MULISSE: Variable-Length Similarity Search for Multivariate Time Series

Balázs Pelok
Eindhoven University of Technology
Eindhoven, the Netherlands
b.pelok@student.tue.nl

Jens E. d'Hondt
Eindhoven University of Technology
Eindhoven, the Netherlands
j.e.d.hondt@tue.nl

***Abstract*—Time series data with multiple channels, known as multivariate time series (MTS), are increasingly common in modern applications. A common task in this context is subsequence search, which involves finding subsequences within large MTS datasets that closely match a given query pattern. While researchers have extensively studied this problem for univariate time series, applying these methods to multiple channels presents key challenges, including higher dimensionality and the need for flexible query specifications. Existing MTS search approaches are limited - they typically only match complete time series rather than subsequences, or restrict searches to predetermined channel combinations. This paper introduces MULISSE, a novel algorithm for exact variable-length k-NN subsequence search on multivariate time series. Extending the state-of-the-art algorithm for univariate time series (ULISSE), our approach incorporates multivariate symbolic approximations and refined node splitting strategies to efficiently handle multiple channels. Our experimental evaluation demonstrates the complexity of the problem: while achieving up to two orders of magnitude speedup on synthetic datasets with many channels, the gains over baseline methods on real-world datasets are more modest, ranging from 10% to 2x. The method's effectiveness is heavily influenced by factors such as time series length, number of channels, and the index's hyperparameters, highlighting both the potential and current limitations of index-based approaches for MTS search. As a first step towards efficient variable-length MTS search, this work not only demonstrates the viability of index-based methods but also identifies several promising directions for future improvements in areas such as dynamic SAX representations and cost-effective exhaustive distance computation.**

I. Introduction

Recent technological advances have led to an explosion in time series data collection across scientific and industrial domains [3], [21], [24], [27], [36]. Particularly prevalent are multivariate time series (MTS) [39], where multiple synchronized measurements capture different aspects of a phenomenon or object: climate stations record temperature, humidity, and air pressure at specific locations [29]; motion capture systems track positions and accelerations of different body parts over time [4]; and hospitals monitor patient vitals like heart rate, blood pressure, and temperature. These individual measurements, known as *channels*, together provide a rich view of the monitored system.

This work has received funding from the European Union's Horizon Europe research and innovation programme STELAR under grant agreement No. 101070122.

One fundamental operation on time series data is similarity search - finding time series that most closely match a given query sequence (referred to as its *Nearest Neighbors*) [15], [16]. Besides its direct applications [10], [12], [17], [26], [28], [31], [37], [38], [43], similarity search serves as a building block for more complex tasks like outlier detection [9], [12], [46], classification [4], [20], [40], [44], [51], [45], and clustering [5], [8], [13], [23], [42].

However, performing similarity search on large collections of multivariate time series poses significant computational challenges [35], [48], [52]. The already high dimensionality (i.e., length) of time series is multiplied by the number of channels, dramatically increasing computational costs. Moreover, the multivariate nature introduces unique requirements: users need the flexibility to select which channels to query, and appropriate distance measures and normalization processes need to be applied to account for the different scales and distortions that may exist across channels [14], [44], [51], [45].

For univariate time series, indexing techniques have successfully addressed the dimensionality challenge by utilizing compact summaries with data structures that enable fast similarity lookups [10], [12], [17], [22], [26], [37], [43], [48]. However, the work on MTS is still rudimentary, with limited solutions that are restricted in different ways: they either (i) only support similarity search with queries of a fixed size [47], (ii) require the user to specify the channels to be searched in advance [51], or (iii) they rely on severely restrictive assumptions of the user query, such as user-defined thresholds on each channel [7]. These limitations make current methods impractical for real-world scenarios, where analysts may not know in advance which channels are relevant or may want to search for subsequences of different lengths within the time series.

To address these challenges, we present MULISSE, the first algorithm for variable-length k-NN similarity search on multivariate time series. Our method offers complete flexibility in query specification - both length and channel selection can be determined at query time - and supports both normalized and raw subsequence searches.[1] The algorithm guarantees

[1]Note that supporting normalized subsequences is not the same as supporting normalized time series. The latter simply involves normalizing the dataset as a preprocessing step, whereas the former poses key constraints on the solution, making it a much harder problem.

exact results, always returning the true nearest neighbors.

MULISSE builds upon ULISSE [31], the state-of-the-art method for variable-length similarity search in univariate series. We extend ULISSE's envelope-based indexing to the multivariate domain through three key innovations: (i) a unified representation combining per-channel SAX summaries, (ii) a channel-aware node splitting policy that prioritizes the most influential channels, and (iii) integration of the MASS algorithm [33] for efficient exact distance computation.

Our extensive experimental study provides key insights into the performance characteristics of index-based MTS search. Results on synthetic data with a high number of channels demonstrate the potential of our approach, showing order-of-magnitude improvements over sequential baselines. However, experiments on real-world datasets paint a more nuanced picture, with more limited performance benefits. Through detailed analysis, we identify several critical factors affecting the algorithm's efficiency: the relationship between query length and time series length, the number and nature of channels being queried, and various implementation choices such as the method used for exact distance computation and the parameters such as the SAX segment length and the size of envelopes.

These findings serve multiple purposes: they validate the feasibility of index-based approaches for variable-length MTS search, provide practical insights for implementation choices, and most importantly, highlight specific areas where future research could yield substantial improvements. Key opportunities include dynamic symbolic representations, more sophisticated pruning strategies, and a cost model for selecting the most efficient distance computation method based on query parameters.

The remainder of this paper is organized as follows: We begin with a review of the preliminary concepts and related work (Section II). We then present the MULISSE algorithm, introducing its index structure and the search process (Section III). Next, we evaluate MULISSE on several datasets and compare it to sequential baselines (Section IV). We conclude with a discussion of the results and future work directions (Section V).

II. Preliminaries

A. Definitions and Notation

Time series. A univariate time series $\boldsymbol{t}$ of length m is denoted as $\boldsymbol{t} = [\boldsymbol{t}_1, ..., \boldsymbol{t}_m]$, where $\boldsymbol{t}_i$ is the data point at time i. A multivariate time series $\boldsymbol{T}$ with c channels is denoted as a matrix $\boldsymbol{T} = [\boldsymbol{T}^{(1)}, ..., \boldsymbol{T}^{(c)}]^T$, where $\boldsymbol{T}^{(i)}$ is the univariate time series of channel i. A subsequence of a multivariate time series $\boldsymbol{T}$ of length l starting at timepoint o is denoted as $\boldsymbol{T}_{o,l} = [\boldsymbol{T}_o, ..., \boldsymbol{T}_{o+l-1}]$, where $1 \leq o \leq m$ and $1 \leq l \leq m - o + 1$. The number of available channels in the dataset is denoted as c, and the set of query channels is denoted as $c_{\boldsymbol{Q}}$, with $c_{\boldsymbol{Q}} \subseteq \{1, \ldots, c\}$.

