# MUFASA: Fast and Accurate Multivariate Time-Series Clustering

HAOJUN LI, The Ohio State University, USA
JOHN PAPARRIZOS, The Ohio State University, USA

Time-series clustering is a core analytical technique for data exploration and as a subroutine in downstream tasks. Although deep learning methods have surged recently, studies show that traditional algorithms, such as *k*-Shape, still achieve state-of-the-art performance, revealing an illusion of progress. *k*-Shape alternates between (i) cluster assignment using the Shape-based distance (SBD) and (ii) centroid update while preserving scale and temporal alignment. However, its cubic-time centroid computation limits scalability to long sequences. Despite advances in univariate time-series (UTS) clustering, multivariate (MTS) clustering remains underexplored—multiple channels complicate inter-channel dependency modeling and amplify the accuracy–runtime trade-off. To overcome these challenges, we introduce FASA and its multivariate extension MUFASA: scalable, accurate, and parameter-light methods. FASA derives a closed-form solution that minimizes within-cluster SBD distance in linear time, while MUFASA (i) introduces SBD-D, a distance measure identifying a global temporal alignment across channels, and (ii) introduces a centroid update that jointly estimates all channels to capture temporal and inter-channel dependencies efficiently. To demonstrate the effectiveness, we conduct the most comprehensive evaluation to date in the MTS clustering area—covering seven MTS distances and 21 clustering algorithms on 30 UEA MTS datasets—and benchmark FASA on 128 UCR UTS datasets against eight baselines. Results show that (i) SBD-D matches the accuracy of elastic distances while being on average two to four orders of magnitude faster, and (ii) MUFASA outperforms *all* scalable traditional, deep learning, and foundation models in accuracy while matching non-scalable ones at much lower runtime. Notably, FASA matches *k*-Shape's accuracy while exhibiting better scalability, especially with increasing length. Overall, MUFASA and FASA deliver efficient and accurate solutions for MTS and UTS clustering, respectively, paving the way for future progress in time-series analytics.

CCS Concepts: • **Computing methodologies** → **Cluster analysis**; • **Information systems** → **Clustering**; **Similarity measures**; • **Theory of computation** → **Design and analysis of algorithms**.

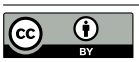
Additional Key Words and Phrases: Multivariate Time Series, Time-Series Clustering

**ACM Reference Format:**
Haojun Li and John Paparrizos. 2026. MUFASA: Fast and Accurate Multivariate Time-Series Clustering. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 213 (June 2026), 29 pages. https://doi.org/10.1145/3802090

## 1 Introduction

Sequential data mining focuses on extracting meaningful patterns and insights from large collections of ordered event sequences [39]. *Time series*, defined as sequences of time-indexed observations [38], are pervasive across domains such as biology, economics, climate science, and engineering [7, 8, 20, 22, 32, 69, 71, 72, 92, 93, 115]. Time series are classified as *univariate* time series (UTS), consisting of a single channel with scalar observations at each timestamp, or *multivariate* time series (MTS), comprising multiple channels where each timestamp is represented by a multidimensional vector of concurrent observations across channels. For example, in meteorological monitoring, temperature

Authors' Contact Information: Haojun Li, li.14118@osu.edu, The Ohio State University, Columbus, Ohio, USA; John Paparrizos, john@paparrizos.org, The Ohio State University, Columbus, Ohio, USA.




ACM 2836-6573/2026/6-ART213
https://doi.org/10.1145/3802090

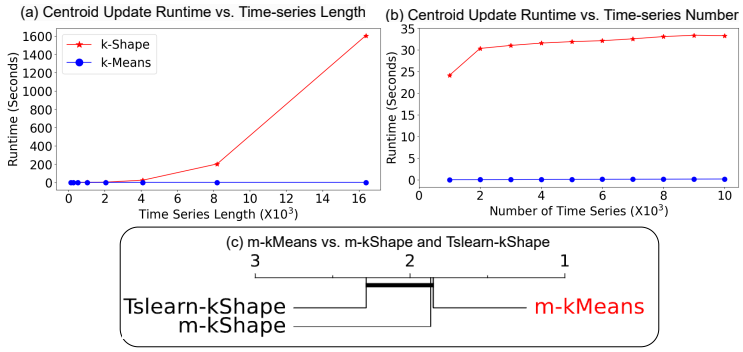



Fig. 1. The centroid-update procedure of $k$-Shape hinders scalability to long sequences, and existing multivariate $k$-Shape variants fail to outperform m-kMeans across 30 UEA datasets: centroid-update runtime comparison between $k$-Shape and $k$-Means on synthetic datasets (a) varying $L$ with $N$ = 1,000; and (b) varying $N$ with $L$ = 4,096; and (c) Critical Difference (CD) diagram with statistical test results comparing m-kMeans to existing multivariate $k$-Shape extensions, where solid lines indicate no significant difference.

and atmospheric pressure are recorded simultaneously; each forms a UTS, and together constitute an MTS capturing their joint dynamics [108].

Driven by the proliferation of massive time-series databases across diverse domains and IoT applications [44, 51, 52, 54, 59, 60, 66, 78, 88], extensive research has focused on time-series analysis techniques, including compact representations and compression strategies for managing high data volumes [44, 51, 60, 83, 110, 111], efficient similarity search frameworks enabled by robust distance measures [24, 26, 77, 82, 87, 89, 90, 94], and approaches to explainable analysis [35, 36]. Within the broader landscape of time-series analysis techniques, time-series clustering has emerged as a critical unsupervised technique for organizing sequences into meaningful groups without manual labeling or human supervision. Providing significant advantages from pattern discovery and summarization to anomaly detection [12–18, 37, 43, 46, 47, 53, 61, 63–65, 67, 75, 80, 81, 86, 101, 104, 105, 111], clustering not only serves as an independent analytical tool but also serves as a critical subroutine in various real-world time-series analytical tasks [3]. Consequently, recent decades have seen a surge in algorithmic development, transitioning from traditional methods to deep learning and foundation models [5, 9, 25, 30, 42, 55, 56, 58, 62, 74, 79, 84, 91, 95, 112–114].

Among clustering approaches, traditional partitional methods have consistently demonstrated superior performance, often leading benchmark evaluations [19, 79, 84]. Despite their effectiveness, these methods face two primary challenges: selecting a distance measure that appropriately accounts for distortions such as misalignment and noise [89] for cluster assignment, and defining a centroid that preserves the structural properties induced by that distance during refinement. Paparrizos et al. [89] showed that shape-based distance (SBD) [84], a sliding measure correcting global misalignment, offers the best accuracy–runtime trade-off. Building on SBD, $k$-Shape [84, 85] is a *scalable* traditional partitional algorithm with linear complexity in the number of series and a specialized centroid update that minimizes the sum of squared SBD within each cluster (see Section 2.1). Recent studies show that $k$-Shape continues to achieve state-of-the-art performance despite the recent emphasis on deep learning methods and foundation models, suggesting that the perceived progress may be illusory [79].

Unlike $k$-Means, whose centroid update scales linearly [67], $k$-Shape relies on eigenvalue decomposition, incurring cubic complexity relative to the time-series length $L$. We evaluate single-centroid update runtimes on synthetic data under two settings: (a) varying length $L$ with number of time series $N$ = 1,000, and (b) varying $N$ with fixed $L$ = 4,096. Figure 1 shows that $k$-Shape's runtime increases rapidly with time-series length $L$, while $k$-Means scales linearly. Although both methods

scale linearly with the number of time series $N$, $k$-Means is substantially faster. Linear scalability is critical in multivariate settings, where higher channel dimensionality amplifies computational costs and exacerbates non-linear inefficiencies.

The high dimensionality of MTS introduces substantial challenges for clustering beyond the univariate setting [55]. As dimensionality and the number of channels increase, balancing accuracy and runtime becomes more challenging. Additionally, modeling channel dependencies adds further complexity: computations can be performed per channel and aggregated (channel-independent) or jointly across all channels (channel-dependent) [100]. Many extensions from univariate clustering overlook inter-channel dependencies during centroid updates, limiting clustering quality [74, 106]. Methodological inconsistencies and insufficiently systematic attempts to extend univariate algorithms further hinder progress; for example, $k$-Shape extensions [74, 106] rely on heuristics that use only partial cluster information and do not fully exploit channel structure. As shown in Figure 1c, none of these attempts outperforms multivariate $k$-Means. Despite heavy parameter tuning and complex designs, many approaches fail to deliver consistent gains: our results (Section 6) show that recent deep learning and foundation models [6, 10, 28, 33, 34, 49, 114] require large-scale (pre)training and sophisticated architectures yet underperform traditional methods.

In this paper, we first introduce k-**FA**st**S**h**A**pe (FASA), a novel, scalable, accurate, single-parameter UTS clustering algorithm. In contrast to $k$-Shape, FASA optimizes a different within-cluster distance minimization problem, resulting in linear dependence (instead of cubic) on time-series length, and addresses $k$-Shape's limitation to long sequences. Specifically, we formulate a new optimization problem for shape-based centroid computation that enhances robustness to outliers compared to $k$-Shape and derive a closed-form solution that reduces the runtime complexity from cubic to linear in time-series length. This principled formulation enables FASA to retain the accuracy of $k$-Shape while substantially improving scalability. Next, we propose **MU**ltivariate FASA (MUFASA), a scalable and accurate algorithm tailored for MTS clustering, designed to address the aforementioned challenges in the MTS domain. MUFASA inherits FASA's advantages while introducing multivariate-specific designs to address the unique challenges of MTS clustering. Specifically, for distance computation, we develop a channel-dependent SBD (SBD-D) that determines a single global temporal alignment across channels, thereby capturing the co-evolving channel dynamics. After each MTS is aligned to the centroid via SBD-D, MUFASA develops a linear-time centroid update formulation grounded in the channel-dependent model that simultaneously estimates all channels, effectively capturing temporal and inter-channel patterns.

To rigorously evaluate MUFASA's effectiveness and efficiency, we conduct the most comprehensive empirical study to date on MTS clustering, using *all* 30 datasets in the UEA archive [7]. To our knowledge, this is the first work to jointly evaluate MTS distances, MTS clustering, and UTS clustering, while benchmarking MTS clustering at scale. Our evaluation includes (i) seven standalone multivariate distance measures spanning three major categories—lock-step, sliding, and elastic—and (ii) 21 MTS clustering algorithms across four families, including traditional scalable and non-scalable methods, deep learning–based approaches, and foundation models. In contrast, prior studies [11, 55, 56, 74] evaluate much fewer methods and typically rely on limited dataset subsets. Additionally, we evaluate FASA against eight strong baselines on all 128 UCR datasets [22], extending evaluation beyond the multivariate setting and addressing the lack of univariate validation in prior MTS studies. Our experiments yield several key findings: (i) SBD-D achieves accuracy comparable to elastic measures while being substantially more efficient and significantly outperforming Euclidean distance (ED) in terms of accuracy, offering an excellent balance between effectiveness and efficiency; (ii) MUFASA surpasses all scalable and deep learning methods in accuracy, while matching the accuracy of non-scalable methods at substantially lower computational cost; and (iii) FASA resolves the cubic centroid-update bottleneck of $k$-Shape, retaining its accuracy while

exhibiting substantially better scalability with respect to sequence length. All experiments use publicly available datasets, and we release open-source implementations and detailed settings in Section 5 to ensure reproducibility [1]. For completeness, we provide an appendix available at [2].

In this paper, we first review the theoretical foundation and related work in the clustering domain, followed by representative MTS distance measures and state-of-the-art MTS clustering algorithms (Section 2). We introduce our novel approach as follows:

- We formulate a novel optimization problem tailored for SBD, yielding a closed-form solution that achieves linear scalability with respect to time series length, and introduce FASA, a partitional clustering algorithm that integrates SBD with this new centroid update formulation (Section 3).
- We extend FASA to the multivariate case by developing distance and centroid update formulations grounded in channel-dependency models, enabling principled and scalable multivariate clustering (Sections 4.1 and 4.2).
- We introduce MUFASA, an efficient partitional clustering algorithm for MTS, which introduces SBD-D for global temporal alignment and a novel centroid update capturing temporal patterns and inter-channel correlations (Section 4.3).
- We demonstrate the robustness of our ideas through an extensive evaluation across over 150 UTS and MTS datasets and over 35 state-of-the-art baselines (Sections 5 and 6).

We conclude by discussing the implications of our work (Section 7).

## 2 Preliminaries & Related Work

In this section, we begin with the background (Section 2.1) and $k$-Shape (Section 2.2), the current state-of-the-art UTS approach. We then review related work on MTS distances (Section 2.3), followed by existing MTS clustering algorithms (Section 2.4).

### 2.1 Background

We now establish the theoretical foundations from the definition of UTS and MTS, outlining the hardness of the clustering problem. We then discuss channel-dependency models and the challenge in the modeling process of multivariate time-series clustering.

**UTS & MTS:** A UTS represents a single channel in which each timestamp corresponds to a scalar observation, whereas an MTS comprises multiple channels that evolve simultaneously, with each timestamp represented by a multidimensional vector capturing the joint evolution of synchronized univariate sequences [76]. In this paper, we define the UTS $\vec{x}$ of length $L$ as $\vec{x} = \{x_1, x_2, \ldots, x_L\}^T \in \mathbb{R}^{L\times 1}$, where $x_i$ denotes the scalar value at timestep $i$. Meanwhile, we denote the MTS $\mathbf{X}$ which has $D$ channels and each channel is a UTS with length $L$ as $\mathbf{X} = \{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(D)}\} \in \mathbb{R}^{L\times D}$.

**Channel-dependency Models [100]:** Compared to UTS, MTS introduces additional complexity due to higher dimensionality and inter-channel interactions. Therefore, two channel-dependency models are considered. The channel-independent model assumes uncorrelated channels, computing per-channel results that are subsequently aggregated. In contrast, the channel-dependent model performs joint computations by treating all channels at each timestamp as a unified multivariate entity to capture inter-channel correlations.

