MOMENTI: Scalable Motif Mining in Multidimensional Time Series

Matteo Ceccarello*
University of Padova
Padova, Italy
matteo.ceccarello@unipd.it

Francesco Pio Monaco
University of Padova
Padova, Italy
monacofran@dei.unipd.it

Francesco Silvestri
University of Padova
Padova, Italy
francesco.silvestri@unipd.it

ABSTRACT

Time series play a fundamental role in many domains, capturing a plethora of information about the underlying data-generating processes. When a process generates multiple *synchronized* signals we are faced with *multidimensional* time series. In this context a fundamental problem is that of *motif mining*, where we seek patterns repeating twice with minor variations, spanning some of the dimensions. State of the art exact solutions for this problem run in time quadratic in the length of the input time series.

We provide a scalable method to find the top-k motifs in multidimensional time series with probabilistic guarantees on the quality of the results. Our algorithm runs in subquadratic time in the length of the input, and returns the exact solution with probability at least $1-\delta$, where δ is a user-defined parameter. The algorithm is designed to be *adaptive* to the input distribution, self-tuning its parameters while respecting user-defined limits on the memory to use.

Our theoretical analysis is complemented by an extensive experimental evaluation, showing that our algorithm is orders of magnitude faster than the state of the art.

1 INTRODUCTION

Time series play a central role in modeling the evolution of data-generating processes in many domains. To capture the multifaceted nature of processes generating data, time series are oftentimes *multidimensional*: collections of co-evolving signals whose measurements are synchronized, collectively describing the evolution of the process. Extracting information from such multidimensional time series is thus fundamental.

In particular, *top-k motifs* mining is a crucial and challenging problem: intuitively, the goal is to find patterns that occur twice with minor modifications, spanning many, but not all, the signals

---

*Corresponding author.

---

of the time series. More formally, given a time series **T** with $D > 1$ dimensions and n points, the problem consists in finding k pairs of **T**'s subsequences with the smallest distance, where the distance captures the similarity between subsequences. Indeed, similar patterns might imply a particular behavior, making motif discovery a crucial step for higher-level analysis. Applications include forecasts for volcanic eruptions [10], healthcare management [26], and machine management in industry [33]. In particular, multidimensional motif discovery is key in pollution control [27], in quality control in industrial settings [4], in activity discovery [7, 38], and in healthcare [6, 26].

A common approach to motif discovery is to extend the approach for motif mining in 1-dimension time series to the multidimensional case: that is, we look for pairs of subsequences of the time series where all the D dimensions are similar. However, this approach might not reveal interesting patterns: for instance, some dimensions might be noisy or uncorrelated with respect to the others, and they might hide similar patterns involving only a subset of the dimensions that are in general unknown beforehand. We aim to discover patterns that involve only a subset of the D dimensions which is unknown and needs to be retrieved as well. This formulation overcomes the limitation of several previous approximate approaches that assumed all dimensions as equally informative.

Multidimensional motifs can then be discovered with the following naive approach: for each of the 2^D subset of dimensions, we compare the $O(n^2)$ subsequences of **T** on the selected dimensions and we return the k closest ones. This solution forces $O(2^D n^2)$ comparisons, which, for large sets of data, are clearly prohibitive.

In this paper, we propose a scalable and efficient solution that aims at minimizing the number of distance computations to perform. We leverage Locality Sensitive Hashing (LSH), a common technique in similarity search that groups together similar elements. At a high level, we build an index of the time series where each multidimensional subsequence is mapped to a set of LSH hash values: by the properties of LSH this ensures that similar subsequences are more likely to hash to the same values across the dimensions spanned by the motif. Thanks to this index we are able to prune a hefty amount of candidates from the search space. We focus on comparisons based on the *z-normalized Euclidean Distance*, however our approach can be generalized to other similarity measures.

One of the challenges of employing LSH is setting its parameters to ensure that good quality results are retrieved efficiently, since the precise setting of the parameters is data-dependent. To overcome this challenge we design an index that automatically tunes its parameters depending on the data at hand while respecting user-specified limits on the memory to be used.

Our contributions are the following:

- We design an approach for top-k motif discovery in multidimensional time series, named MOMENTI. Our approach is based on Locality Sensitive Hashing and returns exact answers with a user-specified failure probability. Furthermore, we provide a theoretical analysis of the correctness of our approach and on its complexity in terms of distance computations carried out, along with several optimizations to speed up the execution.
- We provide an open source implementation of our approach, that we use to carry out an extensive experimental evaluation. We show that our approach outperforms state-of-the-art baselines in terms of scalability, while providing high quality results.

Organization. After reviewing the related work (Section 2) and introducing the background concepts of time series and motif discovery (Section 3), we describe our approach (Section 4). We then proceed in formalizing the details of our approach (Section 5) and we integrate optimizations to improve data adaptability and time complexity (Section 6). Finally, we compare our proposal with other baselines and we extensively test our approach under different conditions to show its effectiveness (Section 7).

2 RELATED WORK

A relatively large body of literature exists on time series and motif discovery, however the multidimensional case, despite aggregating a large interest, has seen only a fraction of the various works developed for the unidimensional case.

Comparisons between subsequences in time series are usually carried out by comparing their shapes rather than the raw values, in order to show invariance to noise and scale. Several similarity measures have been used in time series processing, they can be split in two major subgroups: *elastic* measures, that create a non linear one-to-many mapping between points of sequences, and *lock-step* measures, where the mapping is one-to-one [1]. Between the most commonly used distances is *Dynamic Time Warping* (DWT), an *elastic* measure that allows the comparison of temporally misaligned sequences due to compression or stretching of shapes (i.e., warp in time). *Edit Distances*, are a family of distances that measure the number of *edits* (e.g., substitutions, deletions, insertions) needed to obtain equal subsequences [40]. Another common measure is the *z-normalized Euclidean Distance*, a lock-step measure that z-normalizes the data before computing the Euclidean Distance, allowing variations in amplitude and mean values, so that the measured similarity is between the *shapes* [18]. This distance measure is, up to a constant factor, equivalent to the Pearson correlation coefficient [9].

Many motif discovery techniques rely on symbolic abstraction of the raw data to facilitate matching of common patterns, besides smoothing out noise in the data. *SAX* [25] found great success for its efficiency, requiring only a mean for the *Piecewise Aggregate Approximation* and a table look-up for the symbol association. It is less computationally complex than symbolization methods like *ACA* [35]. Moreover, it is more general compared to methods like *Persist* [30], which require time series with a recognizable underlying structure [35]. We refer to the survey by Daw et al. [15] for a complete overview on symbolization techniques.

Approximate algorithms for multidimensional motif discovery can be categorized in two major families: those that reduce in some way the time series into a unidimensional one and those that effectively work on multidimensional data.

Algorithms in the first category use techniques like *Principal Component Analysis* (PCA) to generate a *meta*-unidimensional time series that can be processed with the standard approaches developed for motif discovery in the univariate case. The work in [37] employs *Minimum Description Length* (MDL), to find the motifs. This approach is based on the strong assumption that all dimensions are relevant, as even a small number of noise dimensions leads to a meta-time series with little to no information, the work of [37] asserts how this algorithm *can extract a motif that can be recognized intuitively by human*, underlining how more work is needed when the structure of the time series is unknown. Moreover, the authors highlight the challenge of dynamically tuning the input parameters, since suboptimal sets can lead to poor outcomes in discovery.

The approaches that fall into the second category can be divided into two subfamilies: those who find motifs that span simultaneously in all dimensions [8] and those who find *subdimensional motifs*. The second category is the one that allows the extraction of the most amount of information, since finding motif in all dimensions falls into a similar assumption of the algorithms that synthesize an univariate meta-time series, considering also irrelevant dimensions in the process.

For the task of subdimensional motif discovery, Minnen et al. [29] introduced the *Random Projection* algorithm, which applies *SAX* to independently symbolize each dimension of the time series, then a matrix of collisions between the subsequences is populated by iteratively random selecting a set of dimensions, creating words by concatenating the selected symbols and finding the matches. The algorithm has a linear running time in expectation, but it is greatly affected by the input threshold on the distance, a data dependent variable that is difficult to have an idea of without knowing in great details the data and the kind of motif searched.

The *matrix profile* [41] is commonly used to solve this problem exactly. It is a data structure that stores the distance between a subsequence and its nearest neighbor. The first motif can be identified by searching the minimal entry in the matrix, the second is the next minimum not overlapping with the first one, and so on. When discovery is limited to motifs with a certain dimensionality, the search can be restricted to only the d-th row of the matrix. To find the set of dimensions that span the motif, a variety of techniques can be used (e.g., finding the subset with minimal distance) with MDL being the one used by the state-of-the-art implementation [23].

Very recently, a different variant of the problem has been introduced [36]: the goal is to find a set of k subsequences of the input, spanning a limited number of dimensions and minimizing the maximum pairwise distance of subsequences in the set.

