Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 9(12): 35551–35569.
DOI: 10.3934/math.20241687
Received: 23 July 2024
Revised: 20 November 2024
Accepted: 27 November 2024
Published: 20 December 2024

Research article

The advantages of k-visibility: A comparative analysis of several time series clustering algorithms

Sergio Iglesias-Perez[1,2,3,*], **Alberto Partida**[1,2] **and Regino Criado**[1,4]

[1] Data, Complex Networks and Cybersecurity Sciences Technological Institute, Univ. Rey Juan Carlos, 28028 Madrid, Spain

[2] Science, Computing, and Technology Department, School of Architecture, Engineering and Design, Universidad Europea de Madrid, Calle Tajo, S/N, Villaviciosa de Odón, 28670 Madrid, Spain

[3] Department of Computer Science and Technology, Universidad Internacional de La Rioja, Logroño, Spain

[4] Departamento de Matematica Aplicada Ciencia e Ingenieria de los Materiales y Tecnologia Electronica ESCET Universidad Rey Juan Carlos C Tulipan, 28933 Mostoles (Madrid), Spain

* **Correspondence:** Email: sergio.iglesias@dcncsciences.com.

Abstract: This paper outlined the advantages of the k-visibility algorithm proposed in [1,2] compared to traditional time series clustering algorithms, highlighting enhanced computational efficiency and comparable clustering quality. This method leveraged visibility graphs, transforming time series into graph structures where data points were represented as nodes, and edges are established based on visibility criteria. It employed the traditional k-means clustering method to cluster the time series. This approach was particularly efficient for long time series and demonstrated superior performance compared to existing clustering methods. The structural properties of visibility graphs provided a robust foundation for clustering, effectively capturing both local and global patterns within the data. In this paper, we have compared the k-visibility algorithm with 4 algorithms frequently used in time series clustering and compared the results in terms of accuracy and computational time. To validate the results, we have selected 15 datasets from the prestigious UCR (University of California, Riverside) archive in order to make a homogeneous validation. The result of this comparison concluded that k-visibility was always the fastest algorithm and that it was one of the most accurate in matching the clustering proposed by the UCR archive.

Keywords: time series; visibility graph; clustering; computational efficiency
Mathematics Subject Classification: 05C85, 68R10, 68W05

1. Introduction

First, the widespread use of data collection systems across different fields has greatly increased the need for effective temporal analysis. Time series data, which records sequential observations over time, is crucial for identifying trends, detecting anomalies, and understanding patterns.

Recently, affordable increases in storage and computational power have transformed various industries. These advancements now enable us to collect and process vast amounts of time-based data, which was previously impossible. As a result, time series data has become essential for predicting trends and clustering related patterns.

Prediction, anomaly detection, and clustering are three key applications of time series data. Prediction forecasts future values based on past observations, which is vital for planning and decision-making in industries like stock market analysis and weather forecasting. Anomaly detection identifies outliers or unusual patterns, helping to spot fraud, equipment failures, or health issues early. Clustering groups similar time series together, uncovering common patterns and behaviors across large datasets, which is useful for market segmentation, user behavior analysis, and more.

Second, natural visibility graphs (NVGs) provide a novel method for analyzing time series by transforming them into graph structures. First introduced by Lacasa et al. [3] and further explored by Partida et al. [4] and Lopes et al. [5], NVGs and their variant, horizontal visibility graphs (HVGs) [6], convert time series data into graphs where each data point is a node, and edges are created based on visibility rays that link "visible" data points. VGs and HVGs have been applied across various fields, including financial market analysis, physiological data studies, and climate research, leveraging the structural properties of these graphs to uncover underlying patterns and dynamics.

Typically, graphs and time series are considered two distinct mathematical fields. Graph theory has traditionally focused on solving problems involving relationships between nodes connected by edges. However, over the past two decades, there has been increasing interest in combining these two fields to leverage their unique strengths. The integration of time series and graph theory through visibility graphs provides a powerful tool for analyzing complex datasets, offering new insights and deepening our understanding of time-dependent phenomena.

There are numerous approaches for practical applications [7]. However, our proposal stands out by offering a faster method for clustering time series data compared to traditional approaches based on time series features.

The structure of this paper is as follows: After this introduction, Section 2 provides a detailed explanation of the most widely used time series clustering techniques. In Section 3, we present our methodology, followed by the results of our experiments in Section 4, along with a discussion of the implications. Finally, Section 5 offers conclusions and suggests future research directions related to this work.

2. Background and related works

Time series clustering is a mathematical process that, given a set of n time series, $D = \{ts_1, ts_2, ts_3, ..., ts_n\}$, groups them into N clusters $C = \{C_1, C_2, C_3, ...C_N\}$.

There are many ways to group time series data today. Based on the existing literature [8] and [9], time series clustering techniques can be broadly categorized into three approaches:

- Model-based;
- Feature-based;
- Shape-based.

In the following sections, we explain these approaches and their practical applications.

2.1. Model-based approach

In this approach, the goal is to model the time series in order to extract relevant information about its structure and behavior. The core idea is that if two time series are similar, their respective models will also be similar. By comparing the parameters of these models, one can identify similarities and group similar time series.

There are several techniques to model time series, with common approaches including ARIMA (autoregressive integrated moving average) models and hidden Markov models (HMM).

ARIMA Models: These are widely used for capturing temporal dependencies, trends, and seasonality in time series. By fitting an ARIMA model to each series, you can compare the resulting parameters, such as autoregressive, moving average, and differencing coefficients, to find similar patterns between series.

HMM: HMMs are useful when the underlying process is believed to follow a sequence of hidden states. Each time series can be modeled as a series of transitions between these states, and the transition probabilities and emission distributions can be compared to cluster similar series.

Other approaches, like GARCH (generalized autoregressive conditional heteroskedasticity) models or STM (structural time series models), can also be employed, depending on the nature of the data. This method is advantageous as it allows for the comparison of time series with different lengths by focusing on the underlying model parameters rather than raw data points.

Sources such as [10], [11], and [12] provide a deeper exploration of these model-based techniques for time series clustering.

