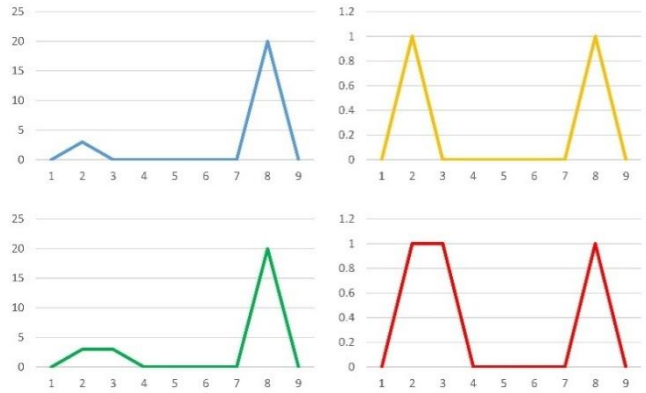

Figure 10 - SW-WTSC's Advantage in Recognizing and Aligning Sub-Patterns in different Vertical scales

The following dataset analyzed is Bird Chicken. In Figure 11, the blue and orange samples belong to Cluster 1, while the yellow and gray samples belong to Cluster 2. Longitudinal scaling and shifts are present in Cluster 2. DeLTa-LI performs best on this dataset, while the other methods (excluding SW-WTSC) show weak results. The strong performance of DeLTa-LI can be attributed to its ability to handle shifts and longitudinal scaling effectively. SW-WTSC also shows notable capability in capturing shifts, as it analyzes sub-patterns without being dependent on the specific position of the sub-pattern within the main time series. However, SW-WTSC is unable to handle longitudinal scaling effectively. This limitation arises because SW-WTSC relies on pointwise Euclidean distance for clustering sub-patterns, which makes it less effective compared to DeLTa-LI in addressing this type of variability, resulting in suboptimal outcomes for this dataset.

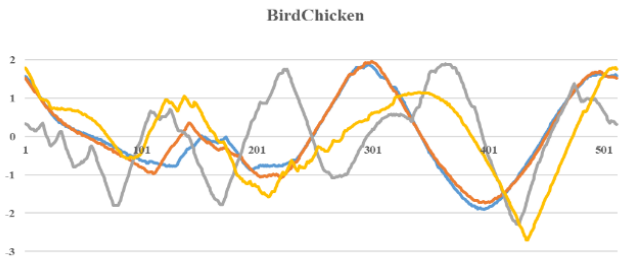

Figure 11 - Visualization of Bird Chicken Instances in Each Cluster

SW-WTSC, in addition to managing shifts in time series, demonstrates strong capability in handling the ordering of sub-patterns within samples. Figure 12 effectively illustrates

this characteristic. Assume we aim for the order of occurrence of sub-patterns within samples not to influence the clustering results so that all four samples belong to a single cluster. By setting the sliding window length to 4, numerous sub-patterns extracted from all four samples are similar, increasing their likelihood of being grouped into the same cluster. If the look-back period for the sliding window is increased, allowing two sub-patterns to be analyzed together, the blue, red, and yellow samples are more likely to cluster together. Further, growing the look-back period to include three sub-patterns restricts acceptable shifts to those that preserve the relative ordering of all three sub-patterns, resulting in only the blue and red samples clustering. This flexibility in controlling the impact of sub-pattern order and scale highlights SW-WTSC's adaptability to different clustering scenarios and its ability to capture nuanced time series characteristics.