Locality Sensitive Hashing (LSH) is a technique often employed in similarity search [3, 24] which we will review in the next section. Relevant for the scope of this paper is the family of hash functions for the Euclidean distance [14].

LSH has already been used in the context of time series to discover motifs: Rong et al. [34] employed LSH to derive fingerprints for earthquake waveforms. In the one dimensional case Ceccarello and Gamper [11] provided an algorithm with guarantees on recall by employing the properties of LSH. Subsequences are matched

Table 1: Table of symbols.

$\mathbf{T}$	Multidimensional time series
$\mathbf{T}^w$	Set of subsequences from $\mathbf{T}$ of length w
$\mathbf{T}_a$	Subsequence at index a
$\mathbf{T}_a^f$	Dimension f of subsequence $\mathbf{T}_a$
D	Dimensionality of the time series
d	Dimensionality of the motifs to discover
dist $(\cdot, \cdot)$	Distance function
$\text{dist}_d (\cdot, \cdot)$	d-dimensional distance function
$\text{dims}_d (\cdot, \cdot)$	dimensions involved in the computation of the d-dimensional distance function
k	Number of motifs to find
$h_{i,j} \left(\mathbf{T}_a^f \right)$	hash value of length i at repetition j for dimension f of subsequence $\mathbf{T}_a$
L	Number of repetitions for LSH
K	Number of concatenations for LSH

using their *fingerprints*, the distances are verified only on matching fingerprint pairs, allowing the algorithm to compute just a fraction of all the distance computations.

Using LSH requires setting up a number of parameters. The framework developed by PUFFIN [5] is capable of automatically tuning these parameters for k-nearest neighbors queries. Christiani et al. [13] developed a technique to automatically find the best number of concatenations and repetitions to successfully answer a nearest neighbor query with a certain probability even when the probability of collision between pairs of points is unknown. We follow the PUFFINN implementation [5], expanding on the case where the pairs are actually ordered sets.

3 PRELIMINARIES

3.1 Time Series and Motifs

We will start our introduction from the special case of a single dimension time series.

Definition 3.1. A *time series* $T \in \mathbb{R}^n$ is an ordered sequence of real valued numbers $T = [t_1, \dots t_n] : t_u \in \mathbb{R}, \forall i \in [1, n]$ where n is the length of the time series.

Definition 3.2. A *time series subsequence* $T_{u,w} \in \mathbb{R}^w$ is a subset of adjacent elements from T starting at position u and length w, $T_{u,w} = [t_u, \dots t_{u+w-1}]$.

Definition 3.3. The *z-normalized Euclidean Distance* between two time series subsequences $T_{u,w}$ and $T_{v,w}$ is defined as:

$$\text{dist} \left(T_{u,w}, T_{v,w} \right) = \sqrt{ \sum_{i \in [w]} \left(\frac{T_u(i) - \mu(T_{u,w})}{\sigma(T_{u,w})} - \frac{T_v(i) - \mu(T_{v,w})}{\sigma(T_{v,w})} \right)^2 }$$

(1)

where μ and σ are the means and the standard deviations of the subsequences, respectively, and $T_u(i)$ is the i-th value of subsequence $T_{u,w}$.

Definition 3.4. A *time series motif* is the pair of subsequences $T_{u,w}$ and $T_{v,w}$:

$$\text{dist} \left(T_{u,w}, T_{v,w} \right) \leq \text{dist} \left(T_{a,w}, T_{b,w} \right)$$

for all $a, b \in [1, \dots n - w + 1]$.

It comes that overlapping sequences are more likely to satisfy the condition above, these pairs are excluded in the problem formulation by ensuring no trivial matches.

Definition 3.5. A pair of subsequences $T_{u,w}$ and $T_{v,w}$ is a *trivial match* when: $v = u$ or if, given an exclusion zone $\epsilon > 0$, then $|u - v| \leq \epsilon$.

Common values for the exclusion zone ϵ are fractions of w to exclude matches in the neighboring area of a subsequence [42, 44]. We employ $\epsilon = w/2$.

Expanding upon the single dimensional case, the notions for the *multidimensional* case are derived.

Definition 3.6. A *multidimensional time series* $\mathbf{T} \in \mathbb{R}^{n \times D}$ is a tuple of D time series $T^{(j)} \in \mathbb{R}^n : \mathbf{T} = \left(T^{(1)}, \dots T^{(D)} \right)$ where D is the dimensionality of $\mathbf{T}$ and n is its length.

Definition 3.7. A *multidimensional subsequence* $\mathbf{T}_{u,w} \in \mathbb{R}^{w \times D}$ is a tuple of unidimensional subsequences from $\mathbf{T}$ starting at position u and length w, $\mathbf{T}_{u,w} = \left(T_{u,w}^{(1)}, \dots T_{u,w}^{(D)} \right)$.

When clear from context we will omit the subscript w from the subsequence notation.

Given two multidimensional subsequences, we are interested in computing their distance. However, as we mentioned oftentimes considering all dimensions in this computation is uninteresting at best, misleading at worst. Following [41] we therefore consider only a subset of d dimensions when computing the distance.

Definition 3.8. The d-dimensional distance between two subsequences $\mathbf{T}_a$ and $\mathbf{T}_b$ is

$$\text{dist}_d (\mathbf{T}_a, \mathbf{T}_b) = \min_{F \subseteq 2^{[D]}, |F|=d} \sum_{f \in F} \text{dist} \left(\mathbf{T}_a^f, \mathbf{T}_b^f \right)$$

In other words, we select a subset of the dimensions of size d such that the sum of the distances between the individual dimensions is minimized. Similarly, with

$$\text{dims}_d (\mathbf{T}_a, \mathbf{T}_b) = \arg\min_{F \subseteq 2^{[D]}, |F|=d} \sum_{f \in F} \text{dist} \left(\mathbf{T}_a^f, \mathbf{T}_b^f \right)$$

we denote the d dimensions that define the distance between two subsequences. Note that this set of dimensions is potentially different for different pairs of subsequences. Furthermore, for a given pair of subsequences we name the distance of the dimensions maximally far apart among the ones belonging to $\text{dims}_d (\mathbf{T}_a, \mathbf{T}_b)$:

$$\text{dist}_d^{\max} (\mathbf{T}_a, \mathbf{T}_b) = \max_{f \in \text{dims}_d(\mathbf{T}_a, \mathbf{T}_b)} \text{dist} \left(\mathbf{T}_a^f, \mathbf{T}_b^f \right) \quad (2)$$

Definition 3.9. A d-dimensional motif is the pair of subsequences $\mathbf{T}_a, \mathbf{T}_b$ such that

$$\text{dist}_d (\mathbf{T}_a, \mathbf{T}_b) \leq \text{dist}_d (\mathbf{T}_u, \mathbf{T}_v) \quad \forall u, v \in [n - w + 1]$$

We are interested in finding the most similar subsequences in an unknown subspace.

Definition 3.10. Given a D-dimensional time series $\mathbf{T}$, motif length w, motif dimensionality d, and a distance function *dist*, the *top-k multidimensional motifs* are the k subsequences and their subspaces F that minimize the d-dimensional distance with respect to all other
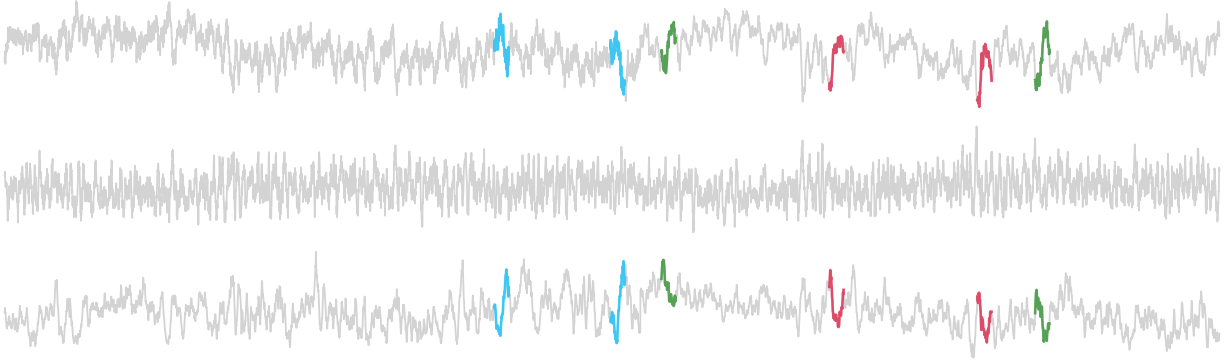
Figure 1: Multidimensional time series from an industrial evaporator [17]. The top-3 two dimensional motifs are highlighted.

subsequences of length w in $\mathbf{T}$, ensuring no trivial matches between all possible pair of indices that are part of the motif pairs (i.e., no motif overlaps with another).