Distance In line with previous studies on MTS similarity search [4], [42], [44], [45], [47], [51], we use Euclidean distance (ED) to measure the distance between time series, which is defined over MTS as:

$$d(\boldsymbol{X}, \boldsymbol{Y}) = ||\boldsymbol{X} - \boldsymbol{Y}||_2 = \sqrt{\sum_{i \in c_{\boldsymbol{XY}}} \sum_{j=1}^{m} \left(\boldsymbol{X}_j^{(i)} - \boldsymbol{Y}_j^{(i)} \right)^2} \quad (1)$$

with $c_{\boldsymbol{XY}} = c_{\boldsymbol{X}} \cap c_{\boldsymbol{Y}}$ the common channels of $\boldsymbol{X}$ and $\boldsymbol{Y}$, and $m = \min(m_{\boldsymbol{X}}, m_{\boldsymbol{Y}})$ the length of the shortest time series.

Similarity Search Queries There are two types of similarity search queries; a k-Nearest-Neighbor (k-NN) query and an r-range query (also called threshold query). A k-NN query aims to identify the k time series within a dataset $\mathcal{D}$ that exhibit the smallest distances from a query time series $\boldsymbol{Q}$ [15]. In the multivariate context, while all time series in $\mathcal{D}$ contain the same c channels, the query time series $\boldsymbol{Q}$ may include only a subset of these channels $c_Q \subseteq 1, \ldots, c$. An r-range query, alternatively, identifies all time series in $\mathcal{D}$ whose distance from query time series $\boldsymbol{Q}$ falls below a specified threshold r [15]. These query types can be further divided into whole-matching and subsequence search queries. Whole-matching evaluates the distance between complete time series, requiring both query and candidate series to have identical lengths [15]. In contrast, subsequence search considers the distances between an entire query series and all subsequences of a candidate series with the same length as the query [15]. This case allows the candidate series from which the subsequences are extracted to vary in length from the query. Our work addresses the more complex problem of subsequence search.

B. Indexing Symbolic Aggregate Approximations (iSAX)

The key challenge in efficient time series mining is the high dimensionality of the data [17], [11]. To address this, the Symbolic Aggregate Approximation (SAX) technique was introduced by Lin et al. [30]. SAX transforms a time series into a symbolic representation by dividing the data into ω equal-length segments and summarizing each segment with the mean value to form Piecewise Aggregate Approximations (PAA) [25], and discretizing the PAA values into a set or *vocabulary* of symbols of size α. To illustrate, consider a time series $\boldsymbol{t} = [1, 2, 5, 4, 7, 5, 6, 2]$ with $\omega = 2$ and the discretization map $\{[1, 3] \rightarrow A, [3, 5] \rightarrow B, [5, 7] \rightarrow C, [7, 9] \rightarrow D\}$. The SAX representation of $\boldsymbol{t}$ is derived by computing its PAA segments $\tilde{\boldsymbol{t}} = [1.5, 4.5, 6, 4]$, and mapping these to the corresponding symbols, resulting in the SAX string $ABCB$ (also known as the *SAX word*). Note that in practice, the symbols are represented with bitstrings rather than characters to reduce and speed up comparison of SAX words [11]. The SAX transformation is further illustrated in Figure 1. Transforming time series into SAX words decreases the dimensionality to ω, while enabling lower-bounding of the Euclidean distance based on their SAX words with a fraction of the original cost [30], [43], [11]. Particularly, given two time series $\boldsymbol{t}$ and $\boldsymbol{q}$, both of length m, their Euclidean distance $d(\boldsymbol{t}, \boldsymbol{q})$ can be

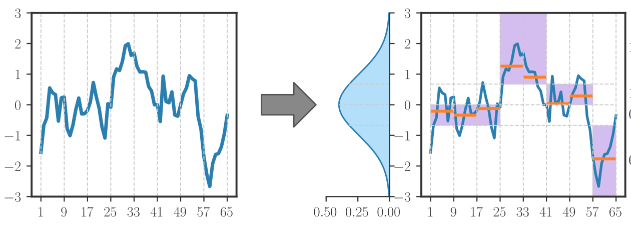
Fig. 1: Visualization of SAX transformation

lower-bounded through their PAA representations $\tilde{\boldsymbol{t}}$ and $\tilde{\boldsymbol{q}}$ as follows [25];

$$d(\boldsymbol{t},\boldsymbol{q}) \leq \sqrt{\frac{m}{\omega}}\sqrt{\sum_{i=1}^{\omega}(\tilde{\boldsymbol{t}}_i - \tilde{\boldsymbol{q}}_i)^2} \tag{2}$$

Then, for SAX representations, the lower bound distances between each symbol pair can be derived by the respective boundaries of the discretization intervals, optionally precomputed and cached in a lookup table for fast retrieval [30].

This opened the door for iSAX, an index for SAX representations that allows for efficient pruning to answer similarity search queries [11]. iSAX is a tree-based index where each node corresponds to a SAX word with a specific vocabulary, with the root node representing the entire dataset and the leaves storing individual time series. For a given node in the tree, the children represent the SAX words that can be formed by increasing the vocabulary size of one of the segments in the parent's SAX word. For example, if the parent's word is $ABAA$ with the map at the first segment being $\{[1,10] \rightarrow A, [10,20] \rightarrow B\}$, the map of the children would be $\{[1,5] \rightarrow A, [5,10] \rightarrow B, [10,15] \rightarrow C, [15,20] \rightarrow D\}$, with, say, the children's SAX words being $ABAA$, $BBAA$, and $CBAA$. As such, the approximation budget increases with each level of the tree, allowing for more accurate distance calculations as the search progresses, and high pruning power high up in the tree [11]. The insertion process of a time series into an iSAX tree is illustrated in Figure 2.

Normalization To ensure a balanced iSAX tree, the authors proposed to normalize the time series before applying the SAX transformation [11], and choosing the discretization intervals (referred to as *breakpoints*) based on the standard normal distribution. This ensures that every SAX symbol is equiprobable to occur in the data. While the SAX-based bounds still hold when the data is not normalized, the breakpoints will have to be adjusted to conform to the data distribution of the original data, in order to ensure a balanced tree. The latter can be tricky, as it requires an additional pass over the data to obtain an empirical distribution before extracting the SAX representations. This increases the cost of the indexing process, and may cause problems when new data arrives that does not conform to the original distribution. In light of this, both SAX and iSAX were proposed solely in context of normalized data.