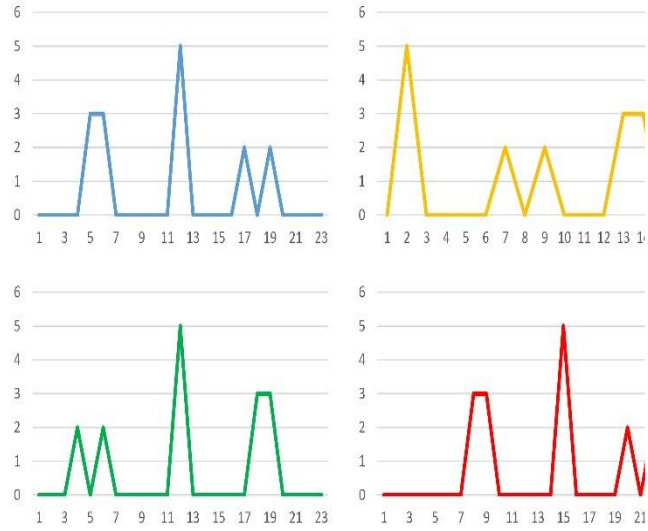

Figure 12 - SW-WTSC's Advantage in Recognizing the Ordering of Sub-Patterns

The CAR dataset comprises four distinct clusters, with a sample from each cluster depicted in Figure 13. The overall patterns among the clusters are highly similar, differing only in subtle and minor details that are challenging to discern. SW-WTSC has shown outstanding performance on this dataset as well. The primary reason behind its success is its ability to capture these minor differences in long time series, which other methods might otherwise overlook. For instance, in representation-based approaches, such sub-patterns could be dismissed as noise after simplification or transformation into a new form, leading to the loss of critical distinguishing features. SW-WTSC's approach to extracting sub-patterns effectively prevents the elimination of such crucial patterns. By focusing on sub-patterns rather than the entire time series, SW-WTSC limits the representation to localized sub-patterns, ensuring that important information is retained. This method avoids the loss of vital details during clustering. As a result, SW-WTSC excels in scenarios where samples appear broadly similar but differ in localized information.

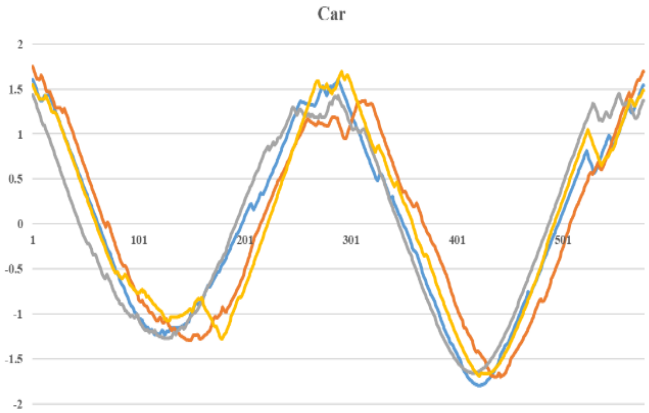

Figure 13 - Visualization of Car Instances in Each Cluster

The CBF dataset, which consists of three clusters, is illustrated in Figure 14. This figure shows that a substantial amount of noise surrounds the three simple patterns that make up the samples. SW-WTSC has successfully identified all the clusters in this dataset. The key reason for this success lies in the two-step data representation process incorporated into the SW-WTSC algorithm, referred to as "data representation." This process is applied both to the original samples and to their sub-samples. Depending on the requirements, either of these representations can be utilized. These two steps effectively handle noise reduction, allowing the algorithm to focus on the essential patterns within the data. By filtering out noise during the data representation stages, SW-WTSC ensures that the clustering results are accurate and robust.

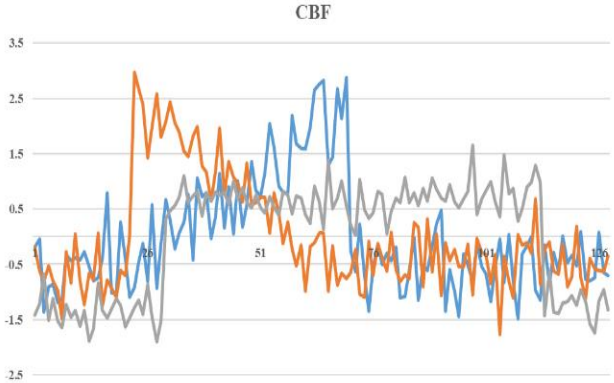

Figure 14 - Visualization of CBF Instances in Each Cluster

The final dataset analyzed is Gun Point. Figure 15 illustrates samples from Cluster 1, while Figure 16 shows samples from Cluster 2. At first glance, it might seem that the clusters are differentiated based on the peak length between the two clusters. However, additional sub-patterns before and after the peak significantly contribute to the clustering process. This dataset also exhibits longitudinal scaling, which adds complexity to the clustering task. Although SW-WTSC outperformed its competitors by a significant margin, its overall performance on this dataset remains suboptimal. One reason for this limitation is SW-WTSC's tendency to generate "fake sub-patterns" under certain conditions, mainly when the time series exhibits relatively stable behavior. This issue arises because SW-WTSC normalizes sub-samples between 0 and 1, focusing solely on the relative positioning of points within sub-samples. As a result, when the distances between points are minimal, and there are no significant variations, the method may erroneously produce sub-patterns that do not indeed exist. These challenges highlight a potential area for improvement in SW-WTSC, particularly in datasets where the primary patterns are subtle and noise or longitudinal scaling plays a significant role.