Example 3.1. Figure 1 reports an example of multidimensional motif discovery: readings of different sensors from an industrial evaporator [17]. We highlight the top-3 motifs of length $w = 75$ and dimensionality $d = 2$. Notice how the middle signal does not participate in the motifs.

3.2 Locality Sensitive Hashing

A powerful technique for approximate similarity search in high-dimensional space is Locality Sensitive Hashing (LSH for short), which we briefly introduce here. Given that the subsequences of length w of a time series can be seen as vectors in $\mathbb{R}^w$, this will prove a useful tool in this setting as well. For an in-depth discussion of LSH, refer to [3, 39]. We provide, as online supplemental material, a short interactive LSH primer[1].

Intuitively, LSH partitions a set of vectors randomly in such a way that close vectors are more likely to end in the same part than far away vectors. To formalize this intuition, the definition below introduces a distance threshold R: vectors that are closer to each other than R are considered close, and vectors farther than cR are considered far away.

Definition 3.11 (Locality Sensitive Hashing [3]). Let $(\mathcal{X}, \text{dist})$ be a metric space and $\mathcal{H}$ be a family of functions $h : \mathcal{X} \to U$ for some set U. For a distance threshold R and a constant $c > 1$ the family $\mathcal{H}$ is called (R, cR, p_1, p_2)-*locality sensitive* if $\forall x, y \in \mathcal{X}$ and for h sampled from $\mathcal{H}$:

$$\begin{aligned} &\text{if dist}(x,y) \leq R \quad \text{then} \quad \Pr_{h \sim \mathcal{H}}[h(q) = h(p)] \geq p_1 \\ &\text{if dist}(x,y) \geq cR \quad \text{then} \quad \Pr_{h \sim H}[h(q) = h(p)] \leq p_2 \end{aligned} \quad . \quad (3)$$

The event of two vectors having the same hash value is called a *collision*. A key quantity to assess the performance of LSH families is

$$\rho = \frac{\log 1/p_1}{\log 1/p_2}$$

A small ρ value entails that the LSH family is effective at discerning close vectors from far-away ones.

For the common case of the Euclidean distance considered in this paper a widely used LSH family is that of *Discretized Random Projections* [14]. For a vector $x \in \mathbb{R}^w$ and quantization parameter $r \in \mathbb{R}^+$ the hash function is

$$h(x) = \left\lfloor \frac{a \cdot x + b}{r} \right\rfloor \quad (4)$$

where $a \in \mathbb{R}^w$ is a vector with random components following the $\mathcal{N}(0,1)$ Gaussian distribution, and $b \in \mathbb{R}$ is chosen uniformly at random in the interval $[0, r]$.

The probability that two vectors x, y at Euclidean distance R collide is:

$$\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] = 1 - 2 \cdot norm\left(-\frac{r}{R}\right) - \frac{2}{\sqrt{2\pi}\, r/R}\left(1 - e^{-\left(\frac{r^2}{2R^2}\right)}\right) \quad (5)$$

where *norm* is the cumulative distribution function of a *Standard normal distribution* [14]. For this family of LSH functions we have $\rho = 1/c$ [2], a fact that we will use in the analysis of the complexity of our algorithm.

As a notational shorthand, for two vectors x and y at distance $\text{dist}(x, y)$ we define

$$P(\text{dist}(x, y)) = \Pr_{h \in \mathcal{H}}[h(x) = h(y)] \ .$$

In order to *amplify* the gap between the collision probability of close vectors (at distance $\leq R$) and far vectors (at distance $\geq cR$) a common strategy is to create a *composite* hash function by sampling K hash functions and concatenating their outputs in a tuple of length K:

$$h'(x) = \langle h_1(x), h_2(x), \ldots, h_K(x) \rangle$$

The resulting LSH family is (R, cR, p_1^K, p_2^K)-locality sensitive. Using larger values of K lowers the probability for both close and far points to collide, with a more marked effect on the latter. Repeating this process with L independent composite hash functions implies that points at distance smaller than R collide in at least one of the repetitions with probability at least $1 - \left(1 - p_1^K\right)^L$.

Setting the parameters K and L requires the knowledge of the distance threshold R, which in our setting is the distance of the top-k motif. Of course we do not know this distance beforehand, therefore in the following we describe an algorithm that automatically tunes them based on the input.

Algorithm 1: MOMENTI

Input : D-dimensional time series $\mathbf{T}$, subsequence length w, number of motifs k, K max allowed number of concatenations, L max number of repetitions, dimensionality of the motifs to find d, failure probability δ

Output: {Set of the top-k motifs}, with probability $1 - \delta$

// Initialization

1 **for** $\mathbf{T}_a \in \mathbf{T}^w$, $f \leftarrow 1$ to D, $j \leftarrow 1$ to L **do**
2 $\quad$ compute $h_{K,j}(\mathbf{T}_a^f)$;

3 TOP = PriorityQueue();
4 **for** $i \leftarrow K$ to 1 **do**
5 $\quad$ **for** $j \leftarrow 1$ to L **do**
6 $\quad\quad$ $E \leftarrow \emptyset$;
7 $\quad\quad$ **for** $f \leftarrow 1$ to D **do**
8 $\quad\quad\quad$ **for** $(\mathbf{T}_a, \mathbf{T}_b) \in \mathbf{T}^w \times \mathbf{T}^w : h_{i,j}^f(\mathbf{T}_a^f) = h_{i,j}^f(\mathbf{T}_b^f)$ **do**
9 $\quad\quad\quad\quad$ **if** $(\mathbf{T}_a, \mathbf{T}_b) \notin E$ **then**
10 $\quad\quad\quad\quad\quad$ $E \leftarrow E \cup \{(a,b)\}$;
11 $\quad\quad\quad\quad\quad$ $W(a,b) \leftarrow 1$;
12 $\quad\quad\quad\quad$ **else**
13 $\quad\quad\quad\quad\quad$ $W(a,b) \leftarrow W(a,b) + 1$;
14 $\quad\quad$ **for** $(a,b) \in E$ **do**
15 $\quad\quad\quad$ **if** $W(a,b) \geq d$ **then**
16 $\quad\quad\quad\quad$ TOP.insert$((\mathbf{T}_a, \mathbf{T}_b))$;
17 $\quad\quad\quad\quad$ **if** $|\text{TOP}| > k$ **then**
18 $\quad\quad\quad\quad\quad$ TOP.pop()
19 $\quad\quad$ **if** $|\text{TOP}| = k \wedge$ STOP(TOP.max(), i, j, δ) **then**
20 $\quad\quad\quad$ **return** TOP

21 **return** true top-k by computing all pairs;

Algorithm 2: Stopping condition

1 **Function** STOP$((\mathbf{T}_a, \mathbf{T}_b), i, j, \delta)$ **is**
2 $\quad$ $p \leftarrow P\left(\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)\right)^d$;
3 $\quad$ **if** $i = K$ **then return** $(1 - p^i)^j \leq \delta$;
4 $\quad$ **else return** $(1 - p^i)^j \cdot (1 - p^{i+1})^{L-j} \leq \delta$;

4 ALGORITHM

We now describe our algorithm, named MOMENTI (for *MOtifs in MultidimEnsioNal TImeseries*), to find the top-k motifs in a multidimensional time series. Our algorithm has a user-defined error probability δ. At a high level, our algorithm is comprised of two main phases. First, it builds a LSH-based index of the time series subsequences, where the dimensions of all subsequences are hashed independently. Then the index is traversed to discover candidate motif pairs, until a data-dependent stopping condition is met. The pseudocode of our algorithm is presented in Algorithm 1.

Index construction. As we have seen in the previous section, two critical parameters in an LSH setup are the number of concatenations K and the number of repetitions L. Furthermore, the hash function of Equation (4) requires a quantization parameter r to be set as well. We shall see how to automatically set r later in Section 6.

For a time series $\mathbf{T}$, setting K and L so to minimize the number of distance computations requires the knowledge of the motif distance *before* we construct the index. To work around the fact that we, of course, do not know this distance beforehand, our index instead sets the *maximum* K and L values to be used in the second phase.

To construct the index of the subsequence of length w of a multivariate time series $\mathbf{T}$ we first sample multiple independent composite hash functions of length K: one for each dimension $f \in [D]$ and repetition $j \in [L]$. As described in the previous section, this is achieved by simply sampling a random vector a and a random value b for each function. We denote the composite hash function at repetition $j \in [L]$ for dimension $f \in [D]$ with $h_{K,j}^f$.

Then, for each subsequence $\mathbf{T}_a$, we compute multiple independent hash values: for each dimension $f \in [D]$ and for each repetition $j \in [L]$, we compute the *composite* hash value of length K of the vector $\mathbf{T}_a^f$ using the corresponding hash function, that is we compute $h_{K,j}^f(\mathbf{T}_a^f)$.