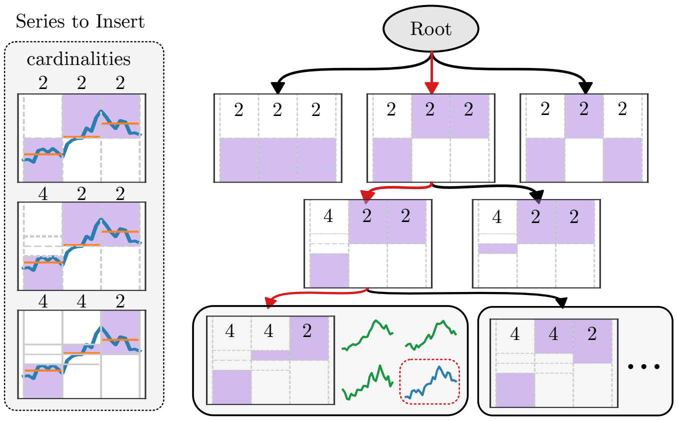

Fig. 2: Visualization of iSAX tree insertion

C. ULISSE

The ULtra compact Index for variable-length Similarity SEarch (ULISSE) is a state-of-the-art method that extends iSAX to support subsequence search with variable-length queries on univariate time series [31]. ULISSE, like other subsequence search indexes, define an allowed range of query lengths $[l_{\min}, l_{\max}]$, but answer these queries, unlike other methods, with a single index. The method relies on indexing multiple candidate subsequences of different lengths as a single entity in an iSAX2+ index, where the iSAX2+ index is an adaptation of the original iSAX, optimized for batch insertion (i.e., bulk loading) and adopting a new node splitting strategy to make for a more balanced tree [11]. The grouping of subsequences relies on two observations: (i) all subsequences of length $[l_{\min}, l_{\max}]$ that *start* at the same timepoint have the same "SAX prefix" (i.e., the first ω symbols of the SAX word), meaning that one can only index the longest $l_{\max}$-length subsequences and derive the shorter ones from these; and (ii) subsequences with similar offsets (also called *time-neighbouring* subsequences) tend to be similar [17], [31], so envelopes derived over their PAA segments are likely tight. These observations are exploited by only indexing the $l_{\max}$-length subsequences (called *master series*), and indexing time-neighbouring master series together through the use of *envelopes*, which are the series composed of the minimum and maximum PAA values of the corresponding master series. This concept is illustrated in Figure 3. To enable the use of envelopes, the iSAX index is adapted s.t. each node stores both a minimum and maximum SAX word, which are used to compute lower-bounds on the distance to the query similar to the process of computing distances from a point to minimum bounding rectangles (MBRs) in spatial databases [19], [6]. This way, ULISSE was shown to outperform the UCR Suite [38] and MASS [33] under a wide range of query lengths and datasets [31].

D. Related Work

Whole search indexes. Literature includes several techniques for time series indexing for both univariate [37], [43], [49], [32], [18], [48] and multivariate time series [47], [51]. These methods are all built on the same principle: they first re-

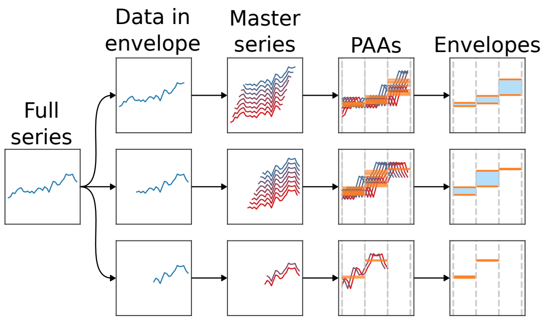

Fig. 3: Visualization of the enveloping process in ULISSE

duce the dimensionality of the time series by applying some summarization technique (e.g., Discrete Fourier Transform (DFT) [32], Piecewise Aggregate Approximation [25], Symbolic Aggregate Approximation [11]), and then index these summaries in a way that allows pruning of series that are dissimilar to the query. However, all the approaches mentioned above are designed for whole-matching queries, and require the length of the query to be decided before index creation. To enable variable-length subsequence search, an index must be created for each possible query length, requiring significant storage when looking to support a wide range of query lengths.
Subsequence search indexes. Besides ULISSE (cf. Section II-C), several other methods have been proposed for subsequence search in univariate time series. For example, Faloutsos et al. [17] introduced the first index for subsequence search on univariate time series, the *ST-index*. Their approach involves constructing a DFT approximation for each subsequence in the dataset, and indexing these approximations in an R-tree [6]. In that, it is able to answer subsequence search queries efficiently, but the method requires the length of the query to be known in advance, making it unsuitable for our problem setting. In contrast, KV-Match [50] is a recent index that supports subsequence search with variable-length queries on univariate time series. Unfortunately, the nature of their approach forces the authors to employ a constrained definition of queries on normalized subsequences to ensure competitive query times, limiting the generality of the method. Furthermore, the index is designed to index subsequences sourcing from a single time series, with no evidence or guarantee that the method remains performant when indexing subsequences from multiple time series. To the best of our knowledge, currently no method exists that is designed for k-NN subsequence search on multivariate time series, neither with fixed nor variable-length queries.
Sequential scan techniques. As an alternative to indexing, several methods have been proposed that rely on sequential scan to answer similarity search queries. These methods generally involve highly optimized distance computations, or pruning techniques that allow for early termination of the search process. For example, the UCR Suite [38] is an algorithm that relies on a wide range (i.e., "suite") of optimizations to speed up subsequence search on univariate time series under Euclidean distance (ED) and Dynamic Time Warping (DTW) [41]. Another notable method is Mueen's Algorithm for Similarity Search (MASS) [33], which relies on the convolution theorem [2] to compute the Euclidean distances between a query $\boldsymbol{Q}$ and all $|\boldsymbol{Q}|$-length subsequences in a candidate time series $\boldsymbol{T} \in \mathbb{R}^m$ in $O(m \log m)$ time rather than $O(m * |\boldsymbol{Q}|)$ time. Both methods can be naturally extended to multivariate time series by utilizing multivariate variants of ED and DTW (cf. [45]) in the case of UCR Suite, and by performing MASS on each channel separately and summing the results in the case of MASS. While these works have shown that sequential scans can offer a more robust alternative to indexing, they obtain their competitive advantage from long time series (i.e., $m \gg |\boldsymbol{Q}|$). In the case of datasets with many short time series, indexes have shown to scale significantly better [31]. As we aim to design a generic method that can handle both long and short time series, we opt for an index-based approach.

III. Methodology

Variable-length similarity search for multivariate time series involves three key challenges. First is the added dimensionality of the multivariate case; while subsequence search already faces rapid growth in candidate subsequences, the presence of multiple channels multiplies the cost of distance computations, making effective pruning and efficient distance computation critical. Second, variable-length queries further expand the search space, as subsequences of different lengths between $l_{\min}$ and $l_{\max}$ must be considered. Finally, supporting ad-hoc selection of query channels prevents utilizing traditional dimensionality reduction techniques like Principal Component Analysis (PCA) [51], as these would combine channels in ways that make individual channel selection impossible.