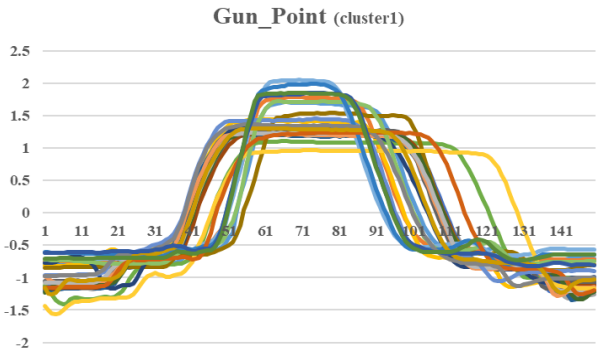

Figure 15 - Visualization of Gun Point Instances in Cluster 1

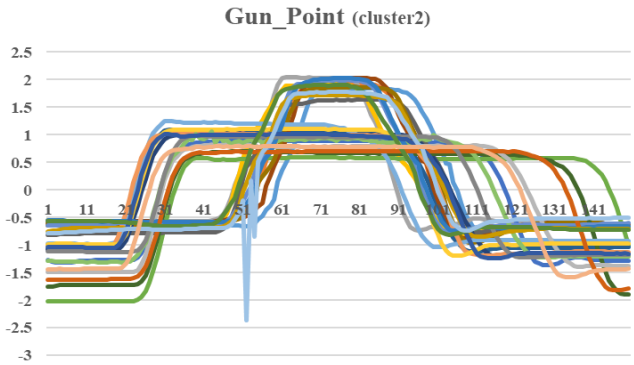

Figure 16 - Visualization of Gun Point Instances in Cluster 2

To clarify the impact of the model's key parameters, a sensitivity analysis has been conducted on the Gun_Point dataset. Section A of Figure 17 illustrates the effect of variations in the sliding window size on the NMI index. This variable has a more significant influence than other parameters, as only a limited range of values result in well-performing models. This is because the sliding window size directly determines the scope of the extracted sub-patterns, defining the portion of the original pattern each sub-pattern represents. Consequently, it affects the standardization of patterns within the range of 0 to 1.

Section B of the figure examines the impact of the number of K-Means clusters, which determines the number of sub-pattern clusters. Depending on the diversity of sub-patterns within a dataset, a specific range of values for this parameter is valid. As shown in the figure, increasing the number of clusters up to 70 positively influences the NMI score. This suggests that, within this range, the number of clusters is insufficient to fully capture all sub-pattern groups, leading to the merging of dissimilar sub-patterns within the same cluster. As a result, sub-patterns are not grouped correctly, reducing model performance. Beyond this threshold, the number of clusters becomes sufficient to represent distinct sub-pattern groups, and further changes do not significantly affect the results.

Section C highlights the impact of the Sub-sample Representation length parameter. This parameter must be determined based on the level of noise in the data and the selected sliding window size. A lower ratio of these two parameters results in more significant simplification, ignoring more variations. Based on these findings, the sliding window size significantly impacts the model, while the other two parameters require fine-tuning to optimize performance.

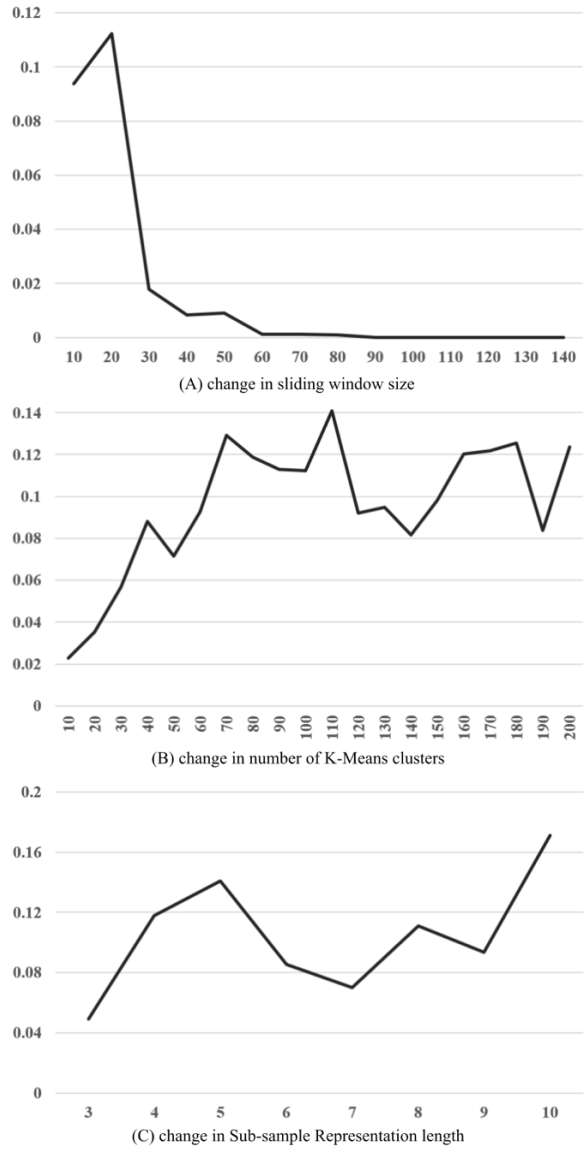

Figure 17 - Sensitivity analysis of key model parameters on the Gun_Point dataset.