2.2. Feature-based approach

As shown in Figure 1, feature extraction involves deriving meaningful characteristics from raw time series data [13]. These features capture various aspects, such as statistical properties, frequency domain information, or other domain-specific attributes. We denote the time series data as x=x1,x2,x3,. . . ,xn, where n is the length of the time series. Some commonly used features in time series analysis include:

- Statistical moments (e.g., mean and variance):
  - mean:
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i,$$
  - variance of a time series:
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2.$$
- Autocorrelation coefficients
$$\rho_k = \frac{\sum_{i=k+1}^{n}(x_i - \mu)(x_{i-k} - \mu)}{\sum_{i=1}^{n}(x_i - \mu)^2}.$$

- Fourier transform coefficients;
- Wavelet transform coefficients;
- Histogram-based features;
- Entropy measures.

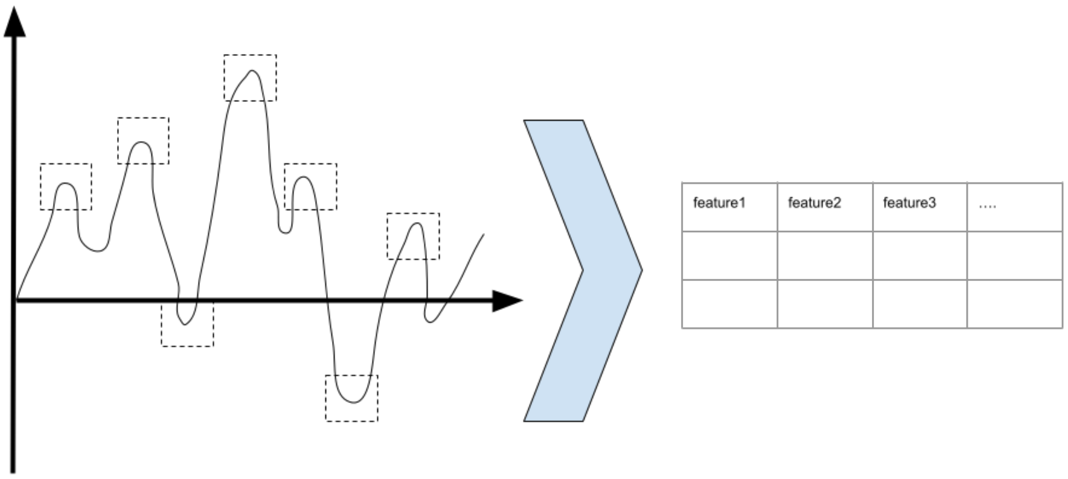

Figure 1. Tsfresh feature approach.

The choice of features depends on the characteristics of the time series data and the specific requirements of the analysis. Various features can be extracted based on these criteria.

After feature extraction, the time series data is represented as a feature vector. The similarity between time series is typically computed using a distance metric applied to these feature vectors.

Feature-based approaches are particularly advantageous when the shapes or patterns of the time series are not the primary focus or when the data exhibits complex patterns that are difficult to capture directly.

2.3. Shape-based approach

The shape-based approach is a specific type of feature-based approach. One of the challenges in successfully clustering all types of time series is the need for a system that captures similarity in two key measures: amplitude and phase.

Among the most commonly used clustering approaches are distance-based methods, such as dynamic time warping (DTW). However, these methods often incur high computational costs due to their reliance on cross-correlation between time series.

DTW was proposed in 1994 [14] as a novel method for comparing and clustering time series. DTW aligns distinctive patterns in the time series, resulting in a more accurate similarity assessment compared to Euclidean distance, which matches timestamps without considering feature values.

Cross-correlation is a similarity measure that compares signal points with a time lag. It is widely used in signal processing to compare sequences that differ in phase. This approach focuses primarily on the shape of the time series rather than the absolute values at each point.

While traditional approaches to time series analysis, such as model-based, feature-based, and shape-based methods, offer robust solutions for classification and pattern recognition, they often suffer from limitations in computational efficiency and scalability. Model-based methods, which rely on fitting parametric or probabilistic models, can be computationally expensive, especially for high-dimensional data, as they require significant tuning and assumptions about data distribution. Feature-based methods, which depend on handcrafted features, may fail to capture subtle temporal dependencies and are sensitive to the chosen features, which can lead to reduced generalizability across different datasets. Shape-based methods, particularly those relying on DTW and other distance measures, are effective for measuring similarity but are computationally intensive and scale poorly with large datasets due to their quadratic time complexity.

The k-visibility approach leverages graph visibility theory to transform time series data into networks, capturing essential structural patterns through a more scalable representation. Unlike model-based and feature-based approaches, k-visibility does not require extensive feature engineering or assumptions about data distribution, allowing it to generalize across various types of time series. Furthermore, by representing data as a graph, k-visibility reduces the computational burden associated with direct time-domain comparisons, offering a more efficient alternative to shape-based methods. This advantage makes k-visibility particularly suitable for large-scale datasets where computational cost and scalability are critical considerations. While k-visibility offers several advantages, particularly in terms of scalability and reduced computational complexity, it also presents certain limitations. One of the main drawbacks is its reliance on the visibility graph transformation, which can oversimplify certain time series by focusing primarily on prominent peaks and valleys, potentially neglecting subtle but meaningful variations within the data. This simplification can be particularly problematic in datasets where fine-grained temporal dependencies are crucial for distinguishing between classes. Additionally, the conversion from time series to a graph-based representation introduces a layer of abstraction that may obscure the interpretability of the original temporal patterns, which can hinder the ability to analyze the time domain directly. Furthermore, while k-visibility is computationally efficient compared to some traditional shape-based methods, it can still be sensitive to data noise, as minor fluctuations in the series can alter the graph structure, potentially impacting classification accuracy. Thus, while k-visibility is a promising alternative, its effectiveness may vary across different applications, particularly where the preservation of subtle temporal information is critical.

3. Methodology

In this section, we introduce a novel approach that transforms a time series into a graph, followed by the application of a traditional clustering technique for time series clustering. This graph-based transformation enables the exploration of previously unconsidered features.

By leveraging this transformation, we propose a new method for clustering time series based on graph features rather than conventional time series attributes. Specifically, this approach allows for a straightforward clustering method by utilizing complex features derived from converting the time series into a visibility graph.

The process of this new time series clustering is based in three phases as we can see in Figure 2:

(1) from time series to networks: we first transform the time series into a visibility graph, using both natural and horizontal visibility methods, as depicted in Figure 3.