Our solution, named MULISSE, extends SAX representations to the multivariate case, while building upon ULISSE's indexing framework for variable-length subsequence search [31]. MULISSE creates individual SAX summaries for each channel and combines them into a unified representation, enabling fast lower-bounding of distances while keeping the channel-axis intact to support ad-hoc channel selection. The method adapts iSAX2+'s node splitting to prioritize the most selective channels, ensuring balanced tree growth and effective pruning. Furthermore, recognizing that distance computation costs vary with query parameters, MULISSE integrates MASS [33] as an alternative to brute-force distance computation, particularly beneficial for shorter queries. The resulting solution supports both raw and normalized subsequence searches, and has the option for an approximate search mode that trades off accuracy for speed. We now present the algorithm in detail, starting with multivariate SAX representations (Section III-A), followed by the index structure (Section III-B), and finally the extension to ULISSE and the search process (Section III-C).

A. Multivariate SAX Representations (mSAX)

As mentioned in Section II, one of the primary challenges in time series is their length, which make distance computations between the query and the database series computationally expensive, as it requires passing over each element in the series. Summarizing the time series through a symbolic representation, such as SAX, allows us to approximate this distance for a fraction of the cost, as the size of the representation ω is significantly smaller than the length of the series m. In the univariate case, SAX represents the time series as a sequence of ω symbols, where each symbol represents a segment of size (at most) $\frac{m}{\omega}$. Formally, given a univariate time series $\boldsymbol{t} \in \mathbb{R}^m$, SAX is a transformation $f : \mathbb{R}^m \rightarrow \Phi^\omega$ that maps $\boldsymbol{t}$ to a sequence of ω symbols $\tilde{\boldsymbol{t}} \in \Phi^\omega$, where $\Phi = \{a_1, a_2, \ldots, a_\alpha\}$ is the alphabet of symbols, and α is the cardinality of the alphabet. In the multivariate case, this transformation is applied along the channel-axis, resulting in a SAX word for each channel. Formally, this extends f to $f : \mathbb{R}^{m \times c} \rightarrow \Phi_i^{\omega_i} \times \ldots \times \Phi_c^{\omega_c}$, where c is the number of channels in the MTS, and Φ_i is the alphabet of symbols for channel i. Note that this definition allows for different alphabets and word lengths for each channel, enabling the user to tune the representation to the characteristics (e.g., scale, noise) of each channel, to improve the quality of the representation.

B. Multivariate SAX index (mISAX)

To enable efficient similarity search on MTS, we extend the iSAX2+ index to handle multiple channels. The resulting multivariate iSAX (mISAX) index maintains the tree structure of iSAX2+, but adapts the node representation and splitting strategy to accommodate multiple channels while ensuring balanced tree growth. Each node in the mISAX tree stores a full mSAX representation with SAX words for all channels in the dataset, even though queries may only use a subset of these channels. This design choice allows for ad-hoc channel selection at query time without requiring multiple indexes or additional data structures. Similar to iSAX2+, the SAX symbols throughout the representation can have different cardinalities, even when two symbols correspond to the same time interval in different channels. Furthermore, the splitting approach remains the same, where the children of each node increase the cardinality of one SAX symbol in the parent node (referred to as *symbol promotion*) [43].

The splitting strategy in mISAX differs from that in iSAX2+ though; where iSAX2+ determines the next symbol to promote with a rule-based approach on the mean and standard deviation of the distribution of PAA values within each segment, we take a more general approach in mISAX [11]. Namely, when a leaf node exceeds its capacity, mISAX computes a selectivity score for each segment based on the entropy of the distribution of symbols *if the cardinality were multipled by two* (i.e., "promoted"). Particularly, the selectivity score S_i for segment i is calculated as:

$$S_i = \left(- \sum_{j=1}^{2*\alpha_i} p_{ij} \log p_{ij} \right) * \sigma_i \tag{3}$$

where p_{ij} is the empirical probability of symbol j occurring in segment i under a cardinality of $2 * \alpha_i$, derived from the time series stored in the node, and σ_i is the standard deviation of PAA values in segment i. The segment with the highest selectivity score (i.e., most uniform distribution) is chosen for splitting, as it is likely to result in the most balanced partitioning of the data. Notice that because we are effectively treating the segments of different channels as independent, the selectivity score is computed separately for each segment in each channel. The reason to include the standard deviation of the underlying data as a weight in the score is to ensure that segments with higher variance – and therefore a higher impact on distances – are more likely to be selected for splitting. While this weighting will likely be fairly uniform for normalized data, it is crucial for raw data, where the variance can differ significantly between channels.

To maintain the lower-bounding property essential for exact search, mISAX extends the distance calculations to account for multiple channels:

$$MINDIST_N(\boldsymbol{Q}) = \sqrt{\sum_{i \in c_{\boldsymbol{Q}}} MINDIST(\tilde{\boldsymbol{q}}_i, w_i)^2} \tag{4}$$

where $MINDIST(\tilde{\boldsymbol{q}}_i, w_i)$ is the standard iSAX minimum distance between query channel i and the corresponding SAX word in node N, and $c_{\boldsymbol{Q}}$ is the set of query channels.

C. Multivariate ULISSE (MULISSE)

Building upon mISAX, we extend ULISSE to handle multiple channels by adapting its envelope-based representation to the multivariate case. Similar to how mSAX extends the SAX representation with an additional channel axis, MULISSE extends ULISSE's envelopes to maintain minimum and maximum PAA values per channel. Formally, given a set of time-neighbouring master series, their multivariate envelope $\boldsymbol{E}$ is defined as:

$$\boldsymbol{E}_{a,\gamma} = [\boldsymbol{E}_{a,\gamma}^{(1)}, ..., \boldsymbol{E}_{a,\gamma}^{(c)}]^T \tag{5}$$

where $\boldsymbol{E}_{a,\gamma}^{(i)}$ is the envelope of channel i with fill factor γ and offset a, consisting of the minimum and maximum PAA values for each segment in that channel. The lower bound distance between a query $\boldsymbol{Q}$ and an envelope $\boldsymbol{E}$ are then computed as:

$$MINDIST_E(\tilde{\boldsymbol{Q}}, \boldsymbol{E}) = \sqrt{\sum_{i \in c_{\boldsymbol{Q}}} MINDIST_E(\tilde{\boldsymbol{Q}}^{(i)}, \boldsymbol{E}^{(i)})^2} \tag{6}$$

where $\tilde{\boldsymbol{Q}}$ is the PAA representation of the query, and $MINDIST_E(\tilde{\boldsymbol{Q}}^{(i)}, \boldsymbol{E}^{(i)})$ is ULISSE's original envelope-based minimum distance computation for channel i. This simple extension preserves all of ULISSE's theoretical properties while enabling efficient multivariate search.