While STSC might appear meaningless due to the shifting of cluster centers during each clustering run, this does not undermine the performance of WTSC. The objective of STSC is not to achieve fixed and meaningful cluster centers; instead, it leverages the similarity among samples within the same cluster and their dissimilarity to those in other clusters. The key to this effectiveness lies in the use of the sub-pattern pool. When all sub-patterns are clustered simultaneously, the similarity property among samples within a cluster can be utilized. This eliminates the need to identify existing patterns as a baseline criterion. Instead, it suffices for the sub-patterns of different samples to rely on a consistent coding system to define similarity. The clustering of the sub-pattern pool enables this capability. Thus, STSC, by providing this functionality, effectively plays its positive and essential role within SW-WTSC.

Figure 18 illustrates the impact of STSC, which serves as the foundation for the first stage of clustering in the proposed algorithm. This stage focuses on transforming data points to improve the effectiveness of the second-stage clustering model in identifying appropriate clusters. The figure presents t-SNE visualizations for four datasets—BeetleFly, FaceFour, BirdChicken, and CBF before and after the first clustering stage. The results indicate that STSC enables clear separation of samples in the FaceFour and CBF datasets, with the clustering models achieving an NMI score of 1 on the test data. This suggests that STSC effectively repositions data points based on the similarity of their sub-patterns, bringing similar samples closer together while increasing the distance between dissimilar ones. Consequently, the second clustering stage is facilitated. In contrast, complete separation is not observed in the BirdChicken and BeetleFly datasets; however, intra-cluster dispersion is reduced, leading to an NMI score of approximately 0.5. Despite this improvement, multiple shared sub-patterns among samples limit the extent of separation in these datasets.

Although sub-pattern clustering using STSC provides several advantages, it is essential to note that this method may not handle specific characteristics of time series well. The presence of outliers, for example, can affect some of the generated sub-patterns. Suppose the effect of an outlier is not mitigated during the sub-pattern representation process. In that case, it may lead to the generation of distorted sub-patterns, making it challenging to represent a given sample correctly. Longitudinal scaling is another characteristic that can result in the formation of unsuitable sub-patterns. When such issues influence the quality of sub-pattern clustering, they can ultimately hinder the accurate identification of cluster labels for samples, thereby reducing the algorithm's overall performance.