Note that a composite hash value can be seen as a string of K integer values. A fundamental operation in the next phase will be retrieving, for a given dimension $f \in [D]$ and repetition $j \in [L]$, all the subsequences whose hash share the prefix of a given length. To efficiently support this operation, we construct an index on hash values consisting of a family of ordered vectors. Specifically, for each dimension $f \in [D]$ and each repetition $j \in [L]$, each vector indexes the string of K hash values of all subsequences. By doing so, hash values with the same prefix appear in contiguous ranges of the array, providing higher locality of reference. In the following, for $0 < i \leq K$ we denote with $h_{i,j}^f(\mathbf{T}_a^f)$ the prefix of length i of the hash value for the subsequence dimension $\mathbf{T}_a^f$ in repetition j.

Index traversal. A key property of the index introduced in the previous paragraph is that very similar subsequences are likely to share long prefixes of their hashes. At the same time, due to the probabilistic nature of LSH it is not certain that similar subsequences share a long prefix in the first repetition.

We first give the intuition behind the algorithm, and then give all the details. Starting from the longest possible hash prefix, K, the L repetitions are considered, focusing on each one on all pairs of subsequences sharing the same hash prefix: for each such pair we can compute the distance, which is used to rank candidate motif pairs in a priority queue. Furthermore, from the distance we can derive the collision probability by means of Equation (5). We use this probability to define a stopping condition that allows to rule out the event that the true motif has *not* been seen in the repetitions considered so far. If the probability of this negative event is less than a user defined threshold δ and the priority queue contains at least k elements then the algorithm stops, returning the top-k pairs in the priority queue. Otherwise the next iteration is considered, with a caveat: if the last of the L repetitions is reached with the stopping condition not satisfied, then it means that hashes of length K are too selective for the dataset at hand. Therefore, the process is restarted considering prefixes of length $K - 1$. This procedure continues, potentially considering shorter and shorter prefixes, until the stopping condition is met.

The above high level intuition is complicated by the fact that we have to deal with multidimensional subsequences whose dimensions are hashed independently. In the following we thus detail the algorithm with reference to the pseudocode in Algorithm 1.

The algorithm proceeds in rounds from K to 1: in round i hash prefixes of length i are considered. In each round i then all repetitions $1 \le j \le L$ are considered, and each such iteration is comprised of three steps: counting the number of collisions, computing distances, and checking the stopping condition.

The first step (line 6 to 13) counts for each pair of subsequences the number of dimensions in which they share a prefix of length i. Note that retrieving the pairs of subsequences colliding in each dimension (line 8) is done efficiently by leveraging the fact that hash values are stored in lexicographically sorted arrays. By the end of the first step the algorithm has built a set E of pairs of subsequences that collided in at least one dimension, and for each pair $(a, b) \in E$ the function $W(a, b)$ reports the number of dimensions on which $\mathbf{T}_a$ and $\mathbf{T}_b$ collided.

In the second step the focus (lines from 14 to 18) is on the pairs that share prefixes in at least d dimensions out of D, where d is the target number of dimensions spanned by the motifs. The intuition is that a motif pair $(\mathbf{T}_a, \mathbf{T}_b)$, being similar in at least d dimensions, will have $W(a, b) \ge d$. For each such pair, the algorithm computes the distance and updates the priority queue of candidates, keeping only the top-k in memory.

Finally, if the priority queue contains k candidates then the algorithm checks the stopping condition (Algorithm 2), which considers the pair at maximum distance in the priority queue. In particular, for this pair $(\mathbf{T}_a, \mathbf{T}_b)$ the stopping condition focuses on $\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)$ as defined in Equation (2). This distance is used to compute an upper bound p to the probability of the two subsequences colliding in d dimensions at the same time independently (line 2). This probability p is then used to compute the probability that a pair of subsequences with a smaller pairwise distance was missed by the algorithm in all the previous iterations. If this probability is smaller than the user-defined δ, then the algorithm can successfully terminate. In particular, line 3 is evaluated when the full hashes are being considered, and only the first j repetition have been executed. Line 4 is executed when prefixes shorter than K are under consideration, and takes into account the fact that the algorithm executed $L - j$ iterations with prefixes of length $i + 1$.

Note that the iterative process of lines 4 to 15 inherently exhibits *anytime* properties, as the priority queue only allows the insertion of better solutions. The user can stop at any time the discovery to retrieve the candidate motifs with their error probabilities.

Example 4.1. Consider the time series shown in Figure 1, let $\mathbf{T}_a, \mathbf{T}_b$ be the names of the subsequences belonging to the first motif (in red) and $\mathbf{T}_c, \mathbf{T}_d$ to the second (in blue), TOP be the queue where we store the top motif with $k = 1$, $K = 4$, $L = 2$ and $\delta = 0.1$. Let us assume that there exist no other collisions other than the two motifs. During the iteration at $i = 4$, $j = 1$ we will scan the hashes for each dimension and update our weights (lines 7-13), we find $W(\mathbf{T}_a, \mathbf{T}_b) = W(\mathbf{T}_c, \mathbf{T}_d) = 2$. The first couple will be stored in the queue since it has a lower distance (lines 14-18). At this point (line 19), $\text{STOP}(\text{dist}_d(\mathbf{T}_a, \mathbf{T}_b), 4, 1) \ge \delta$ so the condition is not satisfied. This happens again for $j = 2$, when we find $(\mathbf{T}_c, \mathbf{T}_d)$ we evaluate

the distance again because we have no memory of the past, and discard it for its distance is greater than the couple stored in TOP. Since the hashes of length $i = 4$ were unable to satisfy the stopping condition, we move to prefix length $i = 3$. This time, for both $j = 1$ and $j = 2$, we find $(\mathbf{T}_a, \mathbf{T}_b)$ and $(\mathbf{T}_c, \mathbf{T}_d)$ with weight 2, we will compare them again even though they already have been evaluated during the previous step, we will later implement a way to avoid this operation and save resources. At the end of repetition $j = 2$ we find that the bound on the failure probability of the pair stored in the queue is satisfied and the algorithm stops.

5 ANALYSIS

In this section, we derive the probabilistic guarantees for our algorithm as well as analyze its complexity.

LEMMA 5.1. *Given a pair of subsequences $\mathbf{T}_a, \mathbf{T}_b$ and parameter d_m, consider iteration i of the outer loop of Algorithm 1. Then, we have $W(a, b) \ge d$ with probability at least $P\left(\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)\right)^{i \cdot d}$ where $\text{dist}_d^{\max}(\cdot, \cdot)$ is defined as in Equation (2).*

PROOF. To have $W(a, b) \ge d$ we need the two subsequences to collide in at least d dimensions.
Consider the distance $\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)$ and recall that, by definition, all dimensions $F = \text{dims}_d(\mathbf{T}_a, \mathbf{T}_b)$ will be at a closer distance.

At iteration i of the outer loop of Algorithm 1 we have that the two subsequences collide in all dimensions F at the same time with probability at least $\prod_{f \in F} P\left(\text{dist}\left(\mathbf{T}_a^f, \mathbf{T}_b^f\right)\right)^i$ given that $\bar{f}$ is the dimension of maximum distance out of the d ones in F, each factor of the above product is lower bounded by $P\left(\mathbf{T}_a^{\bar{f}}, \mathbf{T}_b^{\bar{f}}\right)$, hence the statement follows. ☐

5.1 Correctness

To prove correctness we need to show that motif pairs are considered at least once before the algorithm terminates.

The following lemma bounds the probability that a given pair of subsequences at indices a and b never has a weight $W(a, b) \ge d$. In other words, the following lemma bounds the probability that a pair is never considered for inclusion in the TOP priority queue.

LEMMA 5.2. *Let $(\mathbf{T}_a, \mathbf{T}_b)$ be a pair of subsequences and let*

$$p = P\left(\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)\right)^d$$

Consider iteration i out of K of the outer loop and iteration j out of L of the inner loop of Algorithm 1. The probability that $W(a, b) < d$ in all previous iterations is upper bounded by

$$
\begin{cases}
(1 - p^i)^j & \text{if } i = K \\
(1 - p^i)^j \cdot (1 - p^{i+1})^{L-j} & \text{otherwise}
\end{cases}
\tag{6}
$$

PROOF. Sketch of the proof: from Lemma 5.1 we know the probability of the pair to collide in a single repetition with prefixes of length i. For the given pair the failure event is to have a weight $W(a, b) < d$. Therefore, the probability of never colliding over j independent repetitions is $(1 - p^i)^j$. In the case where $i = K$, the statement follows. When $i < K$, we consider that the first j iterations performed with prefix i fail independently and that the $L - j$ iterations previously run with prefix $i + 1$ failed as well. ☐

From this we now derive two lemmas on the success probability of the discovery. In Lemma 5.3 we allow each motif to fail independently, in Lemma 5.4 we require that all the returned motifs are correct within a probability.