(2) feature extraction from resulting networks: we analyze the best features from a network to capture the behavior of the time series.

(3) clustering based on network features: finally, we cluster with a traditional approach, k-means, based on the features acquired from the built networks.

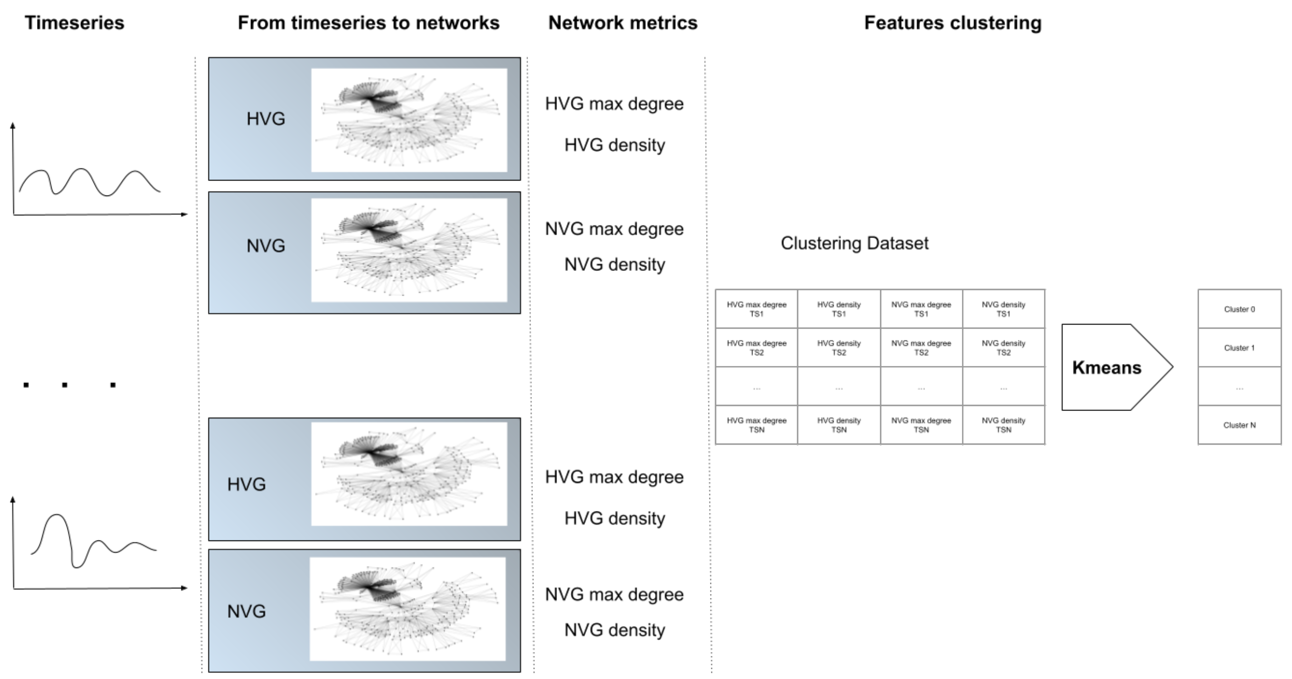

Figure 2. k-visibility flow.

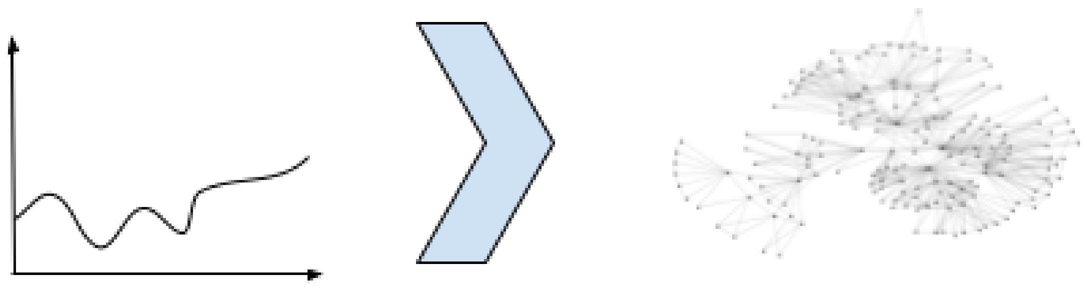

Figure 3. From time series to networks.

3.1. *From time series to networks*

The first step in this approach involves transforming the time series into a graph-based representation using the visibility graph technique as we describe in Figure 3, first introduced by Lacasa et al. in 2008 [3]. This method allows the properties of the time series to be reflected in the network structure: for instance, visibility graphs generated from periodic series form recurrent networks, while those from random series yield random networks. Similarly, fractal series can be represented as scale-free networks [15].

This perspective enables us to analyze visibility graphs to extract attributes that are valuable for clustering time series.

3.1.1. NVGs

Based on Lacasa et al.'s work [3], the visibility graph is defined as a network in which each point in the time series is connected to others that are within its line of sight. In other words, two points, or nodes, are connected if they have a "visibility line" between them.

Formally, two points in the time series, denoted as (t_a, y_a) and (t_b, y_b) are considered connected, and thus become linked nodes in the graph, if any intermediate point (t_c, y_c) satisfies:

$$y_c < y_b + (y_a - y_b)\frac{t_b - t_c}{t_b - t_a}.$$

Algorithm 1 Construction of the NVG

```
Require: $X = \{x_1, x_2, \ldots, x_n\}$, where $x_i$ represents the time series values.
Ensure: Graph $G = (V, E)$, where $V$ are the nodes and $E$ the edges.
 1: Initialize $V \leftarrow \{1, 2, \ldots, n\}$ (one node per time series point).
 2: Initialize $E \leftarrow \emptyset$ (no edges initially).
 3: for each pair $i, j \in V$ with $i < j$ do
 4:     Compute the slope between $x_i$ and $x_j$: $m \leftarrow \frac{x_j - x_i}{j - i}$
 5:     Visible ← True
 6:     for $k = i + 1$ to $j - 1$ do
 7:         if $x_k \geq x_i + m \cdot (k - i)$ then
 8:             Visible ← False
 9:             break
10:         end if
11:     end for
12:     if Visible then
13:         Add edge $(i, j)$ to $E$
14:     end if
15: end for
16: return $G = (V, E)$
```

3.1.2. HVGs

The HVGs, introduced by Luque et al. in 2009 [6], builds upon the NVG previously discussed. While its goal remains the same, i.e., transforming a time series into a graph, the key difference lies in how visibility between two points in the time series is defined. In the HVG, the concept of visibility is more constrained than in the NVG, resulting in a modified approach to connect points.