**Challenges of Traditional Partitional Clustering:** Traditional partitional clustering groups $N$ time series into $K$ clusters to maximize intra-cluster similarity (homogeneity) and inter-cluster dissimilarity (separation) in an unsupervised manner [3]. While various clustering criteria have been introduced to account for both homogeneity and separation [40], one of the most commonly employed criteria is the minimization of the within-cluster sum of squared distances, as it effectively captures both aspects. Let a dataset of $N$ UTS be denoted as $\mathcal{D} = \{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_N}\}$, and the number of clusters as $K$, where $K < N$. The loss function that minimizes the total within-cluster distances, partitioning the $N$ data points into $K$ disjoint clusters $C = \{C_1, C_2, \ldots, C_K\}$, can be formulated as:

$$C^* = \arg\min_{C} \sum_{j=1}^{K} \sum_{\vec{x}_i \in C_j} \text{dist}(\vec{x}_i, \vec{c}_j)^2 \tag{1}$$

where $\vec{c}_j$ is the centroid of cluster $C_j \in C$. Unfortunately, the solution is nontrivial—in Euclidean space, it is NP-hard for the number of clusters $K \geq 2$, even for a small dimension of $D = 2$ [4, 68]. Traditional clustering algorithms, such as $k$-Means [67], alleviate the problem by finding a local optimum solution using the expectation–maximization approach, which iteratively refines cluster assignments until convergence. Specifically, the algorithm (a) assigns each series to the closest centroid in Euclidean space and (b) updates the centroid as the arithmetic mean of the cluster members. Despite its linear scalability with respect to the number of time series $N$ and their length $L$, with a time complexity of $O(N \cdot K \cdot L)$ per iteration, this approach fails to account for key properties of time-series data—such as scale and shifting invariances—leading to degraded performance. In the following sections, we will review state-of-the-art solutions in this domain.

### 2.2 $k$-Shape Algorithm

**Shape-based Distance (SBD) [84]:** SBD, a sliding distance measure, utilizes coefficient-normalized cross-correlation ($NCC_c$), where the subscript $c$ denotes normalization by the norms of two UTS $\vec{x}$ and $\vec{y}$, to identify the optimal shape alignment while preserving both shift and scale invariance. Cross-correlation (CC) between two UTS $\vec{x}$ and $\vec{y}$, each of length $L$, can be efficiently computed using the Fast Fourier Transform (FFT) and Inverse FFT (IFFT), reducing the time complexity of SBD to $O(L \log(L))$. Thus, CC and SBD can be formulated as follows:

$$CC(\vec{x}, \vec{y}) = IFFT(FFT(\vec{x}) * FFT(\vec{y})) \tag{2}$$

$$SBD(\vec{x}, \vec{y}) = 1 - \max_{w} \left( \frac{CC_w(\vec{x}, \vec{y})}{\|\vec{x}\| \cdot \|\vec{y}\|} \right) \tag{3}$$

Here, $*$ denotes the complex conjugate. $CC(\vec{x}, \vec{y})$ in Equation 2 denotes the full CC sequence, whereas $CC_w(\vec{x}, \vec{y})$ in Equation 3 denotes the CC evaluated at shift $w$, where $w$ represents the optimal position that maximize $NCC_c(\vec{x}, \vec{y})$.

**Centroid Update with Shape Extraction [84]:** $k$-Shape is a shape-based UTS clustering method that employs a scalable iterative refinement process similar to the $k$-Means algorithm but with significant differences. Specifically, $k$-Shape utilizes SBD as its distance measure and updates the centroid $\vec{c}_j$ to a new centroid $\vec{c}_j{}^*$ using the UTS $\vec{x}_i$ within cluster $C_j$, formulated as follows:

$$\vec{c}_j{}^* = \underset{\vec{c}_j}{\text{argmax}} \sum_{\vec{x}_i \in C_j} NCC_c(\vec{x}_i, \vec{c}_j)^2 \tag{4}$$

Since SBD has already aligned the time series of a cluster against the cluster's centroid before centroid computation, the denominator of $NCC_c(\vec{x}_i, \vec{c}_j)$ can be omitted, simplifying it to $\vec{x}_i{}^T \cdot \vec{c}_j$ and leading to Equation 5, where $S = \sum_{\vec{x}_i \in C_j} \left( \vec{x}_i \cdot \vec{x}_i{}^T \right)$. Additionally, we substitute $\vec{c}_j$ in Equation 5 with $Q \cdot \vec{c}_j$, where $Q = I - \frac{1}{T}O$, $I$ is the identity matrix, and $O$ is a matrix of ones, ensuring that $\vec{c}_j$ is centered. To enforce a unit norm for $\vec{c}_j$, Equation 5 is further normalized by $\vec{c}_j{}^T \cdot \vec{c}_j$. By letting $M = Q^T \cdot S \cdot Q$, the problem is reformulated as the maximization of the Rayleigh Quotient:

$$\vec{c}_j{}^* = \underset{\vec{c}_j}{\text{argmax}}\ \vec{c}_j{}^T \cdot S \cdot \vec{c}_j = \underset{\vec{c}_j}{\text{argmax}}\ \frac{\vec{c}_j{}^T \cdot M \cdot \vec{c}_j}{\vec{c}_j{}^T \cdot \vec{c}_j} \tag{5}$$

The closed-form solution of $\vec{c}_j{}^*$ is the eigenvector that corresponds to the largest eigenvalue of $M$, resulting in cubic time complexity to the time-series length $L$. In general, $k$-Shape requires $O(\max\{N \cdot K \cdot L \cdot \log(L), N \cdot L^2, K \cdot L^3\})$ time per iteration, where $N$ denotes the number of time series and $K$ is the number of clusters.

### 2.3 MTS Distance Measures

Designing MTS distance measures is substantially more challenging than for univariate sequences. Beyond handling temporal distortions such as misalignment and noise, MTS distance measures must account for inter-channel dependencies, which substantially increase both methodological complexity and computational cost [24]. Consequently, MTS distance measures must jointly address temporal invariance and cross-channel structure, motivating approaches that either extend univariate methods or introduce new formulations.

With the goal of establishing a unified, empirically grounded picture of measure performance and trade-offs across multivariate categories, d'Hondt et al. [24] evaluate 30 multivariate distance measures spanning eight categories and two channel-dependency models under supervised and unsupervised regimes, with an emphasis on classification and preliminary evidence for clustering and anomaly detection tasks. The study provides practical guidelines for tasks that require temporal distortion correction, such as classification: elastic measures trade runtime for accuracy, whereas lock-step measures trade accuracy for efficiency; in contrast, sliding measures offer the best accuracy–runtime balance, consistent with the findings of Paparrizos et al. [89] in the univariate setting. For anomaly detection, where distortions are typically not corrected, lock-step measures remain the most effective and efficient choice. Motivated by these findings, the experimental analysis in this work concentrates on the discriminative power of representative measures from the lock-step, sliding, and elastic categories (see Section 6.1).

### 2.4 MTS Clustering Algorithms

MTS clustering algorithms can be categorized based on their structural approach into traditional methods, which rely on statistical models, similarity measures, or optimization techniques, and deep learning-based methods, which utilize neural networks [95].

**Traditional Clustering Algorithms:** Traditional clustering algorithms for MTS rely on statistical methods rather than deep neural networks (DNNs). These approaches can be broadly categorized into scalable and non-scalable methods. Scalable methods, such as the multivariate extensions of partitional algorithms [74, 106], aim to extract temporal patterns by adapting univariate techniques for each channel independently, reducing complexity to linear time; however, existing extensions often overlook channel dependencies or suffer from performance degradation. For example, two existing extensions of $k$-Shape to the multivariate setting fail to outperform multivariate $k$-Means, as shown in Figure 1c. To address this limitation, clustering approaches like MC2PCA [55] and Time2Feat [11] incorporate cross-channel characteristics by projecting original time series to feature space or latent space. However, such a projection or extraction process in the training stage may suffer from high space complexity on large datasets. Non-scalable methods, including PAM [53], FCFW [56], and T-GMRF [25], rely on distance matrices or complex representations, often involving quadratic or cubic time complexity with respect to dataset size, which limits their applicability to large-scale scenarios in real-world applications.

**Deep Learning–based Clustering Algorithms and Foundation Models:** Deep learning–based clustering and foundation models fall into two groups: (a) end-to-end architectures that directly output cluster assignments, and (b) representation learners followed by traditional clustering [95]. End-to-end methods such as IDEC [34] and SDCN [10] couple autoencoders with clustering or structural objectives to jointly optimize features and assignments. Representation-learning approaches, including USRL [28] and DeTSEC [49], extract embeddings via convolutional or recurrent encoders, then apply standard clustering. Recently, foundation-model–based time-series analysis is an emerging area, yet clustering remains underexplored, with no dedicated models or standard adaptation protocols. A common approach [79] leverages foundation models – such as OFA [114], MOMENT [33], and Chronos [6] – for zero-shot representation learning via large-scale pretraining,

producing task-agnostic embeddings that are subsequently clustered using conventional methods (e.g., $k$-Means). Despite their prominence, studies [79] show that classical algorithms such as $k$-Shape still achieve state-of-the-art results for univariate time-series clustering, indicating that some perceived advances reflect illusory progress. While current work largely targets on univariate cases [79], multivariate time-series clustering has yet to receive systematic empirical evaluation. To test these conclusions and compare representative methods in the multivariate setting, we report accuracy and runtime in Sections 6.4 and 6.5.

## 3 Proposed UTS Clustering Method: FASA

In this section, we first formulate a novel optimization problem for centroid update and mathematically derive a closed-form solution with linear complexity in the length of UTS, addressing a long-standing limitation of the award-winning $k$-Shape method. Next, we introduce FASA, a scalable univariate clustering algorithm that combines SBD with our proposed centroid update to achieve highly efficient and competitive performance.

**FASA Centroid Computation:** As previously discussed in Section 2.2, $k$-Shape integrates SBD into its objective function and applies eigenvalue decomposition to extract shape-preserving centroids. The cubic complexity of eigenvalue decomposition with respect to the length of UTS leads to prohibitively high runtime as the time-series length increases (Figure 1a).

To reduce the high complexity of centroid update while preserving the representative shape of a cluster, we formulate centroid computation as an optimization problem that seeks the centroid minimizing the total distance to all UTS $\vec{x}_i$ within the cluster $C_j$. However, since $NCC_c$ measures similarity between time series, we reformulate the problem as a maximization task to determine the optimal centroid $\vec{c}_j^{\,*}$, yielding Equation 6. We then expand $NCC_c$ by expressing the CC normalized by the norms of $\vec{x}_i$ and $\vec{c}_j$, leading to the reformulation in Equation 7, as shown below.

$$\vec{c}_j^{\,*} = \arg\max_{\vec{c}_j} \sum_{\vec{x}_i \in C_j} NCC_c(\vec{x}_i, \vec{c}_j) \tag{6}$$

$$= \arg\max_{\vec{c}_j} \sum_{\vec{x}_i \in C_j} \frac{max_w\, CC_w(\vec{x}_i, \vec{c}_j)}{\|\vec{x}_i\| \cdot \|\vec{c}_j\|} \tag{7}$$

Equation 7 necessitates determining the optimal global shift $w$ for each $\vec{x}_i \in C_j$. Given that this process is embedded within iterative clustering, we leverage the previously computed centroid as a reference and align all sequences accordingly. Furthermore, since the sequences are already aligned to a reference prior to centroid computation, Equation 7 can be simplified as follows:

$$\vec{c}_j^{\,*} = \arg\max_{\vec{c}_j} \sum_{\vec{x}_i \in C_j} \vec{x}_i^{\,T} \cdot \vec{c}_j \tag{8}$$

This formulation is more robust to outliers than the original $k$-Shape optimization problem and necessitates a fundamentally different solution approach. To ensure the centering of $\vec{c}_j$, we apply the transformation $Q \cdot \vec{c}_j$, where $Q = I - \frac{1}{T}O$, with $I$ as the identity matrix and $O$ as a matrix of ones. Although storing $Q$ becomes expensive for very long time series due to its large space consumption, mathematically, we derive $\vec{x}_i^{\,T} \cdot Q = \vec{x}_i^{\,T} - \bar{x}_i \cdot \mathbf{1}^T$, where $\bar{x}_i$ is the mean of $\vec{x}_i$ and $\mathbf{1}^T$ is a $1 \times L$ row vector of ones to ensure correct broadcasting. Therefore, we directly set $\vec{x}_i^{\,T} = \vec{x}_i^{\,T} - \bar{x}_i \cdot \mathbf{1}^T$. Additionally, to ensure that $\vec{c}_j$ has a unit norm, we impose the constraint $\|\vec{c}_j\|^2 = 1$. Under these conditions, the Lagrangian formulation is given by:

$$\mathcal{L}(\vec{c}_j, \lambda) = \sum_{\vec{x}_i \in C_j} \vec{x}_i^{\,T} \cdot \vec{c}_j - \lambda(\|\vec{c}_j\|^2 - 1) \tag{9}$$

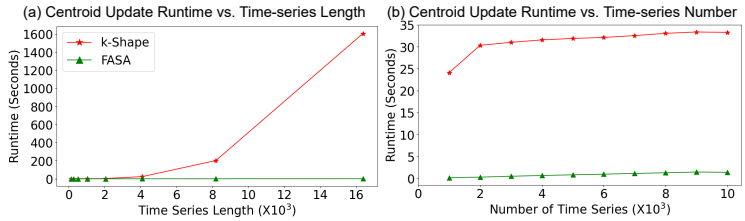



Fig. 2. Comparison of centroid-update runtime between $k$-Shape and FASA on synthetic datasets: (a) varying $L$ with $N = 1000$; and (b) varying $N$ with $L = 4096$.

where $\lambda$ is the Lagrange multiplier. To solve this problem, we take the derivative with respect to each parameter and set the resulting equations to zero, as shown below:

$$\frac{\partial \mathcal{L}}{\partial \vec{c_j}} = \sum_{\vec{x_i} \in C_j} \vec{x_i} - 2\lambda \vec{c_j} = 0; \quad \frac{\partial \mathcal{L}}{\partial \lambda} = -\|\vec{c_j}\|^2 + 1 = 0 \tag{10}$$

Using Equation 10, we express $\vec{c_j}$ in terms of $\lambda$ and derive $\lambda$ using the constraint $\|\vec{c_j}\|^2 = 1$, as shown in Equation 11. Finally, we substituting $\lambda$ into the expression of $\vec{c_j}$ yields the closed-form solution for $\vec{c_j}^*$ in Equation 12:

$$\vec{c_j} = \frac{1}{2\lambda} \sum_{\vec{x_i} \in C_j} \vec{x_i}; \ \lambda = \frac{1}{2} \| \sum_{\vec{x_i} \in C_j} \vec{x_i} \| \tag{11}$$

$$\vec{c_j}^* = \frac{1}{\| \sum_{\vec{x_i} \in C_j} \vec{x_i} \|} \sum_{\vec{x_i} \in C_j} \vec{x_i} \tag{12}$$

This centroid update algorithm eliminates the cubic computation in $k$-Shape, achieving linear complexity in both the number and length of time series while preserving the ability to extract the representative shape under scale, translation, and shifting invariances. Figure 2 presents a comparison of centroid update runtimes between $k$-Shape and FASA under the same settings described in Section 1. In these figures, FASA's centroid update runtime exhibits linear scaling with respect to $L$ and is two to three orders of magnitude faster than $k$-Shape's update. The next section discusses the integration of SBD and the proposed centroid update strategy into an iterative refinement process, resulting in the development of FASA, our proposed UTS clustering algorithm.