Search Process. The search process follows ULISSE's best-first tree traversal using a priority queue, adapted to use our multivariate distance computations (visualized in Figure 4). A key enhancement is the integration of MASS [33] for exact distance computation when appropriate. Specifically, when processing candidates from an envelope, we can choose between two strategies: (a) brute-force computation with early abandoning, which is effective when the query length is

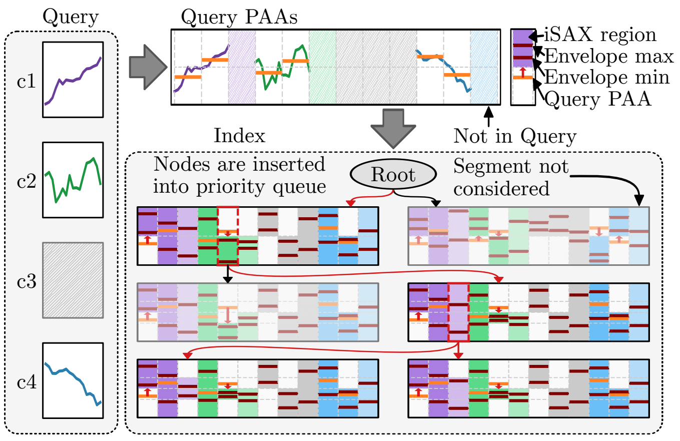

Fig. 4: Visualization of search in MULISSE

close to the length of the whole time series, or (b) MASS-based computation, which is more efficient when the envelope contains many candidate subsequences from the same time series. Particularly, due to its $O(c * m \log m)$ complexity for computing the distances between a query $\boldsymbol{Q}$ and all $|\boldsymbol{Q}|$-length subsequences of a time series $\boldsymbol{T} \in \mathbb{R}^{m \times c}$, MASS is particularly beneficial for shorter queries and longer time series, where the number of candidates is larger. In our evaluation (Section IV), we explore the performance of MULISSE under both strategies and investigate the trade-offs between them.

IV. Evaluation

We conducted experiments with two main objectives: evaluating MULISSE's scalability and efficiency across different configurations, and comparing it against existing approaches. Since all methods provide exact results, we focus solely on performance metrics.

Compared methods. Given that no existing methods support variable-length k-NN subsequence search for MTS, we compare against two adapted sequential scan approaches: (a) **Brute-force**: A naive approach comparing the query against all possible subsequences, (b) **MASS**: An adaptation of MASS [33] that performs MASS on each channel of the MTS, squaring and summing the results to obtain the final distance. These baseline methods are compared to two variants of MULISSE: (c) **MULISSE+BF**, which computes the exact distances of subsequences returned by the index in an exhaustive manner, and (d) **MULISSE+MASS**, which computes exact distances through MASS, by grouping and merging all candidate subsequences by their time series ID, and then passing the resulting subsequence to MASS. As an additional optimization, both Brute-force and MULISSE+BF utilize *early abandoning* to stop distance computations when the distance exceeds the current k-th nearest neighbor distance. Furthermore, for MASS and MULISSE+MASS, the DFTs of the full MTS were computed offline, as part of the preparation phase, such that at query time, only the element-wise multiplication and inverse DFT were required to compute the distances between the query and subsequences from dataset MTS.

TABLE I: Default query configurations.

	Stocks	**Weather**	**Synthetic**
Number of MTS n	5000	5000	100
Series length m	2048	2048	2048
$[l_{\min}, l_{\max}]$	[1560,2048]	[1560,2048]	[1560,2048]
Number of channels c	5	4	1024

Parameters. For MULISSE, we use an envelope size of $\gamma = m - l_{\min} - 1$, as it showed to be optimal in the evaluation of original ULISSE paper [31]. Note that this implies that one envelope covers all candidate subsequences in a time series. Furthermore, we utilized a segment length of 32 as that was the default in the code provided by the authors of ULISSE [31]. Brute-force and MASS are parameter-free.

Hardware and implementations. Experiments ran on a server equipped with a 64-core AMD Genoa 9654 (2.4 GHz) processor and 64 GB of RAM. All algorithms were implemented in C++ and compiled with GCC 12.3.0., running single-threaded in main memory for a fair and unbiased comparison. Source code is available at GitHub.

Datasets. We utilized three datasets: (a) **Stocks**: 28678 stocks with daily metrics (1987-2021), averaging 5590 observations per MTS. (b) **Weather**: ISD dataset [34] containing hourly measurements from 13545 sensors (2020-2021), averaging 8692 observations. (c) **Synthetic**: Random walk data with 100 MTS, each having 1024 channels and 4096 observations, generated following [15], [16] methodology. To control the time series length m, we ensure that all time series have the same length by truncating and ignoring time series with less than m observations. Table I summarizes the key properties for each dataset. We include more details about the datasets in GitHub.

Queries and evaluation metrics. Query workloads were generated by selecting random subsequences of length $|Q|$ from the datasets and adding Gaussian noise with standard deviation $0.1 * \sigma_{Q_i}$ on all channels. Query lengths were sampled uniformly from $[l_{\min}, l_{\max}]$. Unless otherwise stated, we query on all available channels. We focus on the Stocks dataset as our primary test case due to its size, using other datasets for specific comparisons. Lastly, as mentioned in Section III-B, we only query for **z-normalized subsequences** in this evaluation. Querying on raw data was not considered as it requires derivation of iSAX breakpoints based on empirical distributions of the data, for which there currently exist no principled approach. We measure preparation time (indexing and/or DFT pre-computation) and query execution time, reporting median values over 100 queries.

A. Query length

Figure 5(a) shows the impact of query length on query execution time. Both Brute-force and MULISSE+BF exhibit improved performance as query length increases, which is expected as longer queries reduce the number of possible subsequences per MTS to compare against the query. At the extreme case where $|\boldsymbol{Q}| = 2048$, the search simplifies to whole series matching with just one subsequence per time series. In contrast, MASS and MULISSE+MASS demonstrate

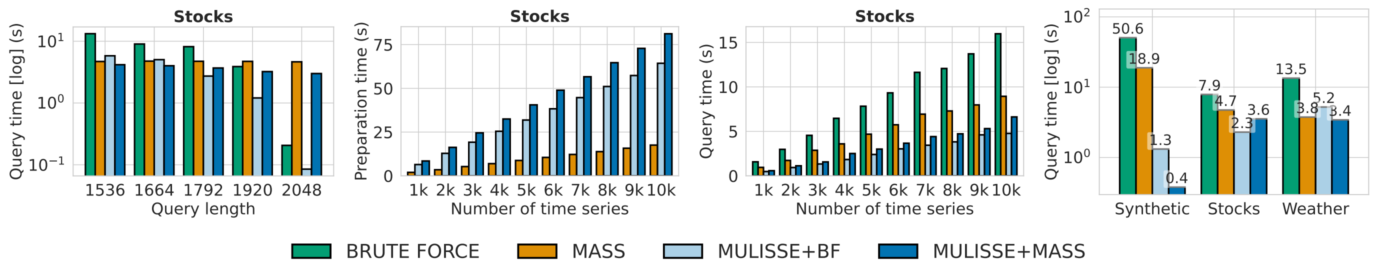

Fig. 5: Sensitivity of algorithms to query parameters under $l_{\min} = 1560$ and $l_{\max} = 2048$: (a) Query time over query length; (b) Preparation time over number of time series n; (c) Query time over number of time series n; (d) Query time over different datasets.

query length invariance. For MASS, this aligns with its complexity of $O(c * m \log m)$, with no dependency on $|\boldsymbol{Q}|$. Interestingly, MULISSE+MASS's pruning effectiveness remains stable across query lengths, suggesting that pruning power is more influenced by the supported length range $[l_{\min}, l_{\max}]$ and series length m, which influence the tightness of envelopes.