In the case of the HVG, visibility must be strictly horizontal; a point that lies below any intermediate point between two other points cannot be connected. Figure 4 illustrates the distinction between the two approaches. In the NVG, we observe one additional edge compared to the HVG. This extra edge is formed due to the inclusion of non-horizontal visibilities.

Algorithm 2 Construction of the HVG

```
Require: $X = \{x_1, x_2, \ldots, x_n\}$
Ensure: Graph $G = (V, E)$
 1: Initialize $V \leftarrow \{1, 2, \ldots, n\}$
 2: Initialize $E \leftarrow \emptyset$
 3: for each pair $i, j \in V$ with $i < j$ do
 4:     Visible ← True
 5:     for $k = i + 1$ to $j - 1$ do
 6:         if $x_k \geq \min(x_i, x_j)$ then                ▷ Horizontal obstruction
 7:             Visible ← False
 8:             break
 9:         end if
10:     end for
11:     if Visible then
12:         Add edge $(i, j)$ to $E$
13:     end if
14: end for
15: return $G = (V, E)$
```

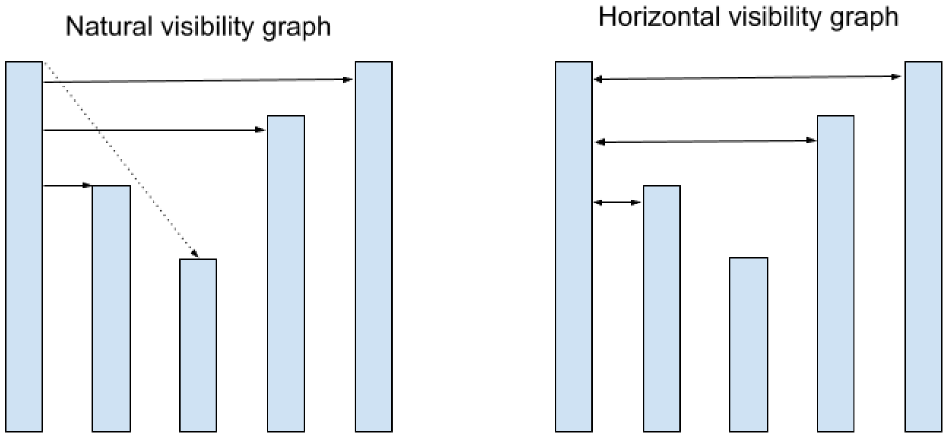

Figure 4. Difference between visibility graphs.

The algorithm assigns each data point in the time series to a node in the HVG (hereafter referred to as “the graph”). Two nodes, represented by the points (t_i, y_i) and (t_j, y_j), are connected if a horizontal line can be drawn between y_i and y_j that does not intersect any intermediate data points. Therefore, y_i and y_j are considered connected nodes if the following geometric criterion is satisfied within the time series:

3.2. Network metrics

Once we have generated both the NVG and the HVG from each time series, the next step is to extract appropriate attributes from these graphs to characterize the original time series. To achieve this, we have examined various metrics available in graph theory and ultimately select the following metrics for our analysis:

3.2.1. Max degree

The degree of a node is defined as the number of connections it has to other nodes. Specifically, the degree of node ii can be quantified as the total number of nodes connected to it. For a graph represented by the adjacency matrix A, the degree of node i can be expressed as:

$$degree_i = \sum_j a_{ij}.$$

After determining the degree of each node, we define the maximum degree of the network as follows:

$$max\_degree = \max_{ij} degree_i.$$

3.2.2. Density

The density provides a measure of how interconnected the network is among all nodes. Specifically, network density is defined as the ratio of the actual number of edges to the maximum possible number of edges in the network. For an undirected graph G with N nodes and E edges, the density can be defined as:

$$density = \frac{2E}{N(N-1)}.$$

By obtaining these features, we can represent each time series with four distinct attributes:

- HVG max degree;
- HVG graph density;
- NVG max degree;
- NVG density.

As a result of this phase, each time series can be represented as a set of four finite values that characterize its behavior.

3.3. Features clustering

After translating each time series into a finite set of attributes, we apply a traditional unsupervised clustering technique to group the time series based on these attributes.

Here, we use the K-means algorithm to facilitate comparison with other methods employing the same technique. This clustering process assigns each time series to a cluster based on the derived features.

4. Evaluation and results

To validate the proposed approach as an effective method for unsupervised time series clustering, we conducted comparisons with several of the most widely adopted algorithms in the field.

We perform this comparison across two key aspects:

- accuracy;
- performance.

The first aspect to investigate is the clustering accuracy for time series with similar behavioral patterns. We assess whether the proposed approach accurately clusters time series that exhibit similar patterns, validating its adequacy to this fundamental clustering premise.

The second aspect is computational efficiency compared to other methods. Given the increasing volume of data, computation time has become a crucial factor in selecting an unsupervised clustering system.

To benchmark our approach, we will compare it with four widely used clustering algorithms available in the Python package tslearn [16].

The tslearn library offers several clustering algorithms specifically designed for time-series data. Below are descriptions and mathematical definitions for three key unsupervised methods: KernelKMeans, K-Shape, and TimeSeriesKMeans.