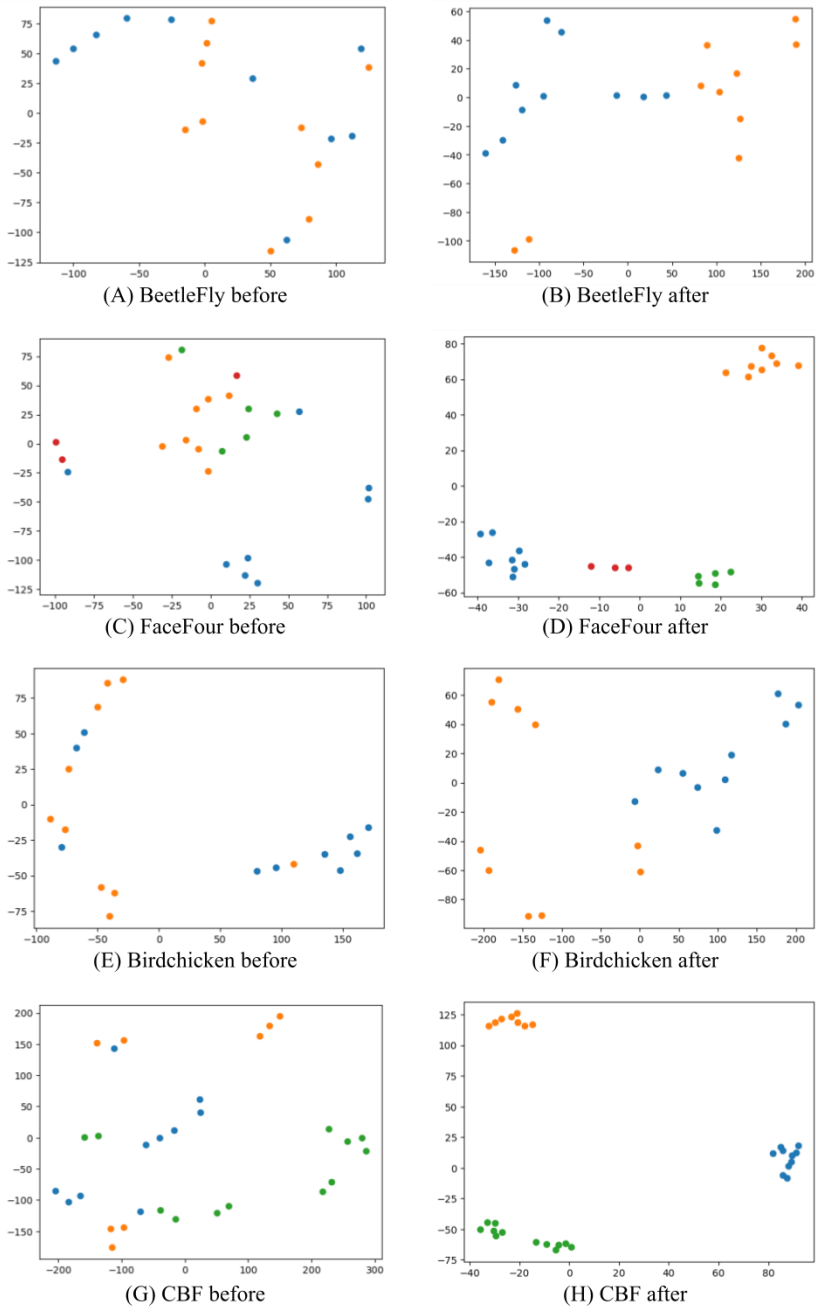

Figure 18 - t-SNE visualization of datasets BeetleFly, FaceFour, BirdChicken, and CBF before and after applying first stage of clustering.

Based on the obtained results and analyses, it can be concluded that the proposed SW-WTSC method outperforms other existing approaches due to its unique capabilities: (i) it demonstrates a lower correlation with problem simplicity, showing strong performance in clustering complex time series; (ii) it effectively differentiates samples by analyzing sub-patterns and clustering based on them, outperforming traditional distance-based measures; (iii) it enables the independent scaling of sub-patterns, allowing for better alignment across varying scales; (iv) it exhibits high robustness to shift and noise in time series data; and (v) it offers order-invariance by disregarding the sequence of sub-patterns, enhancing its flexibility in clustering diverse time series. These attributes collectively establish SW-WTSC as a robust and versatile approach for time series clustering.

Although this method offers numerous advantages, it is not without limitations, which should be acknowledged: (i) the negative impact of time series length on the method's performance, although it shows relative improvement compared to other approaches for long time series; (ii) The lack of support for specific characteristics of time series,

such as longitudinal scaling or the presence of outliers; (iii) the generation of fake sub-patterns when encountering sub-samples with overall stable conditions; and (iv) the high number of parameters in the method, which require trial-and-error adjustments as a tool to achieve the desired outcomes. These limitations highlight areas for potential improvement in the proposed approach.

Ultimately, several key findings can be drawn from these experiments: (i) SW-WTSC demonstrated its effectiveness in WTSC; (ii) although performing STSC is often meaningless, the property of similarity among samples within a cluster can still be utilized; (iii) breaking down the complexities of time series into simpler, analyzable components contributes to better method performance; (iv) most methods focus on capturing specific behavioral patterns in time series, which impacts their generalizability; and (v) clustering does not necessarily have a single definitive answer, and the problem's needs define the desirability of results. Thus, SW-WTSC, with its strong capability to focus on various sub-patterns, exhibits exceptional performance.

7 Conclusions

The SW-WTSC method is introduced in this study as an innovative approach to address WTSC problems. It aims to provide a robust clustering solution for time series by capturing various inherent data characteristics. SW-WTSC achieves this by breaking down the complexity of time series by identifying sub-patterns. Using a sliding window, it extracts subsequences from samples and, when necessary, applies a novel algorithm to represent the data as a shorter time series, whether for the original samples or the sub-samples. The method then scales the sub-sample values between 0 and 1 and creates a pool of sub-patterns, applying a new feature extraction approach. This approach generates features based on the similarity of sub-samples to predefined patterns. Finally, the samples are labeled through a two-stage clustering process: in the first phase, sub-patterns are assigned labels, and in the second phase, these labels serve as features for clustering the samples.