These findings highlight that the choice between Brute-force and MASS-based exact distance computation significantly impacts performance of MULISSE, with the optimal choice depending on query length. This insight suggests potential for a hybrid approach that selectively employs either method based on the expected cost of distance computation, which relies on the length of the query, the candidate subsequences, the probability of early abandoning, and availability of cached DFTs.

B. Number of time series

Figure 5(b) illustrates how preparation time scales with dataset size. All methods exhibit linear scaling with the number of time series, as expected. MASS requires the least preparation time, followed by MULISSE+BF, and then MULISSE+MASS. The relatively low cost of DFT pre-computation compared to index construction suggests that one should seriously consider the amount of queries planned to execute when choosing between MASS or MULISSE, as the added indexing cost of MULISSE will need to be amortized across multiple queries.

Looking now at query time, Figure 5(c) shows that MULISSE+BF achieves the fastest query times, followed by MULISSE+MASS, MASS, and Brute-force. This ordering aligns with our findings in Section IV-A and the fact that query lengths are uniformly sampled from $[l_{\min}, l_{\max}]$. However, it's worth noting again that the relative performance of MULISSE+BF and MULISSE+MASS can vary significantly with $|\boldsymbol{Q}|$, with MULISSE+MASS likely outperforming for shorter queries relative to the full series length m.

Furthermore, all methods demonstrate linear scaling with dataset size at similar rates, indicating that MULISSE's pruning effectiveness remains consistent regardless of the number of time series. This consistent performance gap between index-based and sequential methods suggests the robustness of our indexing approach across different dataset sizes.

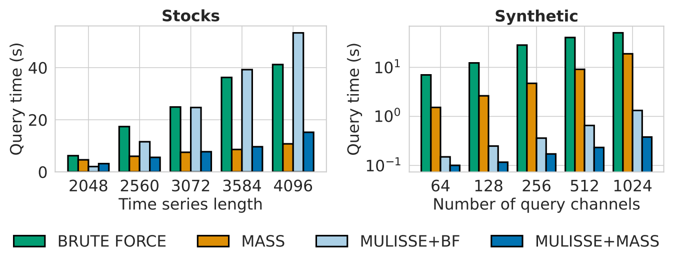

Fig. 6: (a) Query time over query length; (b) Query time over number of channels.

C. Different datasets

Figure 5(d) reveals varying performance gains across different datasets. MULISSE's advantage over baselines ranges from two orders of magnitude on the Synthetic dataset to approximately 2x on Stocks and merely 10% on Weather data. This substantial variation in pruning power can be attributed to the interaction between the fixed segment length (32) and the varying number of channels across datasets (1024, 5, and 4 for Synthetic, Stocks, and Weather, respectively).

The superior performance on the Synthetic dataset stems from its larger number of channels, resulting in more SAX symbols and consequently more accurate distance approximations. This is a key insight, as it emphasizes the potential relevance of tuning SAX parameters like the segment length to the dataset characteristics. Moreover, as channels can have a different impact on the distance, it might even be beneficial to use different segment lengths for different channels, a direction for future research. While the current performance gains on real-world datasets are modest, these results demonstrate the potential of index-based approaches and highlight opportunities for future improvements.

D. Time series length

Figure 6(a) demonstrates how query time scales with time series length while maintaining a fixed query range of $[l_{\min}, l_{\max}] = [1560, 2048]$. Both MULISSE variants show steeper scaling compared to their sequential counterparts, with MULISSE+BF's growth particularly pronounced. Detailed analysis reveals that MULISSE+BF and MULISSE+MASS's pruning power deteriorate significantly as m increases from 2048 to 2580, approaching negligible pruning at $m = 4096$.

This degradation stems from the linear increase in subsequences covered by each envelope as series length grows, resulting in wider bounds and reduced pruning effectiveness. Consequently, while MULISSE not only has to exhaustively consider a larger portion of the search space, the search space itself also grows, resulting in a quadratic increase in query time. MULISSE+MASS moderates this growth somewhat through more efficient distance computation for the increased number of candidate subsequences per time series. For MULISSE+BF, the quadratic growth results in a query time larger than Brute-force for $m = 4096$, caused by the lack of pruning, the additional overhead of index traversal and a lower early abandoning rate.

These results suggest that ULISSE's approach of using $\gamma = m - l_{\min} - 1$ may not be optimal for the multivariate case, where multiple channels contribute to envelope width. Future work should therefore reconsider parameterization for MULISSE, potentially even exploring alternatives to enveloping to control the iSAX tree size and maintain pruning effectiveness across different series lengths.

E. Number of query channels

Figure 6(b) shows all algorithms scaling linearly with the number of channels at similar rates. While expected for sequential approaches, this consistent scaling in MULISSE indicates that pruning power remains stable across different numbers of channels. This behavior is relatively intuitive for the Synthetic dataset, where uncorrelated channels provide no additional pruning information.

The impact of channel count might differ in real-world datasets with correlated channels, where additional channels lead to a higher *relative contrast* of the queries [1]. Relative contrast is the ratio between the distance of the query to the closest and farthest indexed entity, with lower values indicating more difficult queries for index-based methods. While our current datasets' limited channel counts preclude thorough investigation of this aspect, future work could explore how correlated channels might enhance index-based method performance through increased query selectivity.

F. Summary of Findings

Our comprehensive evaluation reveals several key insights about variable-length subsequence search in MTS:

1) **Exact Distance Computation Trade-offs:** The choice between brute-force and MASS-based exact distance computation significantly impacts performance, with the optimal choice depending on query length and time series characteristics. This suggests potential benefits from a hybrid approach that dynamically selects the most efficient method.
2) **Scalability Characteristics:** While all methods scale linearly with the number of time series and channels, MULISSE's performance advantage varies significantly across datasets. The method shows particular promise for datasets with many channels, where its symbolic approximations capture more information.
3) **Parameter Sensitivity:** The effectiveness of MULISSE is heavily influenced by its parameters, particularly the envelope size and segment length. The current approach of using $\gamma = m - l_{\min} - 1$ and fixed segment lengths may not be optimal for the multivariate case, suggesting room for improvement through new parameterization.
4) **Real-world Performance:** While achieving dramatic speedups on synthetic data, performance gains on real-world datasets are more modest (10% to 2x). This gap highlights both the potential of index-based approaches and the need for further research to improve their effectiveness on practical applications.