LEMMA 5.3. *Algorithm 1 finds the true top-k motifs, each with probability $\geq 1 - \delta$.*

PROOF. Let the stopping condition be met at iteration i' and concatenation j', $m_1, \ldots m_k$ be the set of motifs returned by the algorithm, sorted by increasing distances. We have that the failure probability of m_h, $h \in [1, k]$ is upper bounded by the failure probability m_k for the monotonicity of the collision probability. This failure probability is given by Lemma 5.2, by construction the stopping condition ensures that this probability is $\leq \delta$. This is valid for all returned pairs independently.

If the stopping condition is never met the algorithm reaches line 20, where all pairs of subsequences are considered, thus returning the correct motifs with probability 1. □

Given this lemma it is easy to derive the expected recall of the algorithm, that corresponds to $1 - \delta$.

LEMMA 5.4. *When called with failure probability $\delta' = \delta/k$, Algorithm 1 finds the true top-k motifs with probability $\geq 1 - \delta$.*

PROOF. From Lemma 5.3 we have that each pair fails independently with probability $\leq \delta' = \delta/k$. By applying a union bound on the k pairs we obtain the statement. □

5.2 Number of Distance Computations

First, we introduce the concept of *contrast*, which will be useful in capturing the difficulty of a dataset and in relating it to the complexity of our algorithm.

Definition 5.1. For a D-dimensional time series $\mathbf{T}$ of length n, and for parameters k and d let $(\mathbf{T}_{a_k}, \mathbf{T}_{b_k})$ and $(\mathbf{T}_{a_n}, \mathbf{T}_{b_n})$ be the k-th and n-th motifs, respectively. We define

$$\text{contrast}_{d,k|n}(\mathbf{T}) = \frac{\text{dist}_d^{\max}(\mathbf{T}_{a_n}, \mathbf{T}_{b_n})}{\text{dist}_d^{\max}(\mathbf{T}_{a_k}, \mathbf{T}_{b_k})}$$

The following theorem ensures that our algorithm computes, in expectation, a subquadratic number of distances in expectation, assuming we give to the algorithm enough memory. The complexity is parameterized by the contrast of the motifs in the multivariate time series: a large contrast implies a smaller complexity.

THEOREM 5.5. *On a D-dimensional time series $\mathbf{T}$ of length n, with parameters $k \geq 1$, $d \in [1, D]$, $\delta \in (0, 1)$, Algorithm 1 computes*

$$O\left(n^{1+1/c} \log \frac{1}{\delta} + Lk\right)$$

distances in expectation, where $c = \text{contrast}_{d,k|n}(\mathbf{T})$, for $L \in \Omega\left(n^{1/c}\right)$.

PROOF. Let $\mathbb{T}$ be the set of all the subsequence pairs of $\mathbf{T}$. For any pair of subsequence indices (a, b), let $p_{ab} = P\left(\text{dist}_d^{\max}(\mathbf{T}_a, \mathbf{T}_b)\right)^d$.

Consider $(\mathbf{T}_{a_k}, \mathbf{T}_{b_k})$ and $(\mathbf{T}_{a_n}, \mathbf{T}_{b_n})$, the k-th and n-th d-dimensional motifs of $\mathbf{T}$, and let $p_1 = p_{a_k b_k}$ and $p_2 = p_{a_n b_n}$. Furthermore, let $\rho = \frac{\log 1/p_1}{\log 1/p_2}$.

For notational convenience, define the following operator that gives the expected number of distance computations carried out in j independent repetitions with hash prefixes of length i:

$$\text{ED}(i, j) = j \cdot \left(\sum_{(\mathbf{T}_a, \mathbf{T}_b) \in \mathbb{T}} p_{ab}^i\right)$$

Recall that, for hash prefixes of length i, examining $\frac{\log 1/\delta}{p_1^i}$ independent repetitions ensures that the top k motif pairs are seen at least once with probability at least $1 - \delta$. Using the notation shorthand defined above, the number of expected distance computations in this case is $\text{ED}\left(i, \frac{\log 1/\delta}{p_1^i}\right)$.

Now define $\mathbb{T}_{>k}$ and $\mathbb{T}_{>n}$ as the sets of subsequence pairs that are farther than the k-th and n-th motifs, respectively. Then we have

$$
\begin{aligned}
\text{ED}\left(i, \frac{\log 1/\delta}{p_1^i}\right) &\leq \frac{\log 1/\delta}{p_1^i} \cdot \left(n + \sum_{(\mathbf{T}_a, \mathbf{T}_b) \in \mathbb{T}_{>n}} p_{ab}^i\right) \\
&\leq \frac{\log 1/\delta}{p_1^i} \cdot \left(n + \binom{n}{2} p_2^i\right)
\end{aligned}
\tag{7}
$$

where the inequality follows from the definition of $\mathbb{T}_{>n}$.

Defining $i^* = \frac{\log n}{\log \frac{1}{p_2}}$ we have $\binom{n}{2} p_2^{i^*} \leq n$ and $\frac{1}{p_1^{i^*}} = n^{\frac{\log 1/p_1}{\log 1/p_2}} = n^\rho$.

Therefore there is a prefix length i^* for which the number of expected distance computations to see the k-th motif colliding at least once is

$$\text{ED}\left(i^*, \frac{\log 1/\delta}{p_1^{i^*}}\right) = O\left(n^{1+\rho} \log \frac{1}{\delta}\right) = O\left(n^{1+\frac{1}{c}} \log \frac{1}{\delta}\right) \tag{8}$$

where $c = \text{contrast}_{d,k|n}(\mathbf{T})$ and the equality follows from the definition of ρ for the LSH family we employ in our algorithm.

Now, let $i' \geq i^*$ be the largest prefix i such that the stopping condition holds. With probability $1 - \delta$ the algorithm stops at prefix i'. Conditioned on this event, the number of distance computations carried out at level i' is

$$\text{ED}\left(i', \frac{\log 1/\delta}{p_1^{i'}}\right) + \text{ED}\left(i' + 1, L - \frac{\log 1/\delta}{p_1^{i'}}\right) \tag{9}$$

Where the second term accounts for the repetitions considered in iteration $i' + 1$ of the outer loop of the algorithm. First we bound the first term of the addition. Similarly to before, define $\mathbb{T}_{>k}$ as the set of subsequence pairs that are farther away than the k-th motif.

$$
\begin{aligned}
\text{ED}\left(i', \frac{\log 1/\delta}{p_1^{i'}}\right) &\leq Lk + \frac{\log 1/\delta}{p_1^{i'}} \left(\sum_{(\mathbf{T}_a, \mathbf{T}_b) \in \mathbb{T}_{>k}} p_{ab}^{i'}\right) \\
&\overset{(a)}{\leq} Lk + \frac{\log 1/\delta}{p_1^{i^*}} \left(\sum_{(\mathbf{T}_a, \mathbf{T}_b) \in \mathbb{T}_{>k}} p_{ab}^{i^*}\right) \\
&\leq Lk + \text{ED}\left(i^*, \frac{\log 1/\delta}{p_1^{i^*}}\right) \overset{(b)}{=} O\left(Lk + n^{1+\frac{1}{c}} \log \frac{1}{\delta}\right)
\end{aligned}
$$

where (a) follows from the fact that $p_1 \geq p_{ab}$ and (b) follows from Equation (8). The theorem follows by observing that the second term of Equation (9) is a factor $1/p_1 = O(1)$ larger than the first term. □

5.3 Index construction and size

We now consider the contribution to the running time of Algorithm 1 given by the hash index and derive its space complexity.

LEMMA 5.6. *The hash construction at line 1 of Algorithm 1 takes time* $O(D \cdot K \cdot L \cdot n \log n)$.

PROOF. For each multidimensional subsequence we have to evaluate $D \cdot K \cdot L$ hashes. For a fixed hash function we can compute all the dot products in $O(n \log n)$ time, using the cyclical convolution theorem. The result follows. □

Given that for each of the n subsequences we have to store, in each of the L repetitions, a total of D hashes of length K we have the following.

THEOREM 5.7. *Algorithm 1 has space complexity proportional to* $O(K \cdot L \cdot D \cdot n)$.

6 IMPLEMENTATION DETAILS

In this section, we describe the key aspects of the implementation that impact the algorithm's running time and its motif discovery capabilities. Namely, the number of hash evaluations and comparisons and the challenge of not knowing the dimensionality of the motifs to discover.

6.1 Index building

Setting the quantization width r. Recall from Equation 4 the formulation of the hash function we use. Even though the choice of the parameter r does not compromise the correctness of the algorithm, it is fundamental since it influences the performance. A value that is too high will group many subsequences together at long indices, mitigating the filtering effect of LSH, a value that is too low will separate even the smallest perturbation between subsequences, forcing the algorithm to visit shorter prefixes. To deal with this parameter automatically, the algorithm adopts an estimation-based heuristic.

First we sample a number of random vectors from $\sim \mathcal{N}(0, 1)$, computing their dot product with a random sample of subsequences from the time series. This produces an estimate of the distribution of the values that are discretized by Equation 4.