**FASA Clustering Algorithm:** Following the concept of $k$-Shape, FASA minimizes the sum of distances (Equation 1), ensuring (a) well-separated clusters and (b) linear scalability with the number of time series. While FASA employs SBD like $k$-Shape, its centroid update strategy (Equation 12) greatly reduces to linear dependence (from cubic complexity) on time-series length. Compared to $k$-Means, FASA significantly outperforms it while achieving performance comparable to $k$-Shape but with faster execution. This efficiency becomes even more critical in the multivariate extension discussed later in MUFASA (Section 4), as each additional MTS channel amplifies the number of calculations and inefficiency in runtime.

FASA takes as input the UTS set $\mathcal{D}$ and the desired number of clusters $K$, and initializes by randomly assigning time series to clusters. The algorithm then iterates through two stages until convergence or until reaching a small, commonly chosen maximum number of iterations, such as 100: (a) the centroid update stage, where centroids are computed using the method described above; and (b) the assignment stage, which assigns each series to the nearest cluster centroid based on SBD. Since this paper focuses on MTS clustering and adapting the multivariate algorithm to the univariate case is straightforward, we present only the MUFASA pseudo-code (Algorithm 3) in the main text and defer the FASA pseudo-code to Section B of the appendix due to space constraints.

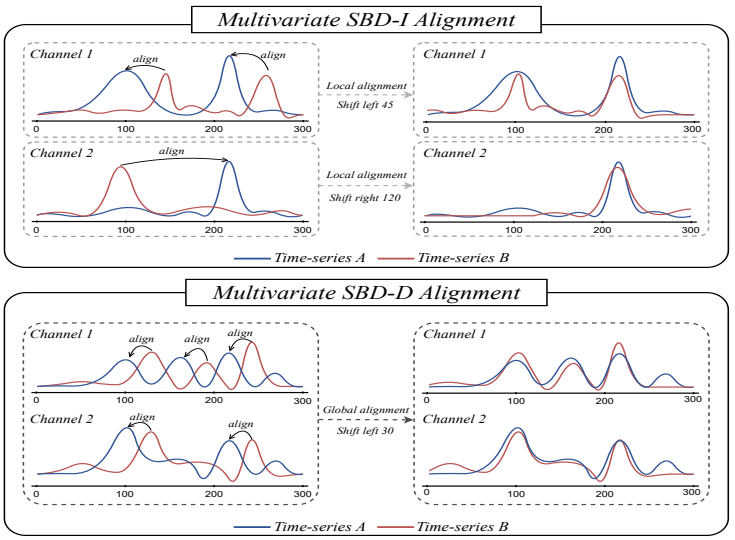



Fig. 3. Multivariate SBD-I alignment vs. SBD-D alignment.

**Complexity of FASA:** We denote $N$ as the number of time series, $K$ as the number of clusters, and $L$ as the length of the time series. In the assignment step, FASA computes the dissimilarity of $N$ time series with respect to the $K$ centroids using SBD, with a complexity of $O(L \log L)$, resulting in a total time complexity of $O(N \cdot K \cdot L \cdot \log L)$. In the refinement step, for every cluster, we centralize each time series by subtracting its mean, sum the centralized time series within the cluster, and divide the summation by the norm of the summation. Thus, the total time complexity is $O(N \cdot L + K \cdot L)$, which simplifies to $O(N \cdot L)$ since, in real-world cases, $N \gg K$. This aligns with our previous statement that FASA's centroid update scales linearly with respect to both $N$ and $L$. Overall, FASA requires $O(\max\{N \cdot K \cdot L \cdot \log L, N \cdot L\}) = O(N \cdot K \cdot L \cdot \log L)$ time per iteration. This confirms that FASA scales linearly with the number of time series, and its runtime complexity is primarily determined by the assignment step. Compared to $k$-Shape, FASA reduces the cubic complexity dependency on $L$ found in $k$-Shape's centroid update, making FASA two to three orders of magnitude faster and better suited for large-scale time-series datasets (see Figure 13).

## 4 Proposed MUFASA Clustering Algorithms

In this section, we generalize FASA to the multivariate domain by formulating a channel-dependent distance and centroid update mechanisms, enabling principled and scalable clustering (Sections 4.1 and 4.2). Subsequently, we present MUFASA, a scalable and accurate multivariate time-series clustering algorithm which enables efficient modeling temporal patterns and capturing co-evolving channel dynamics with a closed form solution (Section 4.3).

### 4.1 Multivariate SBD Extensions

In this section, we introduce and formulate the multivariate SBD from two channel-dependency models: (a) the channel-independent model (denoted by -I after the measure), which assumes no correlation between channels and applies a univariate measure to each channel separately before aggregating the results; and (b) the channel-dependent model (denoted by -D after the measure), which treats all channels as a cohesive unit while incorporating inter-channel correlations. Unlike the simple extension of the channel-independent model, the channel-dependent model requires a carefully designed measure to fully leverage its potential.

Since the channel-independent model serves as a straightforward extension, we first introduce SBD-I. Extending SBD, SBD-I disregards potential relationships and interactions across different

channels of MTS. Instead, it treats each channel of two MTS, $\mathbf{X} = \{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(D)}\}$ &$\mathbf{Y} = \{\vec{y}^{(1)}, \vec{y}^{(2)}, \ldots, \vec{y}^{(D)}\} \in \mathbb{R}^{L \times D}$, independently. Here, $D$ represents the number of channels, each of length $L$. Channel-independent approach allows each channel to determine its optimal alignment without being influenced by others. Due to the simplicity of the method, the corresponding formulas are omitted. The top figure in Figure 3 illustrates the motivation behind designing SBD-I. This approach is well-suited for scenarios where optimal global alignment requires different shifting directions for each channel, providing additional flexibility.

Unlike SBD-I, the design of SBD-D draws inspiration from pattern matching between two 2-dimensional objects, where one serves as a template while the other undergoes all possible shifts to identify the position yielding the highest matching score. However, given the significance and uniqueness of MTS channels, our approach differs from traditional pattern matching by restricting shifts exclusively along the temporal axis while prohibiting shifts across channels. SBD-D treats all dimensions of an MTS as a unified entity, ensuring that alignment operations are performed collectively across all channels. The following equations define SBD-D. Equation 13 computes the full 2-dimensional CC (CC2) sequence $CC2(\mathbf{X}, \mathbf{Y})$ using 2-D FFT and IFFT, denoted as FFT2 and IFFT2. In Equation 14, the operator $\max_T$ restricts the maximization to the temporal axis only; accordingly, $CC2_w(\mathbf{X}, \mathbf{Y})$ denotes the CC2 value evaluated at temporal shift $w$, where $w$ is the shift that yields the best temporal alignment while treating all channels as a unified entity. It is worth noting that trivial solutions can do matches yet *ignore* inter-channel correlation completely if the max is not on the temporal axis range. We show the formula of SBD-D as follows.

$$CC2(\mathbf{X}, \mathbf{Y}) = IFFT2(FFT2(\mathbf{X}) * FFT2(\mathbf{Y})) \tag{13}$$

$$SBD\text{-}D(\mathbf{X}, \mathbf{Y}) = 1 - \max_w T\left(\frac{CC2_w(\mathbf{X}, \mathbf{Y})}{\|\mathbf{X}\| \cdot \|\mathbf{Y}\|}\right) \tag{14}$$

The bottom figure in Figure 3 illustrates the concept of SBD-D, where all channels shift together by the same amount to achieve optimal global alignment as a unified entity. While SBD-I offers greater flexibility by allowing independent shifts for each channel, excessive focus on alignment within individual channels while neglecting global cross-channel relationships can lead to overfitting (see Table 1). Thus, SBD-D, which ensures global shape alignment across all channels and reduces overfitting, is as essential as SBD-I. Since SBD-I is a straightforward extension of SBD that aggregates the computed SBD for each channel, we present only the pseudo-code for SBD-D in Algorithm 1. This algorithm takes two MTS, $\mathbf{X}$ and $\mathbf{Y}$, and returns both their distance based on Equation 14 and the aligned sequence of $\mathbf{Y}$ along the temporal axis. The aligned sequence is padded along the temporal axis to ensure consistency in the MTS shape, with the padding applied using a zero-filled matrix obtained via the *zeros* function with a given shape, resulting in $\mathbf{Y}'$. In Section 6.1, we conduct an evaluation of representative distance measures from lock-step, sliding, and elastic categories.

### 4.2 Multivariate FASA Centroid Update

We first introduce the channel-independent FASA centroid update. Similar to SBD-I, this method extracts the representative shape of a cluster by applying Equation 12 to each channel independently. The formulas are omitted for brevity. This approach enables each channel to independently capture its representative shape without being influenced by cross-channel interference—ideal for scenarios with weak or nonexistent inter-channel correlations. For instance, if the channels of each MTS within a cluster are weakly connected and collected by sensors, a malfunctioning sensor producing random signals will not affect the centroid extraction for the unaffected channels under the channel-independent strategy. This allows for reliable centroid representation and facilitates the identification of the faulty sensor through comparison.

**Algorithm 1** $[dist, \mathbf{Y}']$ = SBD-D($\mathbf{X}, \mathbf{Y}$)

**Input:** Two *Z-score* normalized MTS $\mathbf{X}$ and $\mathbf{Y} \in \mathcal{R}^{L \times D}$
**Ouput:** Dissimilarity $dist$ of $\mathbf{X}$ and $\mathbf{Y}$
Aligned sequence $\mathbf{Y}'$ of $\mathbf{Y}$ towards $\mathbf{X}$
$L, D = length(\mathbf{X}), channel(\mathbf{X})$
$shape = (2^{nextpower2(2 \cdot L-1)}, 2^{nextpower2(2 \cdot D-1)})$
$CC2 = IFFT2\{FFT2(\mathbf{X}, shape) * FFT2(\mathbf{Y}, shape)\}$ ▷ Equation 13
$NCC2 = \frac{CC2}{\|\mathbf{X}\| \cdot \|\mathbf{Y}\|}$
$[value, index] = \max T(NCC2)$
$dist = 1 - value$ ▷ Equation 14
$shift = index - L$
**if** $shift \geq 0$ **then**
    $\mathbf{Y}' = [zeros(shift, D); \mathbf{Y}(1 : L - shift, :)]$
**else**
    $\mathbf{Y}' = [\mathbf{Y}(1 - shift : L, :); zeros(-shift, D)]$
**end if**

While the primary focus of extending the centroid update strategy has been on the channel-independent model, we now introduce the channel-dependent model for FASA's centroid update. This approach adopts a holistic formulation by unifying the multivariate sequence into a structurally integrated entity to facilitate a linear centroid optimization that jointly estimates all channels, effectively capturing both temporal dynamics and inter-channel dependencies. The following equation illustrates this concept:

$$\vec{\mathbf{c}}_j^{\,*} = \frac{1}{\left\| \sum_{\vec{\mathbf{X}}_i \in C_j} \vec{\mathbf{X}}_i \right\|} \sum_{\vec{\mathbf{X}}_i \in C_j} \vec{\mathbf{X}}_i, \tag{15}$$

where $\vec{\mathbf{X}}_i = [\vec{x}_i^{(1)\top}, \vec{x}_i^{(2)\top}, \ldots, \vec{x}_i^{(D)\top}]^\top \in \mathbb{R}^{(L \cdot D) \times 1}$ After obtaining the new channel-concatenated centroids, it is crucial to reverse the concatenation and reshape them back into MTS format. Since SBD determines global alignment by considering all shifts, failing to restore the original structure could lead to unintended cross-channel misalignment (e.g., channel i aligning with channel j). The channel-dependent FASA centroid update considers all channels as an entity, making it especially effective for MTS with strong inter-channel correlations. Moreover, this approach enhances robustness against minor variations within individual channels. For example, noise in an MTS channel may cause a channel-independent model to overfit, degrading centroid quality, whereas a channel-dependent model jointly considers all channels, mitigating such distortions and yielding more stable centroids. Appendix Section C illustrates the advantage of the channel-dependent centroid update over channel-independent updates by jointly considering all channels, thereby suppressing noise from unreliable channels. Our channel-dependent centroid update strategy for extending FASA is not universally applicable. For example, it is unsuitable for $k$-Shape, where concatenating MTS channels greatly increases the sequence length, leading to a dramatic runtime increase due to its cubic complexity. However, FASA's centroid update scales linearly, making this extension feasible. Adapting such strategies to other algorithms requires careful consideration, but this lies beyond the scope of this paper.