V. Conclusions and future work

We have presented MULISSE, the first exact algorithm for variable-length k-NN subsequence search in multivariate time series. By extending ULISSE's enveloping technique with a multivariate iSAX representation and introducing a more generic, channel-aware node splitting, our method effectively handles the additional complexity introduced by multiple channels. Furthermore, through the integration of MASS for distance computation, MULISSE provides efficient filtering of false positives, especially when the query is small compared to the length of time series in the dataset.

Our evaluation reveals that MULISSE can achieve significant speedups over baseline methods, ranging from 10% to two orders of magnitude depending on the dataset characteristics. However, the results also highlight several important dependencies on query parameters, such as dataset, query length, and time series length. This opens up opportunities for future work to improve the algorithm's performance and robustness across a wider range of scenarios. In particular, we identify the following directions for future research:

- Development of a cost model to dynamically select between Brute-force and MASS-based distance computation based on query parameters.
- Investigation of dynamic SAX representations that adjusts segment lengths based on the number of channels and their relative importance.
- Reconsideration of the role of envelopes in the algorithm and exploration of alternative methods to control the size of the index.
- Extended evaluation with a thorough parameterization phase to determine optimal parameter values for each dataset and consideration of real-world datasets with a large number of channels.

While there is room for improvement, particularly for real-world datasets with fewer channels, MULISSE represents an important step towards efficient multivariate time series search. Its ability to handle variable-length queries and ad-hoc channel selection makes it particularly suitable for exploratory data analysis across various domains, from financial analytics to healthcare monitoring.

References

[1] Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: ICDT'01 (2001)

[2] Arfken, G.: Mathematical Methods for Physicists, third edn. Academic Press, Inc., San Diego (1985)

[3] Bach-Andersen, M., Rømer-Odgaard, B., Winther, O.: Flexible non-linear predictive models for large-scale wind turbine diagnostics. Wind Energy **20**, 753–764 (2017). URL https://api.semanticscholar.org/CorpusID:114702958

[4] Bagnall, A.J., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.J.: The UEA multivariate time series classification archive, 2018. CoRR **abs/1811.00075** (2018). URL http://arxiv.org/abs/1811.00075

[5] Bagnall, A.J., Janacek, G.J.: Clustering time series from arma models with clipped data. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, p. 49–58. Association for Computing Machinery, New York, NY, USA (2004)

[6] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The r*-tree: an efficient and robust access method for points and rectangles. SIGMOD Rec. **19**(2), 322–331 (1990). DOI 10.1145/93605.98741. URL https://doi.org/10.1145/93605.98741

[7] Bhaduri, K., Zhu, Q., Oza, N.C., Srivastava, A.N.: Fast and flexible multivariate time series subsequence search. In: 2010 IEEE International Conference on Data Mining, pp. 48–57 (2010). DOI 10.1109/ICDM.2010.36

[8] Bonifati, A., Buono, F.D., Guerra, F., Lombardi, M., Tiano, D.: Interpretable clustering of multivariate time series with time2feat. Proc. VLDB Endow. **16**(12), 3994–3997 (2023). DOI 10.14778/3611540.3611604

[9] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. SIGMOD Rec. **29**(2), 93–104 (2000). DOI 10.1145/335191.335388. URL https://doi.org/10.1145/335191.335388

[10] Camerra, A., Palpanas, T., Shieh, J., Keogh, E.: isax 2.0: Indexing and mining one billion time series. In: 2010 IEEE International Conference on Data Mining, pp. 58–67 (2010). DOI 10.1109/ICDM.2010.124

[11] Camerra, A., Palpanas, T., Shieh, J., Keogh, E.: isax 2.0: Indexing and mining one billion time series. In: 2010 IEEE International Conference on Data Mining, pp. 58–67 (2010). DOI 10.1109/ICDM.2010.124

[12] Dallachiesa, M., Palpanas, T., Ilyas, I.F.: Top-k nearest neighbor search in uncertain data series. Proc. VLDB Endow. **8**(1), 13–24 (2014). DOI 10.14778/2735461.2735463. URL https://doi.org/10.14778/2735461.2735463

[13] Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., Zhang, D.: Yading: fast clustering of large-scale time series data. Proc. VLDB Endow. **8**(5), 473–484 (2015)

[14] d'Hondt, J.E., Papapetrou, O., Paparrizos, J.: Beyond the dimensions: A structured evaluation of multivariate time series distance measures. In: 2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), pp. 107–112 (2024). DOI 10.1109/ICDEW61823.2024.00020

[15] Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. Proc. VLDB Endow. **12**(2), 112–127 (2018). DOI 10.14778/3282495.3282498

[16] Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: Return of the lernaean hydra: Experimental evaluation of data series approximate similarity search. CoRR **abs/2006.11459** (2020)

[17] Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, SIGMOD '94, p. 419–429. Association for Computing Machinery, New York, NY, USA (1994). DOI 10.1145/191839.191925. URL https://doi.org/10.1145/191839.191925

[18] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., El Abbadi, A.: Vector approximation based indexing for non-uniform high dimensional data sets. In: Proceedings of the Ninth International Conference on Information and Knowledge Management, CIKM '00, p. 202–209. Association for Computing Machinery, New York, NY, USA (2000). DOI 10.1145/354756.354820. URL https://doi.org/10.1145/354756.354820

[19] Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, SIGMOD '84, p. 47–57. Association for Computing Machinery, New York, NY, USA (1984). DOI 10.1145/602259.602266. URL https://doi.org/10.1145/602259.602266

[20] Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A.: Classification of time series by shapelet transformation. Data Min. Knowl. Discov. **28**(4), 851–881 (2014)

[21] Huijse, P., Estevez, P.A., Protopapas, P., Principe, J.C., Zegers, P.: Computational intelligence challenges and applications on large-scale astronomical time series databases. IEEE Computational Intelligence Magazine **9**(3), 27–39 (2014). DOI 10.1109/mci.2014.2326100. URL http://dx.doi.org/10.1109/MCI.2014.2326100

[22] Kadiyala, S., Shiri, N.: A compact multi-resolution index for variable length queries in time series databases. Knowledge and Information Systems **15**(2), 131–147 (2008). DOI 10.1007/s10115-007-0097-z. URL https://doi.org/10.1007/s10115-007-0097-z