Based on the 2004 research [17], KernelKMeans enhances the traditional K-means algorithm by incorporating kernel functions to capture nonlinear relationships within the data. This approach is particularly beneficial when clusters are not linearly separable in the original feature space, providing greater flexibility and improved clustering performance. KernelKMeans optimizes the objective function where ϕ is a mapping function applied through a kernel, allowing for nonlinear transformations that enhance cluster separability in the transformed space.

$$min \sum_{k=1}^{K} \sum_{i \in C_k} ||\phi(x_i) - \mu_k||^2.$$

K-Shape [18] is a shape-based clustering method designed specifically for time series and frequently used [19] and [20]. It uses a normalized cross-correlation (NCC) distance metric to capture similarities in shape between time series, making it robust against shifts and distortions. K-Shape clusters are formed by aligning time series based on this shape similarity. The objective function for K-Shape, based on maximizing cross-correlation, can be represented as:

$$\max \sum_{i=1}^{N} \sum_{j=1}^{M} \mathrm{NCC}(x_i, s_j),$$

where $\mathrm{NCC}(x_i, s_j)$ denotes the NCC between time series x_i and the cluster centroid s_j, capturing shape similarity more effectively than Euclidean distance.

K-Shape is a computational enhancement of the well-known DTW and time series forest (TSF) algorithms. Since its proposal in 2015, K-Shape has garnered 865 citations, establishing itself as a benchmark in the field of time series clustering. This algorithm effectively addresses the limitations of previous methods by focusing on shape-based clustering, making it particularly robust in identifying similarities among time series data.

TimeSeriesKMeans adapts the traditional K-means algorithm to time series by using a time-series-specific distance metric, such as DTW, instead of the Euclidean distance. DTW aligns time series by compensating for temporal distortions, allowing clusters to form based on warped similarities. The objective function, minimizing DTW-based distances within clusters, is:

$$\min \sum_{k=1}^{K} \sum_{i \in C_k} \mathrm{DTW}(x_i, \mu_k),$$

where $\mathrm{DTW}(x_i, \mu_k)$ is the DTW distance between time series x_i and the centroid μ_k of cluster C_k.

Soft DTW introduces a soft penalty for misalignment, enabling smoother and more flexible matching, making it differentiable and suitable for gradient-based optimization in machine learning contexts. This differentiability allows soft DTW to be effectively used as a loss function, enhancing its performance in applications that require model training.

These three methods, KernelKMeans, K-Shape, and TimeSeriesKMeans, provide different approaches to handling the unique challenges of time-series clustering within the tslearn library.

Basically, the techniques with which we compare our new approach are:

- K-Shape.
- Kernel K-means.
- K-means clustering for time-series data using DTW.
- K-means clustering for time-series data using soft DTW.

These algorithms are located and implemented in the tslearn library, allowing for a comparison of similar development techniques.

4.1. Dataset to compare

To compare the algorithms, we randomly selected 15 datasets as we can see in Table 1 from the UEA & UCR (University of California, Riverside) Time Series Classification Repository [21]. This repository is one of the most comprehensive and widely utilized collections for time series classification tasks, encompassing a diverse range of datasets from various domains, including health monitoring, motion tracking, and environmental data. It is specifically designed to benchmark machine learning algorithms, providing standardized datasets with varying characteristics in terms of length, dimensionality, and complexity.

The UCR Time Series Classification Archive has been referenced in over 900 scientific papers, making it an essential resource in the field of time series data mining. Since its inception, the archive has expanded significantly, growing from an initial 16 datasets to 128 in its latest version. The randomly selected 15 datasets from this repository offer a diverse and challenging set of time series data, enabling a thorough evaluation of algorithm performance across multiple domains and ensuring a robust and unbiased comparison.

Table 1. List of datasets selected from UCR.

Coffee
DistalPhalanxOutlineCorrect
ECG200
FordA
PhalangesOutlinesCorrect
ShapeletSim
ShapesAll
MoteStrain
ToeSegmentation1
Trace
TwoLeadECG
CBF
Beef
Plane
Lightning2

4.2. Comparison exercise

We applied all the algorithms to cluster the dataset into N clusters, as outlined in the UCR Time Series Classification Repository.

In this section, we present the results of applying unsupervised clustering to the 15 datasets randomly selected from the UEA & UCR Time Series Classification Repository [21]. The aim of this exercise was to evaluate the clustering performance of the selected algorithms by comparing their outcomes with the expected classification labels provided in the UCR repository. To measure the accuracy of the clustering, we utilized the Adjusted Rand Index (ARI), a robust metric that quantifies the similarity between the clustering results and the true labels present in the UCR dataset.

4.3. Experimental setup

For each dataset, unsupervised clustering was performed using various clustering algorithms. Given that the datasets are not pre-labeled for clustering, we compared the clusters generated by each algorithm with the actual labels provided in the repository, treating these labels as the ground truth. The experiments were conducted under identical conditions for all datasets to ensure a fair and consistent evaluation of the algorithms' performance.

We use the ARI as the primary evaluation metric. ARI is an adjusted version of the rand index (RI) that accounts for chance groupings, providing a value between -1 and 1, where 1 indicates perfect agreement between the clustering and the ground truth labels, and 0 indicates random clustering. Negative values suggest worse-than-random clustering.

4.4. Evaluation metric: ARI

The ARI metric [22] was selected due to its appropriateness for evaluating clustering performance in unsupervised learning. It provides a balanced measure of clustering accuracy by penalizing random groupings and is particularly effective when comparing clustering results to a known ground truth, as is the case with the UCR datasets. The ARI is calculated as follows:

$$\text{ARI} = \frac{\text{RI} - \text{expected RI}}{\text{Max RI} - \text{expected RI}},$$

where RI is the Rand Index, expected RI is the expected value of the RI, and Max RI is the maximum value that the RI can achieve. The closer the ARI is to 1, the better the clustering model matches the true labels.

4.5. *Results*

Table 2 analyzes performance using the ARI scores for each dataset and algorithm combination. Higher ARI values indicate a closer match between the algorithm's clustering results and the expected labels from the UCR repository.

Table 2. ARI scores for the algorithms.