Algorithm 2 presents the pseudo-code for the channel-dependent FASA centroid update extension. We omit the pseudo-code for the channel-independent extension, as it is a straightforward adaptation that applies FASA's centroid update independently to each channel and then combines them to form the updated centroid. In Algorithm 2, lines 7–10, each MTS within the cluster is aligned to the current centroid, and its channels are jointly estimated. It is important to emphasize that, in our proposed MUFASA, line 8 is not part of the centroid update stage. Since MUFASA's

**Algorithm 2** $\mathbf{c}' = FASACentroid\text{-}D(\mathcal{D}, \mathbf{c})$

1: **Input:** $\mathcal{D}$ is a matrix $\in \mathcal{R}^{N \times L \times D}$ with Z-score normalized MTS.
2: $\mathbf{c}$ is a matrix $\in \mathcal{R}^{L \times D}$ with the current centroid.
3: **Output:** $\mathbf{c}'$ is a matrix $\in \mathcal{R}^{L \times D}$ with the new centroid.
4: $\mathcal{D}' = [\ ]$
5: $\mathbf{I} = ones(L \cdot D, 1)$
6: $\mathbf{c}' = zeros(L \cdot D, 1)$
7: **for** $i \leftarrow 1$ to $N$ **do**
8: $[dist, \mathbf{X}'] = SBD\text{-}D(\mathbf{c}, \mathcal{D}[i])$ ▷ Algorithm 1
9: $\mathcal{D}' = [\mathcal{D}'; channelconcate(\mathbf{X}')]$
10: **end for** ▷ $\mathcal{D}' \in \mathcal{R}^{N \times (L \cdot D) \times 1}$
11: $\mathcal{D}' = zscore(\mathcal{D}', axis = 1)$
12: **for** $i \leftarrow 1$ to $N$ **do**
13: $\mathbf{c}' = \mathbf{c}' + (\mathcal{D}'[i] - mean(\mathcal{D}'[i]) \cdot \mathbf{I})$
14: **end for**
15: $\mathbf{c}' = \frac{\mathbf{c}'}{\|\mathbf{c}'\|}$ ▷ Equation 15
16: $\mathbf{c}' = reshapeMTS(zscore(\mathbf{c}'))$

assignment step aligns MTS within a cluster to its centroid and provides the aligned results as input for centroid updating, we retain line 8 solely for clarity. In line 11, SBD-D applies shifting and zero-padding, which disrupt normalization; thus, re-normalization restores comparability. The centroid is then computed from the aligned sequences (lines 12–15). Although mean subtraction in line 13 is redundant under Z-score normalization, it is retained for consistency with the general-case induction in Section 3. In line 16, the centroid is re-normalized to match the input scale, ensuring representational consistency. Since aligned MTS are reshaped into a UTS representation, normalization (lines 11 and 16) and centroid computation (lines 12–15) operate globally across channels, after which the centroid is reshaped back to MTS.

### 4.3 MUFASA Clustering Algorithm

We now introduce MUFASA, an efficient and accurate algorithm for MTS clustering that leverages the SBD-D distance measure from Section 4.1 and the channel-dependent centroid update strategy from Section 4.2 to efficiently and effectively cluster MTS.

**MUFASA Clustering Algorithm:** MUFASA extends FASA to the multivariate setting, incorporating correlations between channels throughout its iterative refinement process. Specifically, MUFASA iteratively applies SBD-D to measure distances and assign MTS to the nearest cluster and updates the centroids using the channel-dependent FASA centroid update extension until convergence. As a result, MUFASA achieves the highest performance among scalable algorithms and significantly outperforms scalable baselines.

MUFASA (Algorithm 3) takes as input the MTS set $\mathcal{D}$ and the number of clusters $K$. The process begins with random cluster assignment and initializes all centroids as zero matrices. Then, it computes each cluster centroid using Algorithm 2. After obtaining the centroids, MUFASA reassigns each MTS to the cluster with the closest centroid based on SBD-D (Algorithm 1). It alternates between assignment and refinement until convergence or a predefined maximum number of iterations (typically 100) is reached. Convergence triggers early termination, while the iteration limit serves solely as a conservative upper bound, following common practice in [79, 84], where a small iteration limit (e.g., 100) is used to balance convergence stability and runtime. The algorithm outputs the final cluster assignments and the centroid of each cluster.

**Complexity of MUFASA:** We adopt the same notation from Section 3. In the assignment step, MUFASA computes the dissimilarity between $N$ MTS and $K$ clustering centroids using SBD-D, which has a complexity of $O(L \cdot D \cdot \log(L \cdot D))$, with $D$ multivariate time-series channels. Thus, the

**Algorithm 3** $[IDX, \mathbf{C}] = MUFASA(\mathcal{D}, K)$

**Input:** $\mathcal{D}$ is a matrix $\in \mathcal{R}^{N \times L \times D}$ with Z-score normalized MTS.
  $K$ is the number of clusters to produce.
**Output:** $IDX$ is an vector $\in \mathcal{R}^{N \times 1}$ containing the assignment of $N$
  time series to $K$ clusters (initialized randomly).
  $\mathbf{C}$ is a matrix $\in \mathcal{R}^{K \times L \times D}$ containing $K$ centroids.
$iter = 0$
$IDX' = [\ ]$
**while** $IDX \neq IDX'$ and $iter < 100$ **do**
  $IDX' = IDX$
  **for** $j = 1$ to $K$ **do** ▷ Refinement step
    $\mathcal{D}' = [\ ]$
    **for** $i \leftarrow 1$ to $N$ **do**
      **if** $IDX[i] == j$ **then**
        $\mathcal{D}' = [\mathcal{D}'; \mathcal{D}[i]]$
      **end if**
    **end for**
    $\mathbf{C}[j] = FASACentroid\text{-}D(\mathcal{D}', \mathbf{C}[j])$ ▷ Algorithm 2
  **end for**
  **for** $i \leftarrow 1$ to $N$ **do** ▷ Assignment step
    $Dists = [\ ]$
    **for** $j \leftarrow 1$ to $K$ **do**
      $[dist, \mathbf{X}'] = SBD\text{-}D(\mathbf{C}[j], \mathcal{D}[i])$ ▷ Algorithm 1
      $Dists = [Dists; dist]$
    **end for**
    $IDX[i] = \arg\min(Dists)$
  **end for**
  $iter \leftarrow iter + 1$
**end while**

total runtime complexity for this step is $O(N \cdot K \cdot L \cdot D \cdot \log(L \cdot D))$. In the refinement step, MUFASA applies a channel-dependent extension of FASA's centroid update, where simultaneously estimates all channels effectively capturing the patterns in temporal and channel dimensions. The total complexity of the centroid update step is $O(N \cdot L \cdot D)$. Overall, the per-iteration time complexity of MUFASA is $O(\max\{N \cdot K \cdot L \cdot D \cdot \log(L \cdot D), N \cdot L \cdot D\}) = O(N \cdot K \cdot L \cdot D \cdot \log(L \cdot D))$, demonstrating its linear scalability with respect to $N$.

We now present the experimental evaluation of multivariate SBD, followed by MUFASA against SOTA clustering algorithms, along with a comparative analysis of FASA and representative univariate partitional clustering algorithms as complementary information.

## 5 EXPERIMENTAL SETTINGS

In this section, we outline the experimental setup used to systematically evaluate both the effectiveness and computational efficiency of our multivariate SBD, MUFASA, and FASA.

**Datasets:** We evaluate UTS clustering performance on the UCR archive [22], the largest publicly available collection of labeled UTS datasets, consisting of 128 datasets spanning synthetic and real-world domains. These datasets contain between 40 and 24,000 time-series samples, with time-series lengths ranging from 15 to 2,844. To evaluate multivariate SBD and MUFASA, we leverage the UEA archive [7], the largest repository of labeled MTS datasets for classification and clustering, comprising 30 real-world datasets from diverse domains with varying time-series lengths and channels (see Appendix A Table 1 and 2 for additional details). Moreover, the UCR and UEA archives are widely used as the standard basis for evaluation in the literature [11, 24, 45, 79, 84], thereby making our evaluation both comprehensive and reliable. Considering the high computational cost

of non-scalable methods, which could not complete on a substantial portion of the original datasets within seven days, we follow common practice adopted in the community [24] and construct a downsampled UEA archive. Specifically, datasets exceeding 500 samples in the training/testing sets, 15 channels, or 500 time points per MTS are downsampled along the corresponding dimensions using stratified sampling, random sampling, and polyphase resampling [41] respectively. As a result, 21 out of 30 datasets are downsampled to enable a complete comparison. Detailed per-dataset statistics, including original and reduced dimensions and the applied downsampling strategies, are provided in the appendix A. Following the literature [24], we evaluate multivariate SBD using a one-nearest-neighbor (1NN) classifier on the predefined train-test split. In contrast, for FASA and MUFASA, we merge the train and test sets as clustering is an unsupervised task. Following previous studies [7, 84], we resample time series to the maximum length within each dataset and apply linear interpolation for missing values. To handle varying scales across temporal and channel dimensions, we perform Z-score normalization [79, 84].

**Platform:** Experiments were conducted on a server with dual AMD EPYC 7713 CPUs (64-core), 1 TB RAM, and dual 80 GB NVIDIA A100 GPUs running Ubuntu 22.04.3 LTS. To ensure a fair comparison, all CPU-based algorithms were executed in a single-threaded environment without acceleration libraries. For DNNs and foundation models, runtime was measured exclusively on the CPU, while GPUs were reserved for accuracy-only experiments.

**MTS Distance Baselines:** We compare multivariate SBDs against 6 strong baselines extended based on channel-dependency models [24], including ED [50], DTW [98], Longest Common Subsequence (LCSS) [107], Edit Distance with Real Penalty (ERP) [21], Time Warp Edit (TWE) [70], and Move-Split-Merge (MSM) [102]. Notably, both channel-dependency models for ED are equivalent [99].

**MTS Clustering Baselines:** We compare newly proposed MUFASA, with state-of-the-art algorithms across different categories: (a) *Scalable Methods:* multivariate $k$-Shape (kShape-D-I) – a simple extended version with SBD-D and a channel-independent centroid update – four algorithms from [74] (m-kMeans, m-kShape, m-KDBA, and m-KSC), Tslearn-kShape [106], Time2Feat [11], and MC2PCA [55]; (b) *Non-scalable Methods:* PAM + DTW-I and PAM + DTW-D [53, 100], as well as two recently proposed algorithms, including FCFW [56] and T-GMRF [25]; and (c) *Deep Learning-based Methods:* (i) DNN-based methods, which use traditional deep learning architectures to either learn representations or perform clustering directly: IDEC [34], SOM-VAE [27], USRL [28], DeTSEC [49], and SDCN [10]; and (ii) Foundation model-based methods, which leverage zero-shot learning capabilities from large-scale pre-trained models: OFA [114], MOMENT [33], and Chronos [6]. For the reasons discussed in Section 2.4, our evaluation of foundation models is exploratory, focusing on representation transferability; accordingly, we follow recent practice [79] and use foundation models for zero-shot representation learning. For representation learning-based models, we apply $k$-Means on the learned representations for clustering, a widely adopted strategy in our community.

**UTS Clustering Baselines:** We compare FASA with a similar set of univariate partitional clustering algorithms from [84], including $k$-Means [67], k-SC [112], k-DBA [96], $k$-Shape [84], and PAM + DTW [53]. Additionally, we include FrOKShape [57], a $k$-Shape variant that replaces SBD with a fractional-order, NCC–based distance and adopts DBA for centroid computation, KASBA [45], a $k$-Means–based clustering algorithm that integrates MSM as the distance measure and updates centroids via stochastic subgradient descent, and k-Graph [19], which constructs interpretable clusters through graph learning on variable-length subsequences.

**Implementation:** We utilized the official implementations provided by the authors for baseline comparisons. If the source code was unavailable, we reimplemented the methods in Python to ensure a fair comparison. To facilitate repeatability, we make all datasets and source code available [1].

**Parameterization:** We evaluate multivariate SBDs exclusively in an unsupervised setting to align with the nature of clustering. Following best practices, we utilize a fixed set of robust parameter

Table 1. Pairwise comparison of sliding measures against ED on the original and downsampled UEA archives (denoted in parentheses). ">", "=", and "<" represent the number of datasets where each method surpasses, matches, or underperforms the baseline, respectively. ✓, ✗, and ≈indicate significantly better, worse, or equivalent performance compared to the baseline, based on the pairwise Wilcoxon test ($\alpha = 0.05$). "Runtime" indicates the runtime ratio relative to ED.

| Measure | Average Accuracy | > | = | < | Diff. | Runtime |
|---|---|---|---|---|---|---|
| SBD-D | 0.6336 (0.6190) | 20 (19) | 3 (2) | 7 (9) | ✓(✓) | 11x |
| SBD-I | 0.6266 (0.6133) | 17 (16) | 2 (1) | 11 (13) | ≈(≈) | 11x |
| **Euclidean** | **0.6129 (0.6046)** | **0** | **30** | **0** | - | 1x |

values (Table 2) validated across hundreds of datasets in prior benchmarks [24, 89]. For partitional clustering, we set a maximum of 100 iterations using random initialization over the full time-series length. For all other baselines, we adopt the unsupervised parameters recommended in their original studies or prior studies [79] to ensure a fair and standardized comparison.

**Evaluation Metric:** Following standard practice in prior studies [7, 89, 99], we employ a 1-NN classifier to assess the discriminative power of each distance measure, as its performance (classification accuracy) depends solely on the measure's effectiveness. For clustering, we utilize widely used metrics including Rand Index (RI) [97], Adjusted Rand Index (ARI) [48], and Normalized Mutual Information (NMI) [103]. Given that these measures are highly correlated, we follow common practice [79, 84] and report RI results on fused training and test sets. RI quantifies the similarity between predicted and ground-truth assignments, ranging from 0 (random) to 1 (perfect agreement). To mitigate randomness, we report average results over 10 runs with different random seeds.