Then, we discretize this empirical distribution into 256 equal-width buckets. The width of these buckets will be used as the parameter r in the hash function. The rationale is that by doing so we will be able to represent each hash value with a single byte.

Note that this heuristic can be implemented efficiently by leveraging on the *cyclical convolution theorem*, the same method used by MASS for the *Distance Profile* [43]. We can obtain, with one convolution between one of the random vectors and the sample of the time series, all the dot products for that vector.

Tensoring. Tensoring is a technique to reduce the number of evaluations for the hash functions [12]. Let $\mathcal{H}$ be a LSH family and $K, L \geq 1$ integers. For $m = \sqrt{L}$, we define $\mathcal{H}_l$ as a set of m hash functions sampled from $\mathcal{H}^{\frac{K}{2}}$ and similarly for $\mathcal{H}_r$. Then $(h_a, h_b) \in \mathcal{H}_l \times \mathcal{H}_r, 1 \leq a, b \leq m$ provides m^2 repetitions with Km evaluations. Furthermore, let us define for $1 \leq j \leq L$:

$$h_{K,j} = \left(h_{\frac{K}{2},l}, h_{\frac{K}{2},r} \right) \in \mathcal{H}_l \times \mathcal{H}_r \text{ where } \begin{cases} l = j \div \sqrt{L} \\ r = j \mod \sqrt{L} \end{cases} \quad . \quad (10)$$

The resulting hash is obtained from interleaving values from the selected left and right hash. This reduces the hash evaluations from $K \cdot L$ to $K \cdot \sqrt{L}$ at the cost of losing independence between repetitions. To accommodate this change we give this alternative formulation of Lemma 5.2, omitting the proof for the sake of space.

LEMMA 6.1. *Under the same conditions of Lemma 5.2, using the tensoring approach, the probability that $W(a, b) < d$ in all previous iterations is upper bounded by*

$$\begin{cases} P_t(i/2, j) & \text{if } i = K \\ P_t(i/2, j) \cdot P_t((i+1)/2, L - j) & \text{otherwise} \end{cases}$$

where $P_t(i', j') = (1 - p^{\lceil i' \rceil})^{j' \div \sqrt{L}} \cdot (1 - p^{\lfloor i' \rfloor})^{j' \mod \sqrt{L}}$.

Using Lemma 6.1 in the stopping condition of Algorithm 1 maintains the correctness guarantees of Lemmas 5.3 and 5.4.

Corollary 5.6.1. With the tensoring approach the index construction takes time proportional to $O(D \cdot K \cdot \sqrt{L} \cdot n \log n)$.

6.2 Index traversal

Comparisons on the fly. In Algorithm 1, maintaining the structure of line 4 at runtime is very expensive, since it would require quadratic space to store the weights for, potentially, each pair of subsequences. To deal with this problem, while maintaining the same theoretical approach, we just scan the index over each dimension and every time a unidimensional collision is seen we immediately compute $W(a, b)$, if greater or equal than the searched motif dimensionality d, we perform the insertion of the pair in the priority queue. This approach does not require any additional space at the cost of a slightly higher number of hash comparisons, since we may evaluate a colliding pair up to D times.

Duplicate collisions within the same repetition. Our approach dynamically chooses the length of the composite hashes by progressively iterating on the prefixes of the full ones, this results in looking into prefixes of decreasing size. Consequently, collisions at level i are a *superset* of the collisions at level $i + 1$, in order to avoid unnecessary distance computations, the algorithm will check if the colliding hashes appear at the level $i + 1$, skipping the pair in the positive case.

6.3 Finding motifs of multiple dimensionalities

One of the critical points up to now is the fact that at input we require the number of dimensions d that span the motif. In many real scenarios, it is possible to know only approximately the expected dimensionality of the pattern in a certain domain.

Our method can be easily extended to discover motifs spanning different dimensions, allowing the user to specify a range of dimensionalities d_{low}, d_{high} of the motifs they want to discover. The data structure at line 3 of Algorithm 1 becomes a set of $d_{high} - d_{low} + 1$ priority queues, which are independently updated during the discovery process. This allows the algorithm to reuse information from a single distance computation across the different requested dimensionalities. Since evaluating $\text{dist}_d(\mathbf{T}_a, \mathbf{T}_b)$ for a pair of subsequences requires computing $\text{dist}\left(\mathbf{T}_a^f, \mathbf{T}_b^f\right) \quad \forall f \in [D]$, we can

Table 2: Information about the evaluation datasets.

dataset	n	D	window	d	$c_{d,1\|n}$
potentials	2 500	8	50	8	6.11
evaporator	7 000	5	75	2	2.60
RUTH	14 859	32	500	4	3.24
weather	100 057	8	5000	2	2.43
whales	450 001	32	300	6	1.22*
quake	6 440 998	32	100	4	1.65*
electrical_load	6 960 008	10	1000	5	1.91*
LTMM	25 132 289	6	200	3	5.31*

* obtained through random sampling.

efficiently derive $\text{dist}_d\left(\mathbf{T}_a, \mathbf{T}_b\right) \quad \forall d \in [1, D]$. The condition of line 15 is initially based on d_{low} and gradually increases every time the lowest dimensionality has its motif confirmed. This approach allows to return solutions for each motif dimensionality as soon as their error probabilities satisfy the required quality, inheriting the anytime property.

7 EXPERIMENTS

This section aims at answering the following questions:

- How does our algorithm compare with the state-of-the-art?
- How do parameters K, L, r and δ influence the performance?
- How does the algorithm scale with respect to the input size?
- Is the algorithm able to find motifs in high dimensional noisy time series?

Baselines. Our algorithm is compared to Mstump [23], the state-of-the-art implementation for the multidimensional matrix profile. We stress that Mstump is an *exact* algorithm that computes more information than just the motifs: for instance it can be used to detect discords. Our aim is to investigate the gains that can be attained when only motifs are sought, and when a small failure probability is accepted. Alongside, we consider the Extended Motif Discovery (EMD) algorithm [37], reimplemented in [22], and the Random Projection (RP) algorithm [29], reimplemented by us, for comparisons with approximate approaches. We refer to our algorithm as MOMENTI, which stands for MOtifs in MultidimEnsioNal TImeseries. All algorithms were subject to a global timeout of 4 hours per execution for datasets under the million points, and 24 hours for larger datasets.

Experimental Setup. The evaluation was carried out on a machine with a 8 core Intel Xeon W-2245 @ 3.90 GHz equipped with 128 GB of memory.

Datasets. All the experiments are run on the following real datasets from different domains, whose details can be found in Table 2.

- Potentials is a record of skin potentials [16];
- Evaporator is data from a four-stage evaporator to reduce the water content from products [17];
- Ruth, the Mel-spectrogram of the song *Running Up That Hill* by Kate Bush, extracted with the following parameters: 32-Mel scale filters, 46 milliseconds short time Fourier transform window and 23 milliseconds hop;

Table 3: Time required to find the d-dimensional top motif, for fixed d, averaged over 9 runs. Values in parentheses are estimates.

dataset	MOMENTI		MSTUMP	EMD	RP
	Index build	Total			
potentials	0.11	0.51	3.65	4.80	3.20
evaporator	0.16	0.55	4.45	12.95	6.78
RUTH	2.91	8.10	84.04	1.5h	2.3h
weather	15.04	33.37	1035.73	-	1.2h
whales	60.67	2.2 h	(2.7 days)	-	-
quake	175.3	3.6 h	(7.2 days)	-	-
el_load	180.2	2.8 h	(8.4 days)	-	-
LTMM	240.6	15.6 h	(11.8 days)	-	-

- Weather is representing the hourly climate data near Monash University, in Australia, for about 10 years [19];
- Whales is obtained from data of an underwater passive acoustic network and is a 10 minute recording of humpback whales vocalizations [32];
- Quake is a waveform from the Observatories and Research Facilities for European Seismology (ORFEUS) during the 2014 Aegean Sea earthquake [28];
- El_load includes cleaned electrical consumption data in Watts for 20 households at aggregate and appliance level [31];
- LTMM contains 3-day 3D accelerometer recordings of elder community residents, used to study gait, stability, and fall risk [20, 21].

The choice of the reference motif dimensionality d for each dataset is guided by the additional information available. Specifically, for Potentials, Evaporator and Weather we use metadata information for the expected dimensionality of the pattern. For Ruth and Whales, we follow previous work on audio data to find, respectively, the drum pattern of the song and whale harmonization [41]. Finally, El_load, Quake and Ltmm were chosen based on the domain information.

Furthermore, we characterize each dataset by the *contrast* $c_{d,1\|n}$ of its top d-dimensional motif, as per Definition 5.1. Datasets with a small contrast are expected to be more difficult, i.e. require more time to find the top motif. For instance El_load is expected to be more difficult compared to Weather, since the latter has a pair of subsequences that repeat the same shape at a distance that is significantly smaller than the n-th closest pair.