Dataset	K-Visibility	K-Shape	TS KMeans	TS KMeans soft	kernelKmeans	Best algorithm
Coffee	0.019	0.132	0.016	-0.080	0.004	K-Shape
DistalPhalanxOutlineCorrect	0.161	-0.027	0.270	0.309	0.036	TSKMeans soft
ECG200	0.163	-0.052	0.046	0.046	-0.079	K-Visibility
FordA	-0.063	-0.063	-0.062	0.028	0.000	TSKMeans soft
PhalangesOutlinesCorrect	0.161	-0.027	0.270	0.309	0.036	TSKMeans soft
ShapeletSim	0.731	-0.065	-0.103	-0.014	0.028	K-Visibility
ShapesAll	0.406	0.352	0.886	0.607	0.000	TimeSeriesKMeans
MoteStrain	0.316	0.158	0.491	0.138	-0.097	TSKMeans
ToeSegmentation1	0.503	0.158	0.378	0.021	-0.057	K-Visibility
Trace	0.455	0.315	0.382	0.382	-0.015	K-Visibility
TwoLeadECG	0.506	0.040	0.326	-0.052	0.178	K-Visibility
CBF	0.021	0.071	0.155	0.296	0.133	TSKMeans soft
Beef	-0.014	0.133	0.160	0.114	0.000	TSKMeans
Plane	0.465	0.402	0.544	0.483	0.437	TSKMeans
Lightning2	-0.033	0.322	0.141	0.158	0.000	K-Shape

Figure 5 is a visual representation of Table 2.

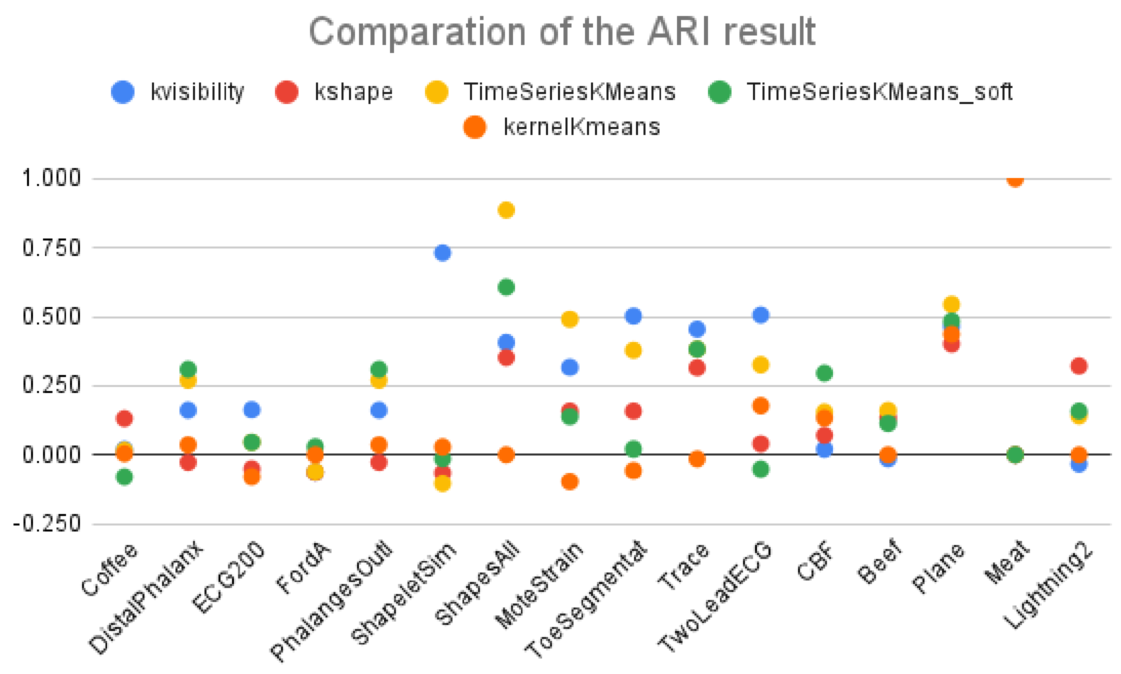

Figure 5. ARI results comparison.

The results indicate that the k-visibility algorithm achieves the highest ARI scores across the greatest number of datasets, specifically 5 out of the 15 datasets used. This demonstrates noteworthy clustering performance, as the algorithm effectively captures the underlying structure of the data. This finding suggests that k-visibility is a promising approach worth exploring further.

4.6. Computational speed analysis

In addition to evaluating clustering accuracy using the ARI, we also analyze the computational speed of each algorithm when applied to the 15 selected datasets. The efficiency of an algorithm in terms of computational time is a critical factor, particularly in real-world applications where the processing time can impact usability and scalability.

4.6.1. Experimental setup

The computational speed of each algorithm was measured as the total time taken to perform clustering on each dataset, from initialization to completion. This measurement includes the time required for converting the time series into visibility graphs as well as the clustering of the resulting features. All experiments were conducted on the same hardware to ensure consistency in the results. The runtime was recorded in seconds, and each experiment was repeated multiple times to minimize variability due to external factors.

4.6.2. Results

The results can be presented based on the two objectives we initially had when comparing k-visibility with other algorithms: speed and accuracy.

Related to accuracy, as we can appreciate in Table 3, k-visibility is the winner in 5 of the 15 datasets based in the ARI performance. This performance gives us the opportunity to consider this naive algorithm as one of the most accurate methods to cluster time series.

We have also analyzed the coherence of the clusters proposed by each of the algorithms in each of the datasets. To do this, we used the Silhouette score [23]. This score is a metric used to evaluate the quality of clustering in computer science. It measures the coherence of clusters, with a higher coefficient indicating more coherent clusters. Based on this score, Figure 6 shows how the k-visibility clustering algorithm can compete with other clustering proposals in most of the tested datasets.