**Statistical Validation:** We evaluate statistical significance in classification and clustering performance using two standard protocols. For pairwise comparisons between competing methods, we utilize the Wilcoxon test [109] with $\alpha = 0.05$, following prior studies [23]. For multi-algorithm comparisons, we employ the Friedman test [29] followed by a post-hoc Nemenyi test [23, 73]. Given that the Friedman-Nemenyi test requires stronger evidence than the Wilcoxon test [31], we set $\alpha = 0.1$ to detect significance across all datasets and algorithms. Results are visualized via Critical Difference (CD) diagrams, where the axis denotes average ranks and horizontal solid lines connect methods with no statistically significant difference.

## 6 EXPERIMENTAL RESULTS

In this section, we first assess SBD-D and SBD-I in comparison to the strong MTS distance baselines (Section 6.1). Next, we examine design choices of MUFASA and compare MUFASA against scalable (Section 6.2), non-scalable (Section 6.3) and DNNs (including foundation models) (Section 6.4) approaches. While this paper primarily focuses on MTS clustering, we also evaluate FASA by comparing it with representative partitional clustering algorithms (Section 6.6).

### 6.1 Evaluation of Multivariate SBD

**Comparison against ED:** Table 1 provides a pairwise comparison of sliding distance measures against ED on both original and downsampled versions of the UEA archives. On the original archive, only SBD-D significantly outperforms ED (winning on 20/30 datasets). While SBD-I improves upon ED with 17 wins, it fails to reach statistical significance. Notably, the superior performance of SBD-D over SBD-I suggests that the increased temporal flexibility of the independent model may lead to overfitting rather than accuracy gains. Downsampling does not alter these conclusions; average accuracy shifts by less than 0.015, and win/tie/loss counts vary by at most two datasets across all measures. Given that SBD-D remains significantly superior to ED and achieves the strongest overall performance as a parameter-free method, we adopt it as our primary baseline and confirm the reliability of the downsampled UEA archive.

**Comparison against Elastic Measures:** We next compare elastic distance measures in an unsupervised setting, using SBD-D as the baseline, as it achieves the best performance in the preceding comparisons. For elastic measures, we adopt the parameter configurations used in prior benchmarks [24, 89]. These values were derived via Leave-One-Out Cross-Validation (LOOCV) to identify settings that perform well on average across datasets, as supported by accuracy and statistical test analyses. This choice avoids disadvantaging elastic measures due to a lack of tuning, while remaining consistent with an unsupervised evaluation protocol. When ED is used as the baseline, several elastic measures significantly outperform it, indicating that ED is a weak reference; in contrast, SBD-D, which significantly outperforms ED, provides a more appropriate baseline. All experiments are conducted on the downsampled version of the UEA archive (see Section 5 for details). Downsampling the original archive is necessary for two main reasons: (a) several elastic baselines, such as variants of MSM and TWE, are highly time-consuming and exceed the seven-day runtime limit on multiple datasets; and (b) several non-scalable clustering algorithms exhibit prohibitively high computational complexity (e.g., T-GMRF [25] has cubic complexity with respect to both $N$ and $D$ of MTS) and cannot complete on a substantial portion of the original datasets within the allotted time. Table 2 reports the performance of elastic measures, ordered by average rank. The results show that the average accuracy gap between MSM-I, the top-ranked elastic measure, and SBD-D is only 0.0082, while LCSS-I, which attains the highest average accuracy among elastic measures, improves over SBD-D by just 0.0149. Importantly, none of the elastic measures significantly outperform SBD-D, and SBD-D significantly outperforms LCSS-D. These findings underscore SBD-D as a strong, parameter-free baseline and suggest that elastic measures are not necessarily superior to sliding measures. Moreover, all elastic methods favor their channel-independent variants, in contrast to the behavior observed for SBD. We further assess statistical significance across multiple elastic measures alongside the baseline using the Friedman–Nemenyi test, which also aligns with our previous findings.

**Efficiency:** We have shown that SBD-D is competitive in accuracy; we now demonstrate that it is also highly efficient. As shown in Table 1, although ED is the fastest method, both SBD-I and SBD-D are about one order of magnitude slower, with SBD-D significantly outperforming ED in accuracy, unlike SBD-I. This establishes SBD-D as offering a stronger accuracy–runtime trade-off than both ED and SBD-I. Table 2 shows that elastic measures are 3 to 6 orders of magnitude slower than ED, whereas SBD-D is only 11x slower. Notably, while LCSS-I achieves the highest accuracy and MSM-I is the top-ranked elastic measure, they are approximately 3 to 4 orders of magnitude slower than SBD-D, respectively, without offering significantly superior performance. More broadly, elastic measures are on average 2 to 4 orders of magnitude slower than SBD-D, yet none significantly outperforms SBD-D in accuracy, with LCSS-D even performing significantly worse than SBD-D. We further note that SBD-D is parameter-free, whereas most elastic measures require careful parameter tuning. Consequently, SBD-D provides a more favorable trade-off between runtime and accuracy than elastic alternatives. Figure 4 reports scalability for pairwise distance computations between two MTS, comparing SBD-D with MSM-I, the top-ranked elastic measure. In Figure 4a, fixing $D = 50$ and varying length $L$, SBD-D is consistently faster; at $L = 8{,}192$, MSM-I requires about 31 hours—over 5 orders of magnitude slower—and is terminated for larger $L$ after exceeding the 24-hour runtime limit, while SBD-D scales to $L = 2^{17}$ in 8 seconds. Figure 4b fixes $L = 1{,}024$ and varies channels $D$: at $D = 1{,}500$, MSM-I takes roughly 15 hours, whereas SBD-D completes in 2 seconds, exceeding a four-order-of-magnitude speedup. Overall, SBD-D exhibits strong scalability in both time-series length and channel dimensionality. In conclusion, while no single measure dominates both runtime and accuracy, SBD-D provides the best trade-off, making it a strong practical default when both efficiency and quality are critical.

Table 2. Pairwise comparison of elastic measures and SBD-D. The rightmost column shows the runtime comparisons against ED. Refer to Table 1 for other column descriptions.

| Measure | Parameter Tuning | Average Accuracy | > | = | < | Diff. | Runtime |
|---|---|---|---|---|---|---|---|
| MSM-I | $c = 1.0$ | 0.6272 | 20 | 1 | 9 | ≈ | 408421x |
| DTW-I | $\delta = 100$ | 0.6265 | 13 | 4 | 13 | ≈ | 42057x |
| | $\delta = 10$ | 0.6309 | 18 | 3 | 9 | ≈ | 16431x |
| | $\delta = 5$ | 0.6303 | 19 | 2 | 9 | ≈ | 13576x |
| LCSS-I | $\delta = 10, \epsilon = 0.5$ | 0.6339 | 19 | 2 | 9 | ≈ | 16137x |
| DTW-D | $\delta = 100$ | 0.6133 | 17 | 2 | 11 | ≈ | 11897x |
| | $\delta = 10$ | 0.6072 | 16 | 2 | 12 | ≈ | 3603x |
| | $\delta = 5$ | 0.6089 | 13 | 2 | 15 | ≈ | 2618x |
| TWE-I | $\lambda = 0.5, \nu = 0.01$ | 0.6245 | 18 | 2 | 10 | ≈ | 183321x |
| MSM-D | $c = 0.5$ | 0.6049 | 15 | 2 | 13 | ≈ | 54712x |
| ERP-I | $\delta = 100$ | 0.6261 | 17 | 1 | 12 | ≈ | 48991x |
| TWE-D | $\lambda = 0.75, \nu = 0.00001$ | 0.6038 | 13 | 2 | 15 | ≈ | 27563x |
| ERP-D | $\delta = 100$ | 0.6115 | 16 | 2 | 12 | ≈ | 16777x |
| LCSS-D | $\delta = 50, \epsilon = 0.5$ | 0.5952 | 7 | 2 | 21 | ✗ | 9113x |
| **SBD-D** | - | **0.6190** | **0** | **30** | **0** | - | 11x |

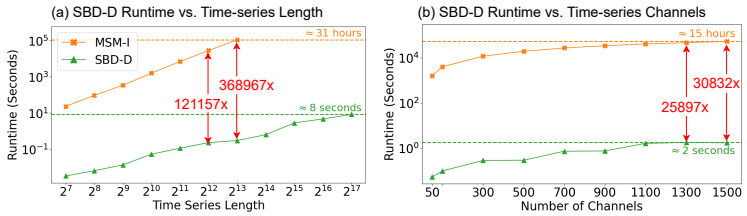



Fig. 4. Runtime comparison of SBD-D and MSM-I as: (a) length varies; (b) the number of channels varies.

## 6.2 MUFASA vs. Scalable Methods

**Examining Design Choices of MUFASA:** In Figure 5, we systematically evaluate four combinations arising from two channel-dependency extensions for SBD and two for FASA's centroid updates. The naming convention is as follows: (a) the first I/D after the method name denotes whether the distance is channel-independent (I) or channel-dependent (D); and (b) the second I/D specifies whether the centroid update is channel-independent or channel-dependent. For example, FASA-D-I indicates FASA with a SBD-D and a channel-independent centroid update. Figure 5a presents a pairwise comparison of multivariate FASA extensions against FASA-D-I—a natural baseline since SBD-D outperforms SBD-I—on the UEA archive. Although MUFASA shows no statistically significant difference from FASA-D-I, it achieves slightly higher average accuracy and outperforms the baseline on 57% of datasets. In contrast, FASA-I-D and FASA-I-I are significantly worse than FASA-D-I, reaffirming the superiority of SBD-D over SBD-I. The Friedman–Nemenyi results in Figure 5b are consistent with the Wilcoxon findings: FASA-D-I significantly outperforms both extensions that use SBD-I, while MUFASA—which combines SBD-D with a channel-dependent centroid update—achieves the highest overall performance by surpassing FASA-I-D significantly and FASA-D-I without statistical significance. Moreover, regardless of the SBD choice, the channel-dependent centroid update consistently outperforms its channel-independent counterpart, underscoring the effectiveness of channel-dependent design in both the distance measure and centroid update when extending from the univariate to multivariate setting. For FASA, both the distance measure and centroid update consistently favor the channel-dependent extension.

Having identified MUFASA's channel-dependent distance and centroid update as the best overall design, we next examine normalization choices within the centroid update—specifically, Z-score variants applied at lines 11 and 16 of Algorithm 2. Each step considers three options: channel-independent (I), channel-dependent (D), or no Z-score (R), yielding nine combinations evaluated via

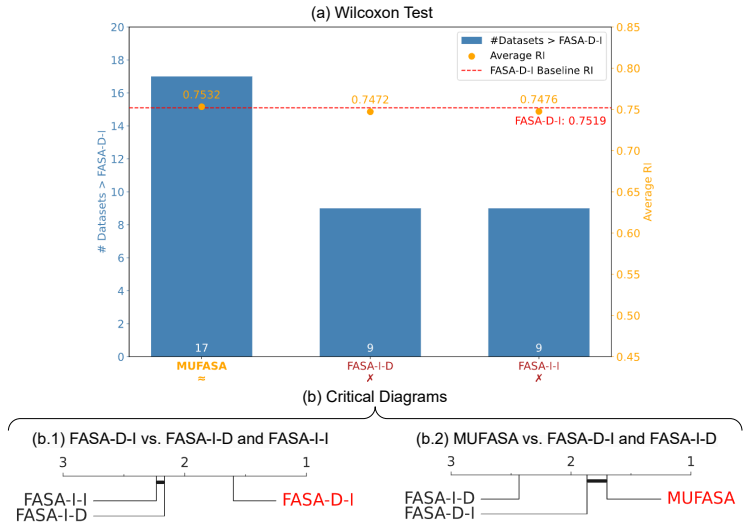



Fig. 5. (a) Pairwise comparison of multivariate FASA extensions with FASA-D-I serving as the baseline method. The bar plot (left axis) reports the number of datasets on which each algorithm outperforms the baseline, while the scatter plot (right axis) presents their average RI. The red dashed horizontal line marks the baseline's average RI. Symbols ✓, ✗, and ≈follow the notation in Table 1. (b) CD diagram with statistical test results comparing the multivariate FASA extensions.

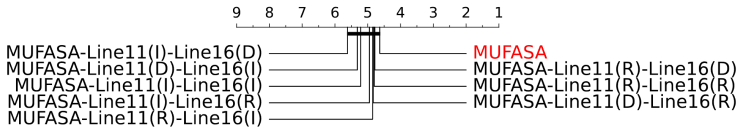



Fig. 6. CD diagram comparing normalization variants in MUFASA's centroid update (Algorithm 2, lines 11 and 16); the solid line indicates no significant difference.

a Friedman–Nemenyi test (Figure 6). The original MUFASA design applies channel-dependent Z-score at both steps and ranks first, but several alternatives perform comparably, with no statistically significant differences. This indicates that the centroid update is not highly sensitive to the specific normalization choices at these steps, and we therefore retain the original normalization design for all experiments to ensure a consistent and principled configuration.