To verify the effectiveness of the SW-WTSC method in solving WTSC problems, the NMI metric was calculated by applying this method to 85 datasets from the UCR archive. Furthermore, a non-parametric Friedman test was conducted on the results obtained from this method and those from eight state-of-the-art methods. The results confirmed a significant difference in the performance of SW-WTSC, demonstrating its superiority over competing methods. This experiment also revealed that the proposed method can handle the behavioral characteristics present in time series data. So, given the widespread applications of WTSC in fields such as finance, healthcare, signal detection, and more, SW-WTSC can be effectively utilized in these domains, delivering promising results.

Based on the experiments, SW-WTSC demonstrates outstanding performance, which can be attributed to several features, such as (i) reduced sensitivity to complex problems, (ii) clustering based on finer-grained data, (iii) independence in vertical scaling of sub-patterns, (iv) a high ability to handle shifts and noise, and (v) flexibility in accommodating the sequence ordering of sub-patterns for clustering. The experimental results confirmed these characteristics collectively contribute to its effectiveness. However, the method has notable limitations, including (i) sensitivity to the length of time series, (ii) inability to handle longitudinal scaling, (iii) the generation of fake sub-patterns under specific conditions, and (iv) reliance on a large number of parameters. Addressing these limitations could significantly enhance the development of future sub-pattern-based clustering methods.

Based on the findings, this study presents several key contributions, including (i) proposing a novel approach, SW-WTSC, for the whole time series clustering problem through sub-pattern identification via a sliding window mechanism, (ii) introducing a new feature selection technique based on similarity to predefined patterns, (iii) effective use of subsequence clustering within whole clustering, (iv) flexibility in focusing on specific time series characteristics based on problem requirements, and (v) verifying the proposed method through experiments on UCR datasets, demonstrating its superiority over competitive methods.

Considering the identified limitations and drawing inspiration from the contributions of this research, several potential directions for future work are proposed: (i) utilizing alternative distance measures, such as DTW, to enable support for longitudinal scaling, (ii) addressing the dependency of results on the length of time series, (iii) improving the algorithm to mitigate the issue of generating fake sub-patterns, and (iv) analyzing the impact of subsequences extracted with different look-back periods on the first-stage clustering process.

Ethical Considerations

All procedures performed in this study were under the ethical standards.

Acknowledgments

The authors gratefully acknowledge Professor Seyed Ali Mirjalili for his valuable guidance and constructive feedback throughout this research.

Conflict of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

References

[1] T. Hayashi *et al.*, "Distance-based one-class time-series classification approach using local cluster balance," *Expert Systems with Applications,* vol. 235, p. 121201, 2024, doi: 10.1016/j.eswa.2023.121201.

[2] J. Wu, Z. Zhang, R. Tong, Y. Zhou, Z. Hu, and K. Liu, "Imaging feature-based clustering of financial time series," *Plos one,* vol. 18, no. 7, p. e0288836, 2023, doi: 10.1371/journal.pone.0288836.

[3] J. Dumler, S. Faatz, M. Friedrich, and F. Döpper, "Automatic time series segmentation and clustering for process monitoring in series production," *Procedia CIRP,* vol. 118, pp. 602-607, 2023, doi: 10.1016/j.procir.2023.06.103.

[4] L. N. Ferreira and L. Zhao, "Time series clustering via community detection in networks," *Information Sciences,* vol. 326, pp. 227-242, 2016, doi: 10.1016/j.ins.2015.07.046.

[5] M. Landauer, F. Skopik, B. Stojanović, A. Flatscher, and T. Ullrich, "A review of time-series analysis for cyber security analytics: from intrusion detection to attack prediction," *International Journal of Information Security,* vol. 24, no. 1, p. 3, 2025, doi: 10.1007/s10207-024-00921-0.

[6] F. Huang and Y. Deng, "TCGAN: Convolutional Generative Adversarial Network for time series classification and clustering," *Neural Networks,* vol. 165, pp. 868-883, 2023, doi: 10.1016/j.neunet.2023.06.033.

[7] Y. Li, D. Shen, T. Nie, and Y. Kou, "A new shape-based clustering algorithm for time series," *Information Sciences,* vol. 609, pp. 411-428, 2022, doi: 10.1016/j.ins.2022.07.105.

[8] K. Mishra, S. Basu, and U. Maulik, "Graft: A graph based time series data mining framework," *Engineering Applications of Artificial Intelligence,* vol. 110, p. 104695, 2022, doi: 10.1016/j.engappai.2022.104695.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press, 1975.

[10] H. Li and T. Du, "Multivariate time-series clustering based on component relationship networks," *Expert Systems with Applications,* vol. 173, p. 114649, 2021, doi: 10.1016/j.eswa.2021.114649.