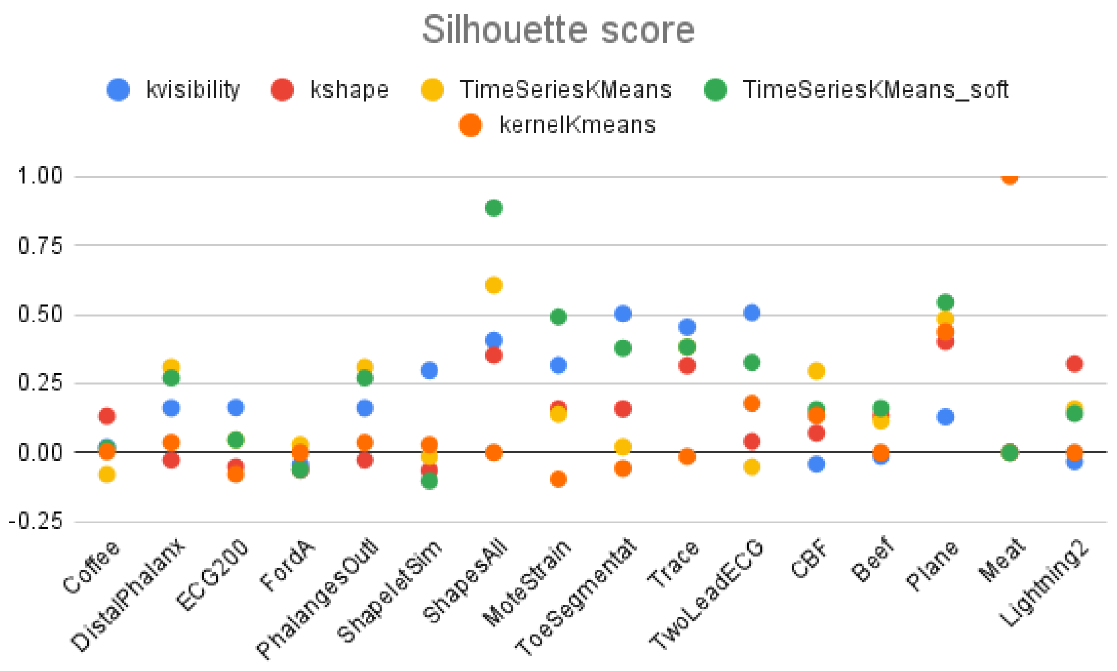

Figure 6. Silhouette score.

Table 3. Summary of best algorithm.

Algorithm	# datasets winner (out of 15)
K-Visibility	5
K-Shape	2
TimeSeriesKMeans	4
TimeSeriesKMeans_soft	4
kernelKmeans	0

Related to speed, Table 4 summarizes the average computational time (in seconds) for each algorithm across the 15 datasets.

Table 4. Algorithm performance.

Dataset	K-Visibility	K-Shape	TSKMeans	TSKMeans soft	kernelKmeans	Fastest Algorithm
Coffee	0.114	0.133	9.958	1.22	0.392	K-Visibility
DistalPhalanxOutlineCorrect	0.019	0.043	1.884	0.219	0.053	K-Visibility
ECG200	0.023	0.046	2.069	0.189	0.068	K-Visibility
FordA	0.118	1.037	32.674	1.993	1.226	K-Visibility
PhalangesOutlinesCorrect	0.039	0.080	1.875	0.143	0.0532	K-Visibility
ShapeletSim	0.0734	0.927	39.707	1.669	1.322	K-Visibility
ShapesAll	0.664	0.811	28.613	2.015	1.349	K-Visibility
MoteStrain	0.021	0.085	1.931	0.219	0.086	K-Visibility
ToeSegmentation1	0.147	0.322	11.385	0.648	0.3745	K-Visibility
Trace	0.055	0.299	9.307	0.611	0.548	K-Visibility
TwoLeadECG	0.021	0.055	1.781	0.117	0.055	K-Visibility
CBF	0.024	0.144	3.385	0.243	0.100	K-Visibility
Beef	0.218	12.549	28.140	2.229	1.049	K-Visibility
Plane	0.052	0.122	3.317	0.259	0.182	K-Visibility
Meat	0.448	0.354	34.615	1.077	1.059	K-Visibility
Lightning2	0.141	3.552	53.015	3.106	2.021	K-Visibility

The results clearly indicate that the k-visibility algorithm consistently achieved the fastest computational times across all datasets. This speed advantage is particularly pronounced with larger datasets, where other algorithms exhibited a significant increase in processing time.

Based of the features of each dataset, listed in Table 5, we can conclude several points:

Table 5. Comparison of UCR time series datasets with winning algorithm.

Dataset	Series Length	Nr. of Classes	Training Series	Test Series	Winner
Coffee	286	2	28	28	K-Shape
DistalPhalanxOutlineCorrect	80	2	600	276	K-Means
ECG200	96	2	100	100	K-Visibility
FordA	500	2	3601	1320	K-Visibility
PhalangesOutlinesCorrect	80	2	1800	858	K-Means
ShapeletSim	500	2	20	180	K-Visibility
ShapesAll	512	60	600	600	K-Means
MoteStrain	84	2	20	1252	K-Means
ToeSegmentation1	277	2	40	228	K-Visibility
Trace	275	4	100	100	K-Visibility
TwoLeadECG	82	2	23	1139	K-Visibility
CBF	128	3	30	900	K-Means
Beef	470	5	30	30	K-Means
Plane	144	7	105	105	K-Means
Lightning2	637	2	60	61	K-Means

- Low number of classes: Most of these datasets have only 2 classes, except for “Trace”, which has 4 classes. This suggests that k-visibility performs well in binary classification problems or those with a low number of classes, where the goal is to distinguish between two primary categories.
- Variety in series length: The length of the time series varies considerably, from 82 to 500. This indicates that k-visibility is adaptable to different time series lengths, making it potentially robust

to variability in the temporal extent of the data.

- Applications in medical and physical series: Some of these datasets, such as ECG200 (electrocardiogram 200) and TwoLeadECG, are related to physiological or medical signal data, while others, like FordA and ShapeletSim pertain to physical or synthetic signals. This suggests that k-visibility is versatile and can capture significant patterns in diverse applications, especially where there is periodic or trending behavior in the signals.
- Training set size: The size of the training sets varies from very small (such as ShapeletSim with only 20 series) to large (such as FordA with 3601 series). k-visibility appears to be effective with both small and large datasets, which might indicate that it does not heavily depend on the amount of training data to generalize well.
- Sensitivity to subtle changes in time series: Since k-visibility is based on graph visibility theory, it likely captures local relationships and repetitive patterns in time series. This could explain its success with datasets like ECG200 and TwoLeadECG, where small time variations are crucial for distinguishing between classes.

The k-visibility algorithm appears particularly effective for:

- Datasets with a low number of classes, especially binary classification.
- Time series with repetitive patterns or smooth trends.
- Time series with diverse lengths and training set sizes, demonstrating robustness across various data configurations.

5. Conclusions and future work

This paper presents the advantages of a novel time series clustering technique based on transforming time series into visibility graphs, enabling clustering based on graph-derived features rather than the raw time series data.