**Comparing MUFASA with Scalable Methods:** Table 3 reports pairwise comparison results of scalable clustering algorithms that successfully complete on both the 30 original and 30 downsampled UEA datasets, using m-kMeans as the baseline. We include kShape-D-I, a multivariate extension of *k*-Shape that addresses known limitations of m-kShape and Tslearn-kShape. Algorithms are ordered by their average rank under the Friedman–Nemenyi test. On the original UEA datasets, MUFASA is the only scalable method that significantly outperforms m-kMeans, achieving the highest average RI and winning on 73% of the datasets. Appendix Section D shows the advantage of MUFASA over m-kMeans in settings with temporal variation, where alignment-aware clustering yields more meaningful results. kShape-D-I ranks second and is statistically indistinguishable from m-kMeans—despite outperforming it on 63% of datasets—while m-kShape also shows no significant difference. All remaining methods are significantly worse than m-kMeans. Results on the downsampled datasets exhibit only minor variations: the change in average RI for each method is below 0.015, and win/tie/loss counts differ by at most four datasets per cell. The statistical conclusions remain unchanged under both settings – MUFASA consistently achieves the highest average RI and is the only scalable algorithm that significantly outperforms m-kMeans. Appendix Figure IVa

Table 3. Pairwise comparison of scalable algorithms that successfully complete on both the 30 original and 30 downsampled UEA datasets, using m-kMeans as the baseline. See Table 1 for other column descriptions.

| Algorithm | Average RI | > | = | < | Diff. |
|---|---|---|---|---|---|
| MUFASA | 0.7532 (0.7527) | 22 (21) | 0 (0) | 8 (9) | ✓(✓) |
| kShape-D-I | 0.7441 (0.7449) | 19 (17) | 0 (0) | 11 (13) | ≈(≈) |
| m-kShape | 0.7268 (0.7339) | 15 (13) | 0 (0) | 15 (17) | ≈(≈) |
| Tslearn-kShape | 0.7060 (0.7061) | 10 (7) | 1 (0) | 19 (23) | ✗(✗) |
| m-KDBA | 0.7201 (0.7068) | 9 (6) | 0 (0) | 21 (24) | ✗(✗) |
| MC2PCA | 0.6990 (0.6871) | 9 (8) | 0 (0) | 21 (22) | ✗(✗) |
| m-KSC | 0.4813 (0.4778) | 4 (5) | 0 (0) | 26 (25) | ✗(✗) |
| **m-kMeans** | **0.7353 (0.7387)** | **0** | **30** | **0** | - |

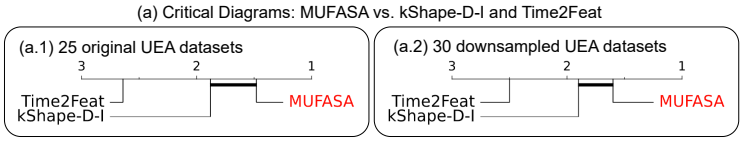



Fig. 7. CD diagrams comparing MUFASA with kShape-D-I and Time2Feat on (a.1) 25 original and (a.2) 30 downsampled UEA datasets; solid lines indicate no significant difference.

reports the overall RI of MUFASA and other scalable clustering methods on the 30 original UEA datasets, revealing a clear performance gap between MUFASA and the baselines. Methods are ranked by average RI, with MUFASA achieving the highest performance, followed by kShape-D-I, m-kMeans, m-kShape, and the remaining scalable approaches. Appendix Figure IVb presents a dual-axis comparison between MUFASA and multivariate *k*-Shape extensions. MUFASA attains Rank 1 on 14 of the 30 datasets—the highest frequency among all methods—and also achieves the highest average RI. kShape-D-I ranks second but reaches Rank 1 on only eight datasets, six fewer than MUFASA. In contrast, m-kShape and Tslearn-kShape achieve Rank 1 on half as many datasets as kShape-D-I and on 10 fewer datasets than MUFASA. Moreover, m-kShape's average RI is approximately 3% lower than MUFASA's, while Tslearn-kShape trails by about 5%.

The above comparisons exclude Time2Feat, as it completes on only 25 original UEA datasets, with memory issues encountered on the remaining five large datasets (DuckDuckGeese, PEMS-SF, EigenWorms, FaceDetection, and InsectWingbeat). We therefore present Figure 7, which reports Friedman–Nemenyi results comparing Time2Feat with the top two scalable clustering algorithms in Table 3, namely MUFASA and kShape-D-I. Figure 7a.1 shows results on the 25 original UEA datasets where Time2Feat completes, while Figure 7a.2 reports results on all 30 downsampled UEA datasets. Under both settings, MUFASA achieves the top rank, outperforming kShape-D-I without statistical significance; nevertheless, the performance gap between the two is large, and both significantly outperform Time2Feat. In summary, MUFASA consistently achieves the strongest accuracy and is therefore used as the baseline for subsequent comparisons. Moreover, the consistent conclusions across the original and downsampled UEA archives confirm that downsampling does not affect the validity of our clustering evaluation.

### 6.3 MUFASA vs. Non-Scalable Methods

Table 4 shows a pairwise comparison of MUFASA and representative non-scalable methods on the downsampled UEA datasets, considering the high computation cost (see Section 6.1). Algorithms are ordered by their average rank from the Friedman–Nemenyi test. PAM + DTW-I and PAM + DTW-D rank as the top two methods. However, MUFASA, being a scalable and accurate algorithm, shows no statistical difference from these methods and achieves a comparable average RI. In contrast, FCFW and T-GMRF perform significantly worse, with MUFASA outperforming them on 77% of the datasets, which is further supported by the Friedman–Nemenyi test (Figure 8a.1). Figure 8a.2 further

Table 4. Pairwise comparison of non-scalable algorithms and MUFASA, on the downsampled UEA archive.

| Algorithm | Average RI | > | = | < | Diff. |
|---|---|---|---|---|---|
| PAM + DTW-I | 0.7615 | 18 | 0 | 12 | ≈ |
| PAM + DTW-D | 0.7506 | 17 | 0 | 13 | ≈ |
| FCFW | 0.6639 | 7 | 0 | 23 | ✗ |
| T-GMRF | 0.6807 | 7 | 0 | 23 | ✗ |
| **MUFASA** | **0.7527** | **0** | **30** | **0** | - |

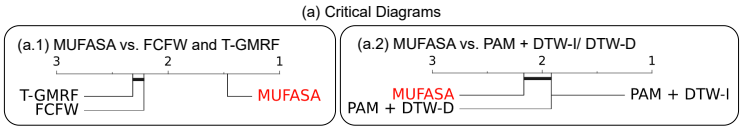



Fig. 8. CD diagrams of MUFASA versus non-scalable methods; solid lines indicate no significant difference.

Table 5. Pairwise comparison of deep learning-based algorithms and MUFASA, on the original UEA archive.

| Category | Algorithm | Average RI | > | = | < | Diff. |
|---|---|---|---|---|---|---|
| **Deep Learning** | SDCN | 0.7334 | 11 | 0 | 19 | ✗ |
| | DeTSEC | 0.7311 | 11 | 0 | 19 | ✗ |
| | USRL | 0.7201 | 9 | 0 | 21 | ✗ |
| | IDEC | 0.7185 | 11 | 0 | 19 | ✗ |
| | SOM-VAE | 0.6430 | 5 | 0 | 25 | ✗ |
| **Foundation Models** | Chronos | 0.7479 | 12 | 0 | 18 | ≈ |
| | OFA | 0.7319 | 10 | 0 | 20 | ✗ |
| | MOMENT | 0.7369 | 8 | 0 | 22 | ✗ |
| | **MUFASA** | **0.7532** | **0** | **30** | **0** | - |

indicates the comparable performance of MUFASA compared to the top non-scalable baselines. However, the lack of scalability of non-scalable methods renders them impractical for large-scale datasets. Owing to its efficiency and competitive accuracy, MUFASA, as a scalable method, stands out as the best choice among traditional clustering algorithms.

### 6.4 MUFASA vs. DNNs & Foundation Models

We evaluate MUFASA against representative DNNs and Foundation Models on the original UEA archive. Table 5 presents pairwise comparisons with baselines. Among DNN-based methods, SDCN achieves the highest performance, followed by DeTSEC, yet both record average RI scores about 2% lower than MUFASA, which outperforms them on 63% of datasets. USRL and IDEC show average RI values roughly 3% lower, and SOM-VAE trails by 11%; MUFASA surpasses USRL on 70% of datasets, IDEC on 63%, and SOM-VAE on 83%. All DNN baselines perform significantly worse than MUFASA according to the Wilcoxon test. Among Foundation Models, Chronos attains a slightly lower average RI than MUFASA, while OFA and MOMENT are each about 2% lower. MUFASA outperforms Chronos without statistical significance, despite outperforming it on 60% of the datasets. In contrast, MUFASA exhibits a statistically significant advantage over the other foundation models. Notably, as these foundation models are pre-trained on vast public repositories, they likely have prior exposure to the benchmark datasets. Despite this potential advantage and their dependence on substantial GPU resources, these models fail to surpass MUFASA, which achieves superior performance without any training or specialized hardware.

Appendix Figure Va illustrates the overall RI of MUFASA against DNNs and foundation models, where MUFASA achieves the top rank, followed by Chronos, MOMENT, and SDCN. Appendix Figure Vb provides a comparison in each category. MUFASA attains the highest frequency of Rank 1, approximately 1.4 times that of SDCN and 1.6 times that of Chronos. As a CPU-only, scalable traditional method based on efficient statistical techniques, MUFASA surpasses algorithms with complex structural designs such as DNNs and Foundation Models, making it the optimal choice.

### 6.5 Runtime Analysis of MUFASA

In this section, we demonstrate the computational efficiency of MUFASA. Figure 9 highlights MUFASA's strong efficiency–accuracy trade-off. RI values are averaged over all 30 UEA datasets [7], while runtime is reported as the sum across eight representative datasets—AtrialFibrillation, BasicMotions, ERing, Epilepsy, JapaneseVowels, Libras, NATOPS, and RacketSports—with each dataset averaged over five independent experimental runs. This evaluation protocol is adopted due to the high computation cost of non-scalable methods —several cannot finish large datasets within seven days—making exhaustive evaluation infeasible. Selecting representative benchmarks, therefore, enables fair inclusion of slower baselines. These datasets were selected by combining runtime-complexity analysis with empirical timing on small datasets to estimate runtime on larger datasets, followed by validation on subsets of the UEA datasets. The final set ensures that all algorithms complete within the seven-day limit, covers diverse MTS shapes, and avoids cases where a single dataset with extremely high runtime dominates the total runtime (e.g., FingerMovements, where T-GMRF requires over 100 hours; despite completing within the seven-day limit, the dataset is excluded to prevent skewing the aggregated runtime).

Across all four comparison groups—scalable (Figure 9a), non-scalable baselines together with Time2Feat, which cannot finish on all 30 original UEA datasets (Figure 9b), DNNs (Figure 9c), and foundation models (Figure 9d)—MUFASA lies on the upper-left frontier of the RI–runtime plots, indicating a superior accuracy–efficiency trade-off relative to competing methods. Notably, average RI can be misleading, as it masks substantial per-dataset accuracy variation; thus, the per-dataset statistical analyses are essential for proper interpretation. Retaining all diverse UEA datasets for fairness naturally compresses average accuracy differences across methods. Among scalable methods, MUFASA is the only approach that significantly outperforms m-kMeans, achieving the best accuracy–efficiency trade-off. Although m-kMeans is slightly faster, Table 3 shows that MUFASA achieves a higher average RI and outperforms it on 73% of datasets, indicating a meaningful accuracy advantage. In contrast, kShape-D-I exhibits slightly higher runtime with lower average RI, MC2PCA trades speed for substantial accuracy loss, and m-KDBA, m-KSC, and Time2Feat are uncompetitive in both accuracy and runtime. Non-scalable methods incur severe runtime cost (about 652x–1,959x) while generally failing to surpass MUFASA in average RI, except for PAM + DTW-I; consistent with Table 4, MUFASA shows no significant difference from the top two non-scalable methods and significantly outperforms the rest. DNNs and foundation models (marked with $*$ to denote CPU-estimated runtimes) are 3x–821x slower without accuracy gains; MUFASA outperforms Chronos on 60% of datasets and significantly surpasses all remaining deep and foundation baselines (Table 5), confirming MUFASA as the preferred accuracy–efficiency trade-off.

Figure 10a compares MUFASA with the strongest scalable baseline, kShape-D-I, on synthetic datasets while varying the number of MTS $N$, fixing $L = 1{,}024$ and $D = 50$. Consistent with the theoretical linear dependence on $N$, both methods exhibit approximately linear runtime scaling, with MUFASA remaining consistently faster and achieving about 2x speedup at $N = 3{,}000$ and $N = 10{,}000$. Figure 11a evaluates scalability with respect to sequence length $L$, fixing $N = 100$ and $D = 50$, where MUFASA maintains lower runtime and a slower growth rate as $L$ increases; the gap widens substantially, with kShape-D-I becoming about 7x slower at $L = 1{,}024$, nearly 18x

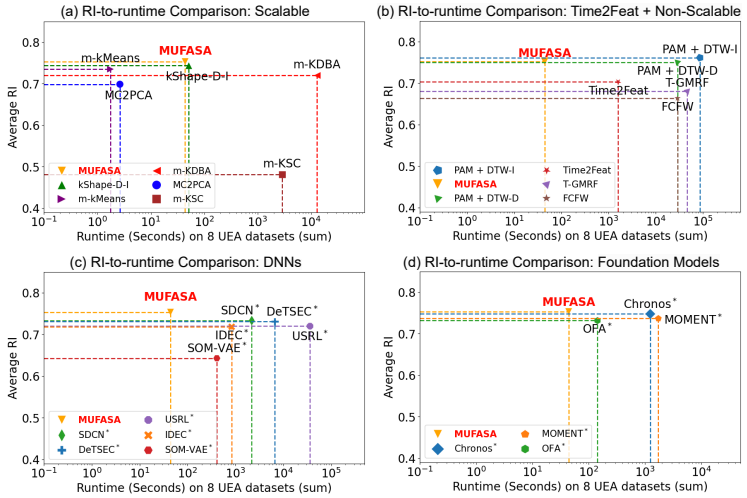



Fig. 9. RI-to-runtime comparisons of MUFASA against (a) scalable algorithms, (b) Time2Feat and non-scalable algorithms, (c) DNNs, and (d) Foundation Models.

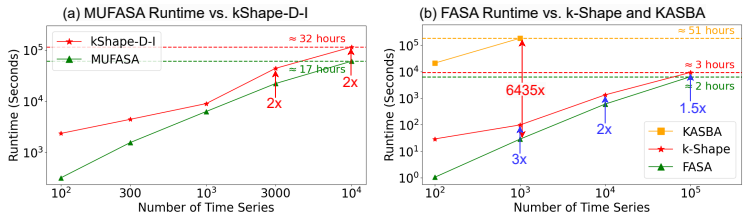



Fig. 10. Runtime comparisons: (a) MUFASA vs. kShape-D-I as the number of MTS varies; and (b) FASA vs. k-Shape and KASBA as the number of UTS varies.