[23] Kalpakis, K., Gada, D., Puttagunta, V.: Distance measures for effective clustering of arima time-series. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 273–280 (2001). DOI 10.1109/ICDM.2001.989529

[24] Kashino, K., Smith, G., Murase, H.: Time-series active search for quick retrieval of audio and video. In: 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258), vol. 6, pp. 2993–2996 vol.6 (1999). DOI 10.1109/ICASSP.1999.757470

[25] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems **3**(3), 263–286 (2001). DOI 10.1007/PL00011669. URL https://doi.org/10.1007/PL00011669

[26] Keogh, E., Palpanas, T., Zordan, V.B., Gunopulos, D., Cardle, M.: Indexing large human-motion databases. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30, VLDB '04, p. 780–791. VLDB Endowment (2004)

[27] Knieling, S., Niediek, J., Kutter, E., Bostroem, J., Elger, C., Mormann, F.: An online adaptive screening procedure for selective neuronal responses. Journal of Neuroscience Methods **291**, 36–42 (2017). DOI https://doi.org/10.1016/j.jneumeth.2017.08.002. URL https://www.sciencedirect.com/science/article/pii/S0165027017302832

[28] Kondylakis, H., Dayan, N., Zoumpatianos, K., Palpanas, T.: Coconut: a scalable bottom-up approach for building data series indexes. Proc. VLDB Endow. **11**(6), 677–690 (2018)

[29] Liess, S., Agrawal, S., Chatterjee, S., Kumar, V.: A teleconnection between the west siberian plain and the ENSO region. Journal of Climate **30**(1), 301 – 315 (2017)

[30] Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, p. 2–11. Association for Computing Machinery, New York, NY, USA (2003). DOI 10.1145/882082.882086. URL https://doi.org/10.1145/882082.882086

[31] Linardi, M., Palpanas, T.: Scalable, variable-length similarity search in data series: the ulisse approach. Proc. VLDB Endow. **11**(13), 2236–2248 (2018)

[32] Mueen, A., Nath, S., Liu, J.: Fast approximate correlation for massive time-series data. In: Proc. SIGMOD'10

[33] Mueen, A., Zhing, S., Zhu, Y., Yeh, M., Kamgar, K., Viswanathan, K., Gupta, C., Keogh, E.: The fastest similarity search algorithm for time series subsequences under euclidean distance (2022). http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html

[34] Oceanic, N., Administration, A.: NOAA integrated surface dataset. https://www.ncei.noaa.gov/access/search/dataset-search

[35] Palpanas, T.: Big sequence management: A glimpse of the past, the present, and the future. In: Proceedings of the 42nd International Conference on SOFSEM 2016: Theory and Practice of Computer Science - Volume 9587, p. 63–80. Springer-Verlag, Berlin, Heidelberg (2016). DOI 10.1007/978-3-662-49192-8_6. URL https://doi.org/10.1007/978-3-662-49192-8_6

[36] Paraskevopoulos, P., Dinh, T.C., Dashdorj, Z., Palpanas, T., Serafini, L., et al.: Identification and characterization of human behavior patterns from mobile phone data. D4D Challenge session, NetMob (2013)

[37] Rafiei, D., Mendelzon, A.O.: Efficient retrieval of similar time sequences using DFT. Proceedings 5th International Conference on Foundations of Data Organizations and Algorithms (FODO '98) (1998)

[38] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the

18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, p. 262–270. Association for Computing Machinery, New York, NY, USA (2012). DOI 10.1145/2339530.2339576. URL https://doi.org/10.1145/2339530.2339576

[39] Roddick, J.F., Hornsby, K.S.: Temporal, spatial, and spatio-temporal data mining. In: Lecture Notes in Computer Science (2001). URL https://api.semanticscholar.org/CorpusID:13374115

[40] Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **35**(2), 401–449 (2021). DOI 10.1007/s10618-020-00727-3. URL https://doi.org/10.1007/s10618-020-00727-3

[41] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43–49 (1978). DOI 10.1109/TASSP.1978.1163055

[42] Salarpour, A., Khotanlou, H.: An empirical comparison of distance measures for multivariate time series clustering. International Journal of Engineering **31**(2), 250–262 (2018)

[43] Shieh, J., Keogh, E.: isax: indexing and mining terabyte sized time series. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, p. 623–631. Association for Computing Machinery, New York, NY, USA (2008)

[44] Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I.: Elastic similarity and distance measures for multivariate time series. Knowledge and Information Systems **65**(6), 2665–2698 (2023). DOI 10.1007/s10115-023-01835-4. URL https://doi.org/10.1007/s10115-023-01835-4

[45] Shokoohi-Yekta, M., Hu, B., Jin, H., Wang, J., Keogh, E.J.: Generalizing dynamic time warping to the multi-dimensional case requires an adaptive approach (2014). URL https://api.semanticscholar.org/CorpusID:14584950

[46] Tuli, S., Casale, G., Jennings, N.R.: Tranad: Deep transformer networks for anomaly detection in multivariate time series data. Proc. VLDB Endow. **15**(6), 1201–1214 (2022). DOI 10.14778/3514061.3514067. URL https://www.vldb.org/pvldb/vol15/p1201-tuli.pdf

[47] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.: 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, KDD '03, p. 216–225. Association for Computing Machinery, New York, NY, USA (2003). DOI 10.1145/956750.956777. URL https://doi.org/10.1145/956750.956777

[48] Wang, J., Zhu, Y., Li, S., Wan, D., Zhang, P.: Multivariate time series similarity searching. ScientificWorldJournal **2014**, 851,017 (2014)

[49] Wang, Y., Wang, P., Pei, J., Wang, W., Huang, S.: A data-adaptive and dynamic segmentation index for whole matching on time series. Proc. VLDB Endow. **6**(10), 793–804 (2013). DOI 10.14778/2536206.2536208. URL https://doi.org/10.14778/2536206.2536208

[50] Wu, J., Wang, P., Pan, N., Wang, C., Wang, W., Wang, J.: Kv-match: A subsequence matching approach supporting normalization and time warping. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 866–877. IEEE Computer Society, Los Alamitos, CA, USA (2019). DOI 10.1109/ICDE.2019.00082. URL https://doi.ieeecomputersociety.org/10.1109/ICDE.2019.00082

[51] Yang, K., Shahabi, C.: A pca-based similarity measure for multivariate time series. In: Proceedings of the 2nd ACM International Workshop on Multimedia Databases, MMDB '04, p. 65–74. Association for Computing Machinery, New York, NY, USA (2004). DOI 10.1145/1032604.1032616. URL https://doi.org/10.1145/1032604.1032616

[52] Zoumpatianos, K., Idreos, S., Palpanas, T.: Indexing for interactive exploration of big data series. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, p. 1555–1566. Association for Computing Machinery, New York, NY, USA (2014). DOI 10.1145/2588555.2610498. URL https://doi.org/10.1145/2588555.2610498