This approach achieves clustering capabilities comparable to established methods while being computationally efficient. Our initial findings pave the way for future research to enhance and validate this technique across various domains. Key areas for further study include the exploration of new metrics, beyond maximum degree and density, to improve stability and efficiency.

The next phase involves applying this method to real-world datasets with time series lacking easily recognizable patterns, which will provide a rigorous test for the algorithm. The proposed technique is particularly suitable for large temporal datasets, such as those used in financial market analysis and climate data monitoring, due to its reduced computational demands.

In summary, this paper introduces a promising time series clustering approach that leverages visibility graph structures. By drawing on the structural insights from visibility graphs, this technique provides a powerful, efficient tool for temporal data analysis, supporting deeper insights and expanding the potential applications of complex time series data.

Author contributions

Sergio Iglesias-Perez, Alberto Partida, and Regino Criado: Design, preparation and execution of the article. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This paper has been supported by INCIBE/URJC Agreement M3386/2024/0031/001 within the framework of the Recovery, Transformation and Resilience Plan funds of the European Union (Next Generation EU).

Authors Sergio Iglesias and Alberto Partida thank their professor and mentor Regino Criado for his guidance and ideas.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. S. Iglesias-Perez, R. Criado, Temporal metagraph: A new mathematical approach to capture temporal dependencies and interactions between different entities over time, *Chaos Soliton Fract.*, **175** (2023), 113940. http://dx.doi.org/10.1016/j.chaos.2023.113940
2. S. Iglesias-Perez, R. Criado, Increasing the effectiveness of network intrusion detection systems (NIDSs) by using multiplex networks and visibility graphs, *Mathematics*, **11** (2023), 107. http://dx.doi.org/10.3390/math11010107
3. L. Lacasa, B. Luque, F. Ballesteros, J. Luque, J. C. Nuno, From time series to complex networks: The visibility graph, *Proc. Natl. Acad. Sci. USA*, **105** (2008), 4972–4975. http://dx.doi.org/10.1073/pnas.0709247105
4. A. Partida, R. Criado, M. Romance, Visibility graph analysis of IOTA and IoTeX price series: An intentional risk-based strategy to use 5G for IoT, *Electronics*, **10** (2021), 2282. https://doi.org/10.3390/electronics10182282
5. J. Lopes, P. Pinto, A. Partida, A. Pinto, Use of visibility graphs for the early detection of DoS attacks, In: *2024 IEEE international conference on cyber security and resilience (CSR)*, 2024, 101–106. https://doi.org/10.1109/CSR61664.2024.10679430
6. B. Luque, L. Lacasa, F. Ballesteros, J. Luque, Horizontal visibility graphs: Exact results for random time series, *Phys. Rev. E*, **80** (2009), 046103. http://dx.doi.org/10.1103/PhysRevE.80.046103
7. G. Liu, L. Li, L. Zhang, Q. Li, S. S. Law, Sensor faults classification for SHM systems using deep learning-based method with Tsfresh features, *Smart Mater. Struct.*, **29** (2020), 075005. https://doi.org/10.1088/1361-665X/ab85a6
8. S. Aghabozorgi, A. S. Shirkhorshidi, T. Y. Wah, Time-series clustering–a decade review, *Inform. Syst.*, **53** (2015), 16–38. http://dx.doi.org/10.1016/j.is.2015.04.007
9. T. W. Liao, Clustering of time series data—a survey, *Pattern Recognit.*, **38** (2005), 1857–1874. http://dx.doi.org/10.1016/j.patcog.2005.01.025

10. S. Fröhwirth-Schnatter, S. Kaufmann, Model-based clustering of multiple time series, *J. Bus. Econ. Stat.*, **26** (2004), 78–89.
11. C. Bouveyron, J. Jacques, Model-based clustering of time series in group-specific functional subspaces, *Adv. Data Anal. Classif.*, **5** (2011), 281–300. https://doi.org/10.1007/s11634-011-0095-6
12. C. Pamminger, S. Frühwirth-Schnatter, Model-based clustering of categorical time series, *Bayesian Anal.*, **5** (2010), 345–368. https://doi.org/10.1214/10-BA606
13. M. Christ, N. Braun, J. Neuffer, A. W. Kempa-Liehr, Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package), *Neurocomputing*, **307** (2018), 72–77. http://dx.doi.org/10.1016/j.neucom.2018.03.067
14. D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, In: *Proceedings of the 3rd international conference on knowledge discovery and data mining*, 1994, 359–370.
15. A. Partida, R. Criado, M. Romance, Identity and access management resilience against intentional risk for blockchain-based IOT platforms, *Electronics*, **10** (2021), 378. https://doi.org/10.3390/electronics10040378
16. R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, et al., Tslearn, a machine learning toolkit for time series data, *J. Mach. Learn. Res.*, **21** (2020), 1–6.
17. I. S. Dhillon, Y. Guan, B. Kulis, Kernel k-means: Spectral clustering and normalized cuts, In: *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining*, 2004, 551–556. http://dx.doi.org/10.1145/1014052.1014118
18. J. Paparrizos, 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, 1855–1870. http://dx.doi.org/10.1145/2723372.2737793
19. S. Iglesias-Pérez, S. Moral-Rubio, R. Criado, A new approach to combine multiplex networks and time series attributes: Building intrusion detection systems (IDS) in cybersecurity, *Chaos Soliton Fract.*, **150** (2021), 111143. https://doi.org/10.1016/j.chaos.2021.111143
20. S. Iglesias-Pérez, S. Moral-Rubio, R. Criado, Combining multiplex networks and time series: A new way to optimize real estate forecasting in New York using cab rides, *Physica A*, **609** (2023), 128306. https://doi.org/10.1016/j.physa.2022.128306
21. H. A. Dau, A. Bagnall, K. Kamgar, C. C. Michael Yeh, Y. Zhu, S. Gharghabi, et al., The UCR time series archive, *IEEE/CAA J. Autom. Sin.*, **6** (2019), 1293–1305. http://dx.doi.org/10.1109/JAS.2019.1911747
22. M. J. Warrens, H. van der Hoef, Understanding the adjusted rand index and other partition comparison indices based on counting object pairs, *J. Classif.*, **39** (2022), 487–509. https://doi.org/10.1007/s00357-022-09413-z
23. P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *J. Comput. Appl. Math.*, **20** (1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7