[11] G. Richard, B. Grossin, G. Germaine, G. Hebrail, and A. de Moliner, "Autoencoder-based time series clustering with energy applications," *arXiv preprint arXiv:2002.03624,* 2020. [Online]. Available: https://arxiv.org/abs/2002.03624.

[12] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh, "A review of subsequence time series clustering," *The Scientific World Journal,* vol. 2014, no. 1, p. 312521, 2014, doi: 10.1155/2014/312521.

[13] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research," *Knowledge and information systems,* vol. 8, pp. 154-177, 2005, doi: 10.1007/s10115-004-0172-7.

[14] L. G. B. Ruiz, M. C. Pegalajar, R. Arcucci, and M. Molina-Solana, "A time-series clustering methodology for knowledge extraction in energy consumption data," *Expert Systems with Applications,* vol. 160, p. 113731, 2020, doi: 10.1016/j.eswa.2020.113731.

[15] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, "Clustering-based anomaly detection in multivariate time series data," *Applied Soft Computing,* vol. 100, p. 106919, 2021, doi: 10.1016/j.asoc.2020.106919.

[16] T. C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 1, pp. 164-181, 2011, doi: 10.1016/j.engappai.2010.09.007.

[17] D. Tiano, A. Bonifati, and R. Ng, "Feature-driven time series clustering," in *24th International Conference on Extending Database Technology, EDBT 2021*, 2021, doi: 10.1145/3448016.3452757.

[18] S. Y. Chen, T. T. Tseng, H. R. Ke, and C. T. Sun, "Social trend tracking by time series based social tagging clustering," *Expert Systems with Applications,* vol. 38, no. 10, pp. 12807-12817, 2011, doi: 10.1016/j.eswa.2011.04.073.

[19] P. D'Urso, L. De Giovanni, and V. Vitale, "Robust DTW-based entropy fuzzy clustering of time series," *Annals of Operations Research,* pp. 1-35, 2023, doi: 10.1007/s10479-023-05720-9.

[20] R. Umatani, T. Imai, K. Kawamoto, and S. Kunimasa, "Time series clustering with an EM algorithm for mixtures of linear Gaussian state space models," *Pattern Recognition,* vol. 138, p. 109375, 2023, doi: 10.1016/j.patcog.2023.109375.

[21] C. Holder, M. Middlehurst, and A. Bagnall, "A review and evaluation of elastic distance functions for time series clustering," *Knowledge and Information Systems,* vol. 66, no. 2, pp. 765-809, 2024, doi: 10.1007/s10115-023-01952-0.

[22] S. Kamro, M. Rafiee, and S. Mirjalili, "Metaheuristic-driven space partitioning and ensemble learning for imbalanced classification," *Applied Soft Computing,* vol. 167, p. 112278, 2024, doi: 10.1016/j.asoc.2024.112278.

[23] G. J. Oyewole and G. A. Thopil, "Data clustering: application and trends," *Artificial Intelligence Review,* vol. 56, no. 7, pp. 6439-6475, 2023, doi: 10.1007/s10462-022-10325-y.

[24] S. Rodpongpun, V. Niennattrakul, and C. A. Ratanamahatana, "Selective subsequence time series clustering," *Knowledge-Based Systems,* vol. 35, pp. 361-368, 2012, doi: 10.1016/j.knosys.2012.04.022.

[25] G. Simon, J. A. Lee, and M. Verleysen, "Unfolding preprocessing for meaningful time series clustering," *Neural Networks,* vol. 19, no. 6-7, pp. 877-888, 2006, doi: 10.1016/j.neunet.2006.05.020.

[26] K. A. Peker, "Subsequence time series (STS) clustering techniques for meaningful pattern discovery," in *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005: IEEE, pp. 360-365, doi: 10.1109/KIMAS.2005.1427109.