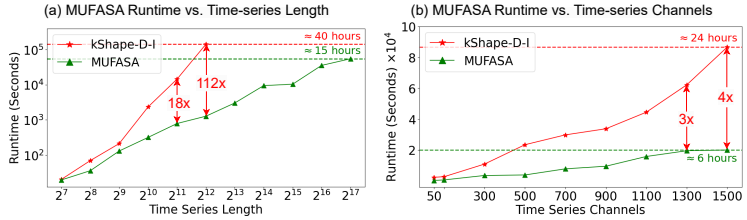



Fig. 11. Runtime comparisons: (a) MUFASA vs. kShape-D-I as time-series length varies; and (b) MUFASA vs. kShape-D-I as the number of time-series channels varies.

slower at $L = 2{,}048$, and reaching approximately 40 hours at $L = 4{,}096$ (about 112x slower), after which it exceeds the 24-hour limit, while MUFASA continues to scale and completes in roughly 15 hours at $L = 2^{17}$. Figure 11b further examines scalability with respect to channel $D$, fixing $N = 100$ and $L = 1{,}024$, where MUFASA scales smoothly and remains consistently faster, whereas kShape-D-I grows steeply, becoming 3x slower at $D = 1{,}300$ and requiring nearly 24 hours at $D = 1{,}500$, while MUFASA remains roughly 4x faster. For context, the median dimensions of the original UEA datasets ($N = 413$, $L = 167$, and $D = 6$) are substantially smaller than those used here, explaining the fast runtime observed in typical settings. The synthetic scalability experiments demonstrate that MUFASA offers a favorable scalability profile with respect to $N$, $L$, and $D$ relative to the strongest scalable baseline.

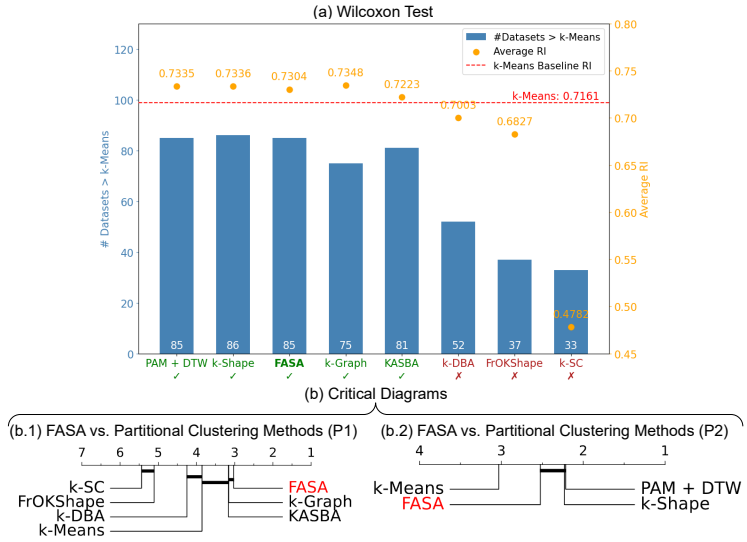



Fig. 12. (a) Pairwise comparison of FASA with SOTA partitional clustering methods using *k*-Means as the baseline. Description refers to Figure 5. (b) CD diagram with statistical test results comparing FASA to SOTA partitional clustering methods, where solid lines indicate no significant difference.

### 6.6 Evaluation of FASA

**Comparing FASA with Partitional Clustering Algorithms:** Although this paper primarily focuses on MTS clustering, we also evaluate FASA as complementary evidence by comparing it with state-of-the-art partitional clustering algorithms on the UCR dataset. Figure 12a uses *k*-Means as the reference, and methods are ranked by average rank from the Friedman–Nemenyi test. Four algorithms outperform *k*-Means on at least 63% of the datasets: *k*-Shape, PAM + DTW, FASA, and KASBA. FASA ranks third by average rank and significantly outperforms *k*-Means, yet its average RI is within 0.5% of the top two methods, and the number of datasets where FASA beats *k*-Means differs by at most one. k-Graph, a non-scalable algorithm, ranks fourth, significantly outperforming *k*-Means and achieving a slightly higher average RI than FASA, but it surpasses *k*-Means on 10 fewer datasets than FASA. KASBA also significantly outperforms *k*-Means but attains a lower average RI than FASA and beats *k*-Means on four fewer datasets. In contrast, k-DBA, FrOKShape, and k-SC are all significantly outperformed by *k*-Means, with k-SC exhibiting low average RI; FrOKShape and k-SC outperform *k*-Means on fewer than 40 datasets, less than half of FASA.

The Friedman–Nemenyi test in Figure 12b.1 shows that FASA significantly outperforms *k*-Means, k-DBA, FrOKShape, and k-SC, and also surpasses k-Graph and KASBA without statistical significance. Figure 12b.2 ranks PAM + DTW first and *k*-Shape second, with FASA showing no significant difference from these top two methods. All three methods outperform *k*-Means with statistical significance. Notably, PAM + DTW is a non-scalable UTS clustering algorithm, whereas *k*-Shape and FASA are scalable and require substantially less runtime, making them more practical for large-scale applications. Moreover, FASA provides an optimal balance between accuracy and efficiency by reducing the cubic complexity of *k*-Shape's centroid update with respect to UTS length to linear complexity, making it a highly efficient choice.

**Efficiency:** We evaluate the efficiency of FASA by comparing it with the strongest scalable partitional baselines, *k*-Shape and KASBA, on synthetic datasets. To ensure a fair comparison, we disable any sampling strategy and require all methods to process the full dataset. Although FrOKShape is a *k*-Shape variant, it is both substantially less accurate and significantly slower than FASA, exceeding FASA's runtime by over 400x at $L = 256$ and $L = 512$, and by 2,453x at $L = 1{,}024$; consequently, it is

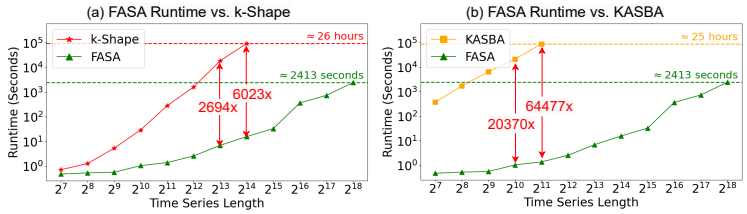



Fig. 13. (a) and (b) Runtime comparisons between FASA and top-performing scalable partitional clustering algorithms on synthetic datasets, varying $L$ with $N = 100$.

excluded from full scalability experiments. Fixing $L = 1{,}024$ and varying $N$, Figure 10b shows that KASBA is the slowest method, reaching approximately 51 hours at $N = 1{,}000$ (6,435x slower than FASA) and exceeding the 24-hour limit, while $k$-Shape remains consistently slower than FASA, ranging from 3x at $N = 1{,}000$ to 1.5x at $N = 100{,}000$; both exhibit near-linear scaling in practice, consistent with their theoretical linear dependence on $N$. Fixing $N = 100$ and varying $L$, Figure 13a shows that although $k$-Shape and FASA incur comparable costs on short sequences, $k$-Shape's runtime grows sharply with length, becoming 2,694x slower at $L = 8{,}192$ and 6,023x slower at $L = 2^{14}$, after which it exceeds a 24-hour limit, while FASA continues to scale efficiently, completing in under an hour even at $L = 2^{18}$. Figure 13b further shows that KASBA scales poorly with length, requiring 20,370x and 64,477x the runtime of FASA at $L = 1{,}024$ and $L = 2{,}048$, respectively. Given that the median UCR dataset dimensions ($N = 636$, $L = 344$) are substantially smaller than those used in the scalability experiments, fast execution is expected in typical real-world settings. The scalability experiments further show that FASA scales favorably with respect to both $N$ and $L$.

## 7 Conclusion

In this paper, we present MUFASA, an accurate and scalable algorithm for MTS clustering. As its foundation, we first introduce FASA, a scalable univariate clustering method that combines SBD with our novel centroid update procedure scaling linearly with both the number and length of time series. Building on FASA, MUFASA introduces SBD-D alongside a novel multivariate centroid optimization that unifies the multivariate sequence into a structurally integrated entity to jointly estimate all channels, effectively capturing temporal dynamics and inter-channel dependencies. Extensive experiments demonstrate that SBD-D achieves accuracy comparable to elastic measures at a fraction of their runtime and significantly outperforms ED. Across comprehensive evaluations spanning multiple categories of clustering algorithms, MUFASA consistently delivers superior accuracy and efficiency, while complementary experiments confirm FASA's effectiveness and scalability. Together, MUFASA and FASA provide accurate, scalable, and parameter-light solutions for both MTS and UTS clustering, making them well-suited for real-world applications.

## References

[1] 2026. MUFASA: Fast and Accurate Multivariate Time-Series Clustering. https://github.com/thedatumorg/MUFASA. Accessed: 2026-03-15.

[2] 2026. MUFASA: Fast and Accurate Multivariate Time-Series Clustering – Appendix. https://github.com/thedatumorg/MUFASA/blob/main/MUFASA_appendix.pdf. Accessed: 2026-03-15.

[3] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information systems* 53 (2015), 16–38.

[4] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning* 75, 2 (2009), 245–248.

[5] Ali Alqahtani, Mohammed Ali, Xianghua Xie, and Mark W Jones. 2021. Deep time-series clustering: A review. *Electronics* 10, 23 (2021), 3001.

[6] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. 2024. Chronos: Learning the language

of time series. *arXiv preprint arXiv:2403.07815* (2024).

[7] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075* (2018).

[8] Ziv Bar-Joseph, Georg Gerber, David K Gifford, Tommi S Jaakkola, and Itamar Simon. 2002. A new approach to analyzing gene expression time series data. In *Proceedings of the sixth annual international conference on Computational biology*. 39–48.

[9] Mohini Bariya, Alexandra von Meier, John Paparrizos, and Michael J Franklin. 2021. k-shapestream: Probabilistic streaming clustering for electric grid events. In *2021 IEEE Madrid PowerTech*. IEEE, 1–6.

[10] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Proceedings of the web conference 2020*. 1400–1410.

[11] Angela Bonifati, Francesco Del Buono, Francesco Guerra, and Donato Tiano. 2022. Time2Feat: learning interpretable representations for multivariate time series clustering. *Proceedings of the VLDB Endowment (PVLDB)* 16, 2 (2022), 193–201.

[12] Paul Boniol, Ashwin K Krishna, Marine Bruel, Qinghua Liu, Mingyi Huang, Themis Palpanas, Ruey S Tsay, Aaron Elmore, Michael J Franklin, and John Paparrizos. 2025. VUS: effective and efficient accuracy measures for time-series anomaly detection. *The VLDB Journal* 34, 3 (2025), 32.

[13] Paul Boniol, Qinghua Liu, Mingyi Huang, Themis Palpanas, and John Paparrizos. 2024. Dive into Time-Series Anomaly Detection: A Decade Review. *arXiv preprint arXiv:2412.20512* (2024).

[14] Paul Boniol, John Paparrizos, Yuhao Kang, Themis Palpanas, Ruey S Tsay, Aaron J Elmore, and Michael J Franklin. 2022. Theseus: Navigating the Labyrinth of Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 12 (2022), 3702–3705.

[15] Paul Boniol, John Paparrizos, and Themis Palpanas. 2023. New Trends in Time Series Anomaly Detection.. In *EDBT*. 847–850.

[16] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND in Action: Subsequence Anomaly Detection for Streams. *Proc. VLDB Endow.* 14, 12 (2021), 2867–2870.

[17] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *Proc. VLDB Endow.* 14, 10 (2021), 1717–1729.

[18] Paul Boniol, Emmanouil Sylligardos, John Paparrizos, Panos Trahanias, and Themis Palpanas. 2024. Adecimo: Model selection for time series anomaly detection. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 5441–5444.

[19] Paul Boniol, Donato Tiano, Angela Bonifati, and Themis Palpanas. 2025. *k*-Graph: a graph embedding for interpretable time series clustering. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[20] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[21] Lei Chen and Raymond Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 792–803.

[22] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica* 6, 6 (2019), 1293–1305.

[23] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research* 7 (2006), 1–30.

[24] Jens E d'Hondt, Haojun Li, Fan Yang, Odysseas Papapetrou, and John Paparrizos. 2025. A Structured Study of Multivariate Time-Series Distance Measures. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–29.

[25] Wangxiang Ding, Wenzhong Li, Zhijie Zhang, Chen Wan, Jianhui Duan, and Sanglu Lu. 2023. Time-Varying Gaussian Markov Random Fields Learning for Multivariate Time Series Clustering. *IEEE Transactions on Knowledge and Data Engineering* 35, 11 (2023), 11950–11966. doi:10.1109/TKDE.2022.3232331

[26] Jens E d'Hondt, Odysseas Papapetrou, and John Paparrizos. 2024. Beyond the Dimensions: A Structured Evaluation of Multivariate Time Series Distance Measures. In *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 107–112.

[27] Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. 2018. Som-vae: Interpretable discrete representation learning on time series. *arXiv preprint arXiv:1806.02199* (2018).

[28] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems* 32 (2019).

[29] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.