Default parameter values. MOMENTI will have its parameters set to default unless otherwise indicated. The failure probability is set at $\delta = 0.01$, the maximum hash length is set to $K = 8$, while the maximum number of repetitions is set to $L = 200$. r is automatically estimated using the heuristic introduced in Section 6.1.

7.1 Finding the top motif

In this first experiment, the task is to find the top motif at a given dimensionality d (as per Table 2). For each dataset we report the total running time of the baselines Mstump, EMD and RP in the last three columns, whereas for our approach we report both the total time and the time required to set up the index. All times are in

Table 4: Number of cumulative distance computations to find the top d-dimensional motif.

dataset	MOMENTI	MSTUMP	EMD	RP
potentials	64	$2.4 \cdot 10^7$	$1.0 \cdot 10^2$	$2.7 \cdot 10^3$
evaporator	$4.7 \cdot 10^2$	$1.4 \cdot 10^8$	5	$4.4 \cdot 10^4$
RUTH	$1.2 \cdot 10^2$	$3.3 \cdot 10^9$	$5.4 \cdot 10^3$	$2.3 \cdot 10^5$
weather	$9.8 \cdot 10^4$	$3.9 \cdot 10^{10}$	-	$2.1 \cdot 10^6$
whales	$3.1 \cdot 10^8$	$(3.2 \cdot 10^{12})$	-	-
quake	$1.1 \cdot 10^6$	$(6.6 \cdot 10^{14})$	-	-
el_load	$4.4 \cdot 10^2$	$(2.4 \cdot 10^{14})$	-	-
LTMM	$4.4 \cdot 10^2$	$(1.9 \cdot 10^{15})$	-	-

seconds, unless otherwise noted. Furthermore, for timed out runs of Mstump we report, in parentheses, an estimate of the running time, which is made possible by the very regular behavior of Mstump with respect to the input size. For EDM and RP the running time is more unpredictable, therefore we refrain from providing estimates for timed out runs.

As can be seen, our algorithm is faster on all settings, completing the execution orders of magnitude faster than the baselines on the larger datasets. To substantiate this observation, Table 4 reports the number of distance computations carried out by each approach: MOMENTI usage of LSH allows to effectively prune most distance computations, whereas Mstump computes a quadratic number of distances.

The EMD algorithm appears to be the slowest, its major drawback being the set up of the parameters for a successful discovery. On the other hand, RP scales better to larger datasets but its parameter tuning heuristic repeatedly discards the collision matrix when a sparsification target is not met, this is especially costly on high dimensional datasets. We stress that the Matrix Profile computed by Mstump allows the discovery of motifs of any dimensionality, while in this test MOMENTI only finds the top motif for a fixed dimensionality. In the next section we will investigate how MOMENTI behaves when finding motifs for all dimensionalities at the same time.

Finally, Table 5 reports the memory usage, in gigabytes, of the different algorithms. We observe that MOMENTI requires the least memory to execute.

Quality of the motifs. To better understand what pairs are returned by the different algorithms we measured the Mean Absolute Relative Error (MARE) of the distances of the discovered motifs with respect to the exact ones. We found that RP can reach MARE values as high as 40%, while EMD stays below 20%. In contrast, our method consistently maintains MARE under 3%, underlining how beneficial LSH, that mainly explores close pairs, is in this context. Further details on the quality of our results will be given in Section 7.5.

7.2 Finding the top motifs with different dimensionalities

In this second experiment the aim is to find the motifs for all dimensionalities $2 \leq d \leq D$ in the same execution. MSTUMP does this natively, whereas our algorithm MOMENTI can be adapted to do so as described in Section 6.3. EMD and RP can only achieve

Table 5: Memory required to find the top d-dimensional motif.

dataset	Space (GB)			
	MOMENTI	MSTUMP	EMD	RP
potentials	0.016	0.028	0.79	0.07
evaporator	0.020	0.030	0.81	0.1
RUTH	0.025	0.084	1.19	0.5
weather	0.027	0.106	-	0.8
whales	0.24	3.2	-	-
quake	2.13	23.0	-	-
el_load	1.50	7.4	-	-
LTMM	3.00	15.7	-	-

Table 6: Time required for the top motifs for all dimensionalities. Mean over 9 runs. Values in parentheses are estimates.

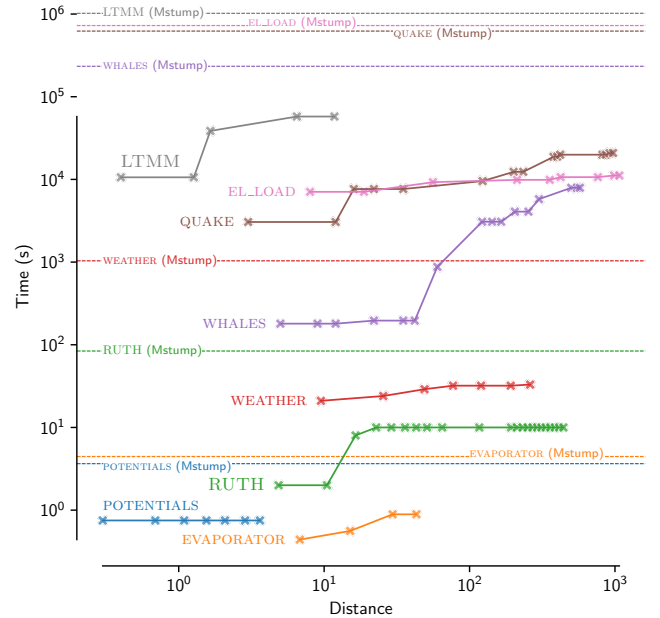| dataset	MOMENTI	MSTUMP
potentials	0.75	3.65
evaporator	0.89	4.45
RUTH	12.46	84.04
weather	47.85	1035.73
whales	2.5 h	(2.7 days)
quake	5.8 h	(7.2 days)
el_load	3.1 h	(8.4 days)
LTMM	16 h	(11.8 days)

Figure 2: Solid lines mark the time required by MOMENTI to find the top motif of each dimensionality for all datasets; dashed lines mark the time required by the Mstump baseline for the same task.

this through multiple executions and are thus excluded from this experiment. In Table 6 we report the results for this experiment.
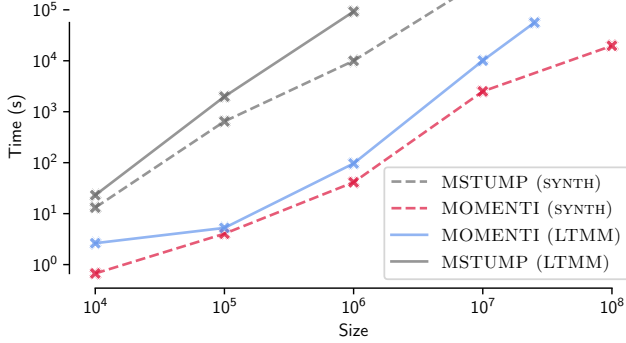
Figure 3: Scalability vs. input size (log scale).

Figure 4: Time and space requirements for motif discovery at different maximum allowed values of K.

Note that the running times for MSTUMP are identical to the ones reported in Table 3, given that MSTUMP finds motifs for all dimensionalities natively. As for MOMENTI the indexing time is the same in this setting as the one reported in Table 3, therefore we omit it for clarity. We observe that even in this more challenging scenario our algorithm is able to discover the motifs for all dimensionalities faster than the MSTUMP baseline. Indeed, the running time of MO-MENTI is always within a factor ≈ 1.6 of its running time on the one dimensional case.

To further investigate this behavior, in Figure 2 we report the relation between the discovery time of a motif (x axis) and its distance (y axis). Each color identifies a dataset, and each point represents a motif for some dimensionality: for instance the WEATHER dataset (orange) has 7 points in this figure because it has 8 dimensions, and we set the algorithms to find the 7 motifs spanning between 2 and 8 dimensions. The dashed horizontal lines mark the runtime performance of MSTUMP, which reports all the motifs at the same time at the end of its execution. For instance, the orange dashed horizontal line reports that MSTUMP takes $\approx 10^3$ seconds on the WEATHER dataset. The times marked by dashed lines and by the rightmost points of each solid line are the same as in Table 6.

As can be observed in Figure 2, by virtue of how the algorithm runs, motifs at a shorter distance are found earlier. For instance, for WEATHER the motif of dimensionality 4 has a $\text{dist}_4 \simeq 49$ while the motif of dimensionality 5 has $\text{dist}_5 \simeq 77$, which requires more iterations (and thus more time) to meet the stopping condition of the algorithm. As we discussed earlier, this fact can be used in an *anytime* fashion: the execution can be stopped at any point after the first motifs have been returned, knowing that motifs yet to be found are at larger distances and thus might be uninteresting.