[27] J. Wang, X. Sun, M. F. She, A. Kouzani, and S. Nahavandi, "Unsupervised mining of long time series based on latent topic model," *Neurocomputing,* vol. 103, pp. 93-103, 2013, doi: 10.1016/j.neucom.2012.09.008.
[28] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon, "Similarity preserving representation learning for time series clustering," *arXiv preprint arXiv:1702.03584,* 2017. [Online]. Available: https://arxiv.org/abs/1702.03584.
[29] G. Anand, "Unsupervised visual perception-based representation learning for time-series and trajectories," Doctoral dissertation, Queensland University of Technology, 2021. [Online]. Available: https://eprints.qut.edu.au/212901/
[30] H. Zhang, T. B. Ho, Y. Zhang, and M. S. Lin, "Unsupervised feature extraction for time series clustering using orthogonal wavelet transform," *Informatica,* vol. 30, no. 3, 2006. [Online]. Available: http://www.informatica.si/index.php/informatica/article/view/98/91.
[31] C. P. Lai, P. C. Chung, and V. S. Tseng, "A novel two-level clustering method for time series data analysis," *Expert Systems with Applications,* vol. 37, no. 9, pp. 6319-6326, 2010, doi: 10.1016/j.eswa.2010.02.089.
[32] G. Anand and R. Nayak, "DeLTa: Deep local pattern representation for time-series clustering and classification using visual perception," *Knowledge-Based Systems,* vol. 212, p. 106551, 2021, doi: 10.1016/j.knosys.2020.106551.
[33] X. Zhang, J. Liu, Y. Du, and T. Lv, "A novel clustering method on time series data," *Expert Systems with Applications,* vol. 38, no. 9, pp. 11891-11900, 2011, doi: 10.1016/j.eswa.2011.03.081.
[34] B. Cai, G. Huang, N. Samadiani, G. Li, and C. H. Chi, "Efficient time series clustering by minimizing dynamic time warping utilization," *IEEE access,* vol. 9, pp. 46589-46599, 2021, doi: 10.1109/ACCESS.2021.3067833.
[35] 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,* vol. 534, pp. 97-116, 2020, doi: 10.1016/j.ins.2020.04.009.
[36] A. Srivastava, "DTW+S: Shape-based Comparison of Time-series with Ordered Local Trend," *arXiv preprint arXiv:2309.03579,* 2023. [Online]. Available: https://arxiv.org/abs/2309.03579.
[37] G. Soleimani and M. Abessi, "DLCSS: A new similarity measure for time series data mining," *Engineering Applications of Artificial Intelligence,* vol. 92, p. 103664, 2020, doi: 10.1016/j.engappai.2020.103664.
[38] Y. Teng, J. Liu, K. Wu, Y. Liu, and B. Qiao, "Multivariate time series clustering based on fuzzy cognitive maps and community detection," *Neurocomputing,* vol. 590, p. 127743, 2024, doi: 10.1016/j.neucom.2024.127743.
[39] P. D'Urso and J. M. Leski, "OWA-based robust fuzzy clustering of time series with typicality degrees," *Information Sciences,* vol. 651, p. 119706, 2023, doi: 10.1016/j.ins.2023.119706.
[40] F. Setoudehtazangi, T. Manouchehri, A. R. Nematollahi, and M. Caporin, "Time series clustering based on latent volatility mixture modeling with applications in finance," *Mathematics and Computers in Simulation,* vol. 223, pp. 543-564, 2024, doi: 10.1016/j.matcom.2024.04.031.
[41] X. Huang, Y. Ye, L. Xiong, R. Y. Lau, N. Jiang, and S. Wang, "Time series k-means: A new k-means type smooth subspace clustering for time series data," *Information Sciences,* vol. 367, pp. 1-13, 2016, doi: 10.1016/j.ins.2016.05.040.
[42] J. F. Vera and J. M. Angulo, "An MDS-based unifying approach to time series K-means clustering: application in the dynamic time warping framework," *Stochastic Environmental Research and Risk Assessment,* vol. 37, no. 12, pp. 4555-4566, 2023, doi: 10.1007/s00477-023-02470-9.
[43] 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*, 2015, pp. 1855-1870, doi: 10.1145/2723372.2737793.
[44] H. Li and M. Chen, "Time series clustering based on normal cloud model and complex network," *Applied Soft Computing,* vol. 148, p. 110876, 2023, doi: 10.1016/j.asoc.2023.110876.
[45] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 947-956, doi: 10.1145/1557019.1557122.
[46] B. Cai, G. Huang, S. Yang, Y. Xiang, and C. H. Chi, "SE-shapelets: Semi-supervised Clustering of Time Series Using Representative Shapelets," *Expert Systems with Applications,* vol. 240, p. 122584, 2024, doi: 10.1016/j.eswa.2023.122584.
[47] A. M. Jarman, "Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method," *Georgia Southern University,* vol. 29, 2020. [Online]. Available: https://www.researchgate.net/profile/Angur-Jarman/publication/339443595_Hierarchical_Cluster_Analysis_Comparison_of_Single_linkageComplete_linkage_Average_linkage_and_Centroid_Linkage_Method/links/5e52f582a6fdcc2f8f5d6f2e/Hierarchical-Cluster-Analysis-Comparison-of-Single-linkage-Complete-linkage-Average-linkage-and-Centroid-Linkage-Method.pdf.