[30] Apostolos Giannoulidis, Anastasios Gounaris, and John Paparrizos. 2025. Burst: Rendering clustering techniques suitable for evolving streams. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4054–4063.
[31] Rafael Giusti and Gustavo EAPA Batista. 2013. An empirical comparison of dissimilarity measures for time series classification. In *2013 Brazilian Conference on Intelligent Systems*. IEEE, 82–88.
[32] Rahul Goel, Sandeep Soni, Naman Goyal, John Paparrizos, Hanna Wallach, Fernando Diaz, and Jacob Eisenstein. 2016. The social dynamics of language change in online networks. In *International conference on social informatics*. Springer, 41–57.
[33] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885* (2024).
[34] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation.. In *Ijcai*, Vol. 17. 1753–1759.
[35] Suchit Gupte and John Paparrizos. 2025. ShapX Engine: A Demonstration of Shapley Value Approximations. In *Companion of the 2025 International Conference on Management of Data*. 107–110.
[36] Suchit Gupte and John Paparrizos. 2025. Understanding the Black Box: A Deep Empirical Dive into Shapley Value Approximations for Tabular Data. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–31.
[37] Michael Hahsler and Matthew Bolaños. 2016. Clustering data streams based on shared density between micro-clusters. *IEEE transactions on knowledge and data engineering* 28, 6 (2016), 1449–1461.
[38] James D Hamilton. 2020. *Time series analysis*. Princeton university press.
[39] Jiawei Han, Jian Pei, and Hanghang Tong. 2022. *Data mining: concepts and techniques*. Morgan kaufmann.
[40] Pierre Hansen and Brigitte Jaumard. 1997. Cluster analysis and mathematical programming. *Mathematical programming* 79, 1 (1997), 191–215.
[41] Fred Harris. 2021. Polyphase Interpolators with Reversed Order of Up-Sampling and Down-Sampling. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 918–924.
[42] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.
[43] Douglas M Hawkins. 1980. *Identification of outliers*. Vol. 11. Springer.
[44] Kaisei Hishida, Chunwei Liu, John Paparrizos, and Aaron Elmore. 2025. Beyond Compression: A Comprehensive Evaluation of Lossless Floating-Point Compression. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4396–4409.
[45] Christopher Holder and Anthony Bagnall. 2024. Rock the KASBA: Blazingly Fast and Accurate Time Series Clustering. *arXiv preprint arXiv:2411.17838* (2024).
[46] Mingyi Huang, Qinghua Liu, Paul Boniol, and John Paparrizos. 2026. GlassboxAD: An Interactive System for Dissecting Hierarchical Time-Series Anomaly Detection. In *Companion of the International Conference on Management of Data (SIGMOD Companion '26)*. doi:10.1145/3788853.3801594
[47] Mingyi Huang, Qinghua Liu, Paul Boniol, and John Paparrizos. 2026. HYDRA: A Multi-Level Hierarchy-Driven Approach for Robust Anomaly Detection in Time Series. *Proceedings of the ACM on Management of Data* 4, 2, Article 197 (June 2026), 28 pages. doi:10.1145/3802074
[48] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2 (1985), 193–218.
[49] Dino Ienco and Roberto Interdonato. 2020. Deep multivariate time series embedding clustering via attentive-gated autoencoder. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*. Springer, 318–329.
[50] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data clustering: a review. *ACM computing surveys (CSUR)* 31, 3 (1999), 264–323.
[51] Hao Jiang, Chunwei Liu, Qi Jin, John Paparrizos, and Aaron J Elmore. 2020. PIDS: attribute decomposition for improved compression and query performance in columnar storage. *Proceedings of the VLDB Endowment* 13, 6 (2020), 925–938.
[52] Hao Jiang, Chunwei Liu, John Paparrizos, Andrew A Chien, Jihong Ma, and Aaron J Elmore. 2021. Good to the Last Bit: Data-Driven Encoding with CodecDB. In *Proceedings of the 2021 International Conference on Management of Data*. 843–856.
[53] Leonard Kaufman and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.
[54] Sanjay Krishnan, Aaron J Elmore, Michael Franklin, John Paparrizos, Zechao Shang, Adam Dziedzic, and Rui Liu. 2019. Artificial intelligence in resource-constrained and shared environments. *ACM SIGOPS Operating Systems Review* 53, 1 (2019), 1–6.
[55] Hailin Li. 2019. Multivariate time series clustering based on common principal component analysis. *Neurocomputing* 349 (2019), 239–247.
[56] Hailin Li and Miao Wei. 2020. Fuzzy clustering based on feature weights for multivariate time series. *Knowledge-Based Systems* 197 (2020), 105907.

[57] Yucheng Li, Derong Shen, Tiezheng Nie, and Yue Kou. 2022. A new shape-based clustering algorithm for time series. *Information Sciences* 609 (2022), 411–428.

[58] T Warren Liao. 2005. Clustering of time series data—a survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[59] Chunwei Liu, Hao Jiang, John Paparrizos, and Aaron J Elmore. 2021. Decomposed bounded floats for fast compression and queries. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2586–2598.

[60] Chunwei Liu, John Paparrizos, and Aaron J Elmore. 2024. AdaEdge: A Dynamic Compression Selection Framework for Resource Constrained Devices. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1506–1519.

[61] Qinghua Liu, Paul Boniol, Themis Palpanas, and John Paparrizos. 2024. Time-Series Anomaly Detection: Overview and New Trends. *Proceedings of the VLDB Endowment (PVLDB)* 17, 12 (2024), 4229–4232.

[62] Qinghua Liu, Sam Heshmati, Zheda Mai, Zubin Abraham, John Paparrizos, and Liu Ren. 2025. MLLM4TS: Leveraging Vision and Multimodal Language Models for General Time-Series Analysis. *arXiv preprint arXiv:2510.07513* (2025).

[63] Qinghua Liu, Seunghak Lee, and John Paparrizos. 2025. EasyAD: A Demonstration of Automated Solutions for Time-Series Anomaly Detection. *Proceedings of the VLDB Endowment* 18, 12 (2025), 5431–5434.

[64] Qinghua Liu, Seunghak Lee, and John Paparrizos. 2025. Tsb-autoad: Towards automated solutions for time-series anomaly detection. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4364–4379.

[65] Qinghua Liu and John Paparrizos. 2024. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. *Advances in Neural Information Processing Systems* 37 (2024), 108231–108261.

[66] Shinan Liu, Tarun Mangla, Ted Shaowang, Jinjin Zhao, John Paparrizos, Sanjay Krishnan, and Nick Feamster. 2023. AMIR: Active Multimodal Interaction Recognition from Video and Network Traffic in Connected Environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–26.

[67] J MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*.

[68] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. 2012. The planar k-means problem is NP-hard. *Theoretical computer science* 442 (2012), 13–21.

[69] Michael E Mann, Raymond S Bradley, and Malcolm K Hughes. 1998. Global-scale temperature patterns and climate forcing over the past six centuries. *Nature* 392, 6678 (1998), 779–787.

[70] Pierre-François Marteau. 2008. Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence* 31, 2 (2008), 306–318.

[71] Kathy McKeown, Hal Daume III, Snigdha Chaturvedi, John Paparrizos, Kapil Thadani, Pablo Barrio, Or Biran, Suvarna Bothe, Michael Collins, Kenneth R Fleischmann, et al. 2016. Predicting the impact of scientific concepts using full-text features. *Journal of the Association for Information Science and Technology* 67, 11 (2016), 2684–2696.

[72] Charles R Nelson and Charles R Plosser. 1982. Trends and random walks in macroeconmic time series: some evidence and implications. *Journal of monetary economics* 10, 2 (1982), 139–162.

[73] Peter Bjorn Nemenyi. 1963. *Distribution-free multiple comparisons*. Princeton University.

[74] Mert Ozer, Anna Sapienza, Andrés Abeliuk, Goran Muric, and Emilio Ferrara. 2020. Discovering patterns of online popularity from time series. *Expert Systems with Applications* 151 (2020), 113337.

[75] Anastasios Papadopoulos, Apostolos Giannoulidis, Anastasios Gounaris, and John Paparrizos. 2026. The Power of Anomaly Detection in Predictive Maintenance. *Proceedings of the ACM on Management of Data* 4, 2, Article 242 (June 2026), 34 pages. doi:10.1145/3802119

[76] Ioannis Paparrizos. 2018. *Fast, scalable, and accurate algorithms for time-series analysis*. Ph. D. Dissertation. Columbia University.

[77] Ioannis Paparrizos, Hoyoung Jeung, and Karl Aberer. 2011. Advanced search, visualization and tagging of sensor metadata. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 1356–1359.

[78] John Paparrizos. 2018. ucr time-series archive: Backward compatibility, missing values, and varying lengths. *URL: https://github. com/johnpaparrizos/UCRArchiveFixes* (2018).

[79] John Paparrizos and Sai Prasanna Teja Reddy Bogireddy. 2025. Time-series clustering: A comprehensive study of data mining, machine learning, and deep learning methods. *Proceedings of the VLDB Endowment* 18, 11 (2025), 4380–4395.

[80] John Paparrizos, Paul Boniol, Qinghua Liu, and Themis Palpanas. 2025. Advances in time-series anomaly detection: Algorithms, benchmarks, and evaluation measures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 6151–6161.

[81] John Paparrizos, Paul Boniol, Qinghua Liu, and Themis Palpanas. 2026. A Comprehensive Guide to Time-Series Anomaly Detection. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*. 1363–1366.

[82] John Paparrizos, Ikraduya Edian, Chunwei Liu, Aaron J Elmore, and Michael J Franklin. 2022. Fast Adaptive Similarity Search through Variance-Aware Quantization. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2969–2983.

[83] John Paparrizos and Michael J Franklin. 2019. GRAIL: efficient time-series representation learning. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1762–1777.
[84] John Paparrizos and Luis Gravano. 2015. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1855–1870.
[85] John Paparrizos and Luis Gravano. 2017. Fast and Accurate Time-Series Clustering. *ACM Transactions on Database Systems (TODS)* 42, 2 (2017), 1–49.
[86] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin. 2022. TSB-UAD: An End-to-End Benchmark Suite for Univariate Time-Series Anomaly Detection. *Proc. VLDB Endow.* 15, 8 (2022), 1697–1711.
[87] John Paparrizos, Haojun Li, Fan Yang, Kaize Wu, Jens E d'Hondt, and Odysseas Papapetrou. 2024. A Survey on Time-Series Distance Measures. *arXiv preprint arXiv:2412.20574* (2024).
[88] John Paparrizos, Chunwei Liu, Bruno Barbarioli, Johnny Hwang, Ikraduya Edian, Aaron J Elmore, Michael J Franklin, and Sanjay Krishnan. 2021. VergeDB: A Database for IoT Analytics on Edge Devices.. In *CIDR*.
[89] John Paparrizos, Chunwei Liu, Aaron J Elmore, and Michael J Franklin. 2020. Debunking four long-standing misconceptions of time-series distance measures. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. 1887–1905.
[90] John Paparrizos, Chunwei Liu, Aaron J Elmore, and Michael J Franklin. 2023. Querying Time-Series Data: A Comprehensive Comparison of Distance Measures. *Data Engineering* (2023), 69.
[91] John Paparrizos and Sai Prasanna Teja Reddy. 2023. Odyssey: An Engine Enabling the Time-Series Clustering Journey. *Proceedings of the VLDB Endowment* 16, 12 (2023), 4066–4069.
[92] John Paparrizos, Ryen W White, and Eric Horvitz. 2016. Detecting devastating diseases in search logs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 559–568.
[93] John Paparrizos, Ryen W White, and Eric Horvitz. 2016. Screening for pancreatic adenocarcinoma using signals from web search logs: Feasibility study and results. *Journal of Oncology Practice* 12, 8 (2016), 737–744.
[94] John Paparrizos, Kaize Wu, Aaron Elmore, Christos Faloutsos, and Michael J Franklin. 2023. Accelerating Similarity Search for Elastic Measures: A Study and New Generalization of Lower Bounding Distances. *Proceedings of the VLDB Endowment* 16, 8 (2023), 2019–2032.
[95] John Paparrizos, Fan Yang, and Haojun Li. 2024. Bridging the gap: A decade review of time-series clustering methods. *arXiv preprint arXiv:2412.20582* (2024).
[96] François Petitjean, Alain Ketterlin, and Pierre Gançarski. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition* 44, 3 (2011), 678–693.
[97] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.
[98] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.
[99] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I Webb. 2023. Elastic similarity and distance measures for multivariate time series. *Knowledge and Information Systems* 65, 6 (2023), 2665–2698.
[100] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. 2017. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data mining and knowledge discovery* 31 (2017), 1–31.
[101] Hongchao Song, Zhuqing Jiang, Aidong Men, and Bo Yang. 2017. A Hybrid Semi-Supervised Anomaly Detection Model for High-Dimensional Data. *Computational intelligence and neuroscience* 2017, 1 (2017), 8501683.
[102] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. 2012. The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering* 25, 6 (2012), 1425–1438.
[103] Colin Studholme, Derek LG Hill, and David J Hawkes. 1999. An overlap invariant entropy measure of 3D medical image alignment. *Pattern recognition* 32, 1 (1999), 71–86.
[104] Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos Trahanias, and Themis Palpanas. 2023. Choose wisely: An extensive evaluation of model selection for anomaly detection in time series. *Proceedings of the VLDB Endowment* 16, 11 (2023), 3418–3432.
[105] Emmanouil Sylligardos, John Paparrizos, Themis Palpanas, Pierre Senellart, and Paul Boniol. 2025. MSAD: A deep dive into model selection for time series anomaly detection. *The VLDB Journal* 34, 6 (2025), 1–25.
[106] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. 2020. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research* 21, 118 (2020), 1–6. http://jmlr.org/papers/v21/20-091.html
[107] Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn Keogh. 2003. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 216–225.

[108] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. 2019. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics* 8, 8 (2019), 876.

[109] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 196–202.

[110] Fan Yang and John Paparrizos. 2025. SAIL: A Voyage to Symbolic Approximation Solutions for Time-Series Analysis. *Proceedings of the VLDB Endowment* 18, 12 (2025), 5419–5422.

[111] Fan Yang and John Paparrizos. 2025. Spartan: Data-adaptive symbolic time-series approximation. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–30.

[112] Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 177–186.

[113] Nan Zhang and Shiliang Sun. 2023. Multiview Unsupervised Shapelet Learning for Multivariate Time Series Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4981–4996. doi:10.1109/TPAMI.2022.3198411

[114] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems* 36 (2023), 43322–43355.

[115] Robert J Zupko, Tran Dang Nguyen, J Claude S Ngabonziza, Michee Kabera, Haojun Li, Thu Nguyen-Anh Tran, Kien Trung Tran, Aline Uwimana, and Maciej F Boni. 2023. Modeling policy interventions for slowing the spread of artemisinin-resistant pfkelch R561H mutations in Rwanda. *Nature Medicine* 29, 11 (2023), 2775–2784.

Received October 2025; revised January 2026; accepted February 2026