7.3 Scalability

We now test the scalability of MOMENTI compared to MSTUMP. We omit EMD and RP from this comparison, as previous sections show that they do not scale beyond moderately sized time series. While the behavior of MSTUMP is data-independent, i.e. its running time depends only on the size of the data, for MOMENTI the running time is influenced by both the size of the time series and the contrast of the motifs. Therefore, to test the scalability in a robust way we employ our largest dataset and a synthetic one. We generate a random walk with $D = 5$ of length between 10000 and 100
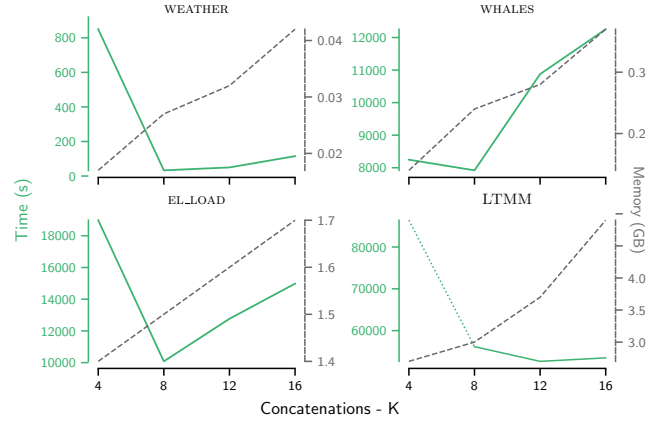
million, planting a sinusoidal motif of length $w = 300$ across $d = 2$ dimensions perturbed with Gaussian noise such that $c_{2,1|n}$ is 1.1. For LTMM we pick chunks of increasing size centered around the true motif, we underline that each chunk has a different contrast, so some could be harder than others for our algorithm. The results are reported in Figure 3, where both axes use a logarithmic scale. We observe that our algorithm scales better with the input size, with a sub-quadratic running time.

7.4 Influence of parameters on the running time

We now study the impact of the parameters K, L, and r on the running time, that in the previous experiments were either fixed ($K = 8$ and $L = 200$) or estimated from the data (the quantization parameter r). Remember that our algorithm finds the correct answer with probability $1 - \delta$ for any setting of the parameters, that influence only the performance. Each experiment in the following has been repeated 10 times: the plots report the average as a colored line, additionally, a gray dashed line reports the sizes of the hash indices built under different settings. Dotted lines indicate that the result is estimated due to timeout.

Impact of concatenations K. We test $K \in \{4, 8, 12, 16\}$, reporting the results in Figure 4. We observe that a higher number of concatenations is related to longer computing times, both because more repetition of the outer loop of Algorithm 1 have to be executed and because the index construction takes longer. Conversely, using short hash values with $K = 4$ incurs high execution times as well, mainly because fewer distance computations are pruned. In all tested cases the best tradeoff is achieved with $K = 8$, which is the recommended value and the one we used in all previous experiments. We stress that for any value of K considered in this section MOMENTI is faster than the baseline MSTUMP.

Impact of repetitions L. We test L in the range from 10 to 400, setting $K = 8$: a large L will allow the stopping condition to be satisfied in earlier iteration of the outer loop of Algorithm 1, thus requiring fewer distance computations, at the expense of a higher index build time. Figure 5 explores this trade-off, with values of L between 100 and 200 minimizing the search time. Adding more repetitions increases the memory usage, which however remains rather
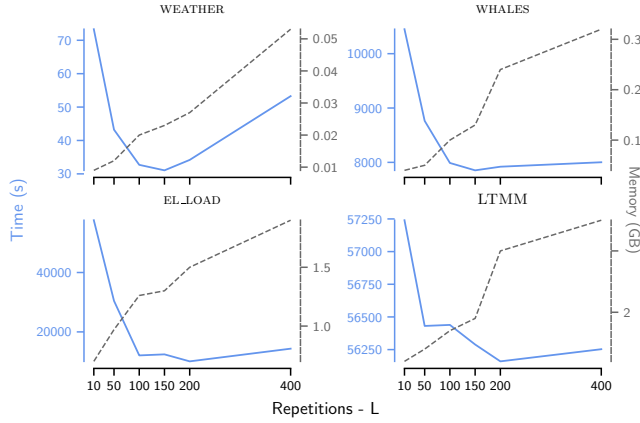
Figure 5: Time and space requirements for motif discovery at different maximum allowed values of L.
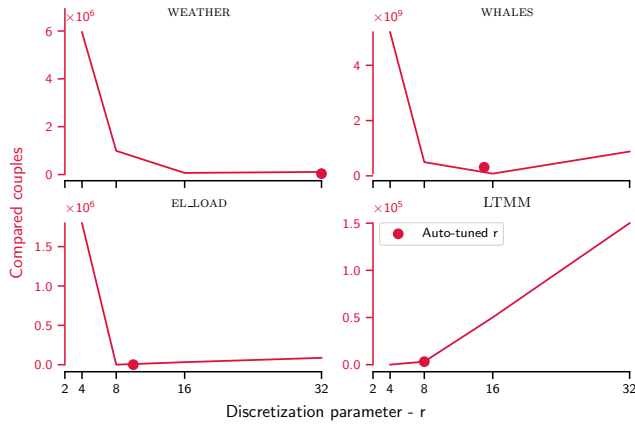

Figure 6: Number of comparisons for different values of r. The red dot is the value found by the heuristic in Section 6.1.
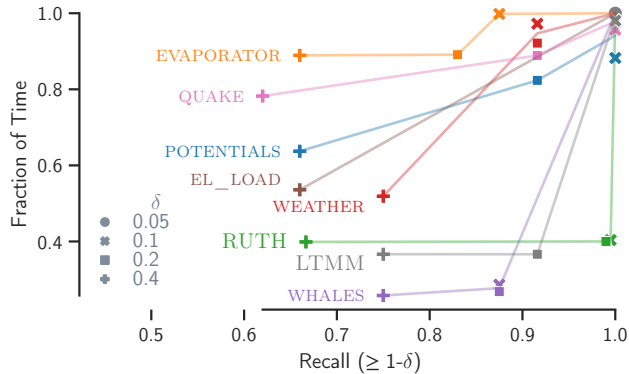

Figure 7: Relation between δ, represented by the markers, the measured recall and the fraction of time required with respect to the tighter bound.

moderate, with 3.5Gb being used for the largest dataset, LTMM. Datasets that benefit more from an increased number of repetitions are those whose relative contrast is higher (e.g. LTMM), because

motifs are more likely to share long hash prefix. It is important to note that the sublinear behavior in memory growth is due to our use of the tensoring implementation.

Impact of the quantization parameter r. In this experiment, we manually fix $r \in \{4, 8, 12, 16\}$ and compare it with the value automatically chosen by the algorithm using the procedure described in Section 6.1 in terms of the resulting number of distance computations, and hence running time. Figure 6 shows that the choice of r has a dramatic impact on the number of distance computations. Remarkably, the value automatically picked by our algorithm in a data-dependent way (and used in the rest of the experiments we presented) attains the same performance as the best fixed parameter considered in this experiment.

7.5 Quality of the results

Impact of probability threshold δ. We test the influence of δ (that controls the algorithm's failure rate) in the range from 0.05 to 0.4: larger values allow the algorithm to satisfy the stopping condition earlier, improving runtime at the cost of reduced recall. For each setting we measured the achieved recall, the relative runtime with respect to the stricter setting ($\delta = 0.05$), and the mean absolute relative error (MARE) of distances of the returned motifs with respect to the ground truth. In Figure 7 we present the recall-efficiency trade-off. We observe that the target recall imposed with δ is always satisfied, moreover, the fraction of time required with respect to our highest target is stable under similar contrast values (note the similar behavior of LTMM, WEATHER and RUTH, which share a relatively high contrast). About the quality of the discovered motifs at lower deltas, we found that the MARE never exceeds 3% for $\delta \leq 0.2$, indicating that the quality of the result is still preserved.

Impact of noise. In this last experiment we replicate the experimental design of [41], where the task is to reliably find motifs even when datasets are cluttered by noisy dimensions. To evaluate the robustness of our algorithm, we added from 4 to 256 additional dimensions, each being a random walk, to datasets POTENTIALS, EVAPORATOR, RUTH, WEATHER. Then we ran MOMENTI looking for the d-dimensional motif, with d as per Table 2, repeating each experiment 12 times. In all settings, irrespective of the number of additional noisy dimensions, our algorithm attained recall values of 1, showing its robustness to the number of irrelevant dimensions.

8 CONCLUSIONS

We presented a LSH based algorithm for the motif discovery problem in multidimensional time series, with strong guarantees on the quality of the results. Experimental evidence supports the efficacy and efficiency of our approach, demonstrating that it is able to find motifs with only a fraction of all possible distance computations.

REFERENCES

[1] Amaia Abanda, Usue Mori, and Jose A. Lozano. 2019. A review on distance based time series classification. *Data Min. Knowl. Discov.* 33, 2 (March 2019), 378–412. https://doi.org/10.1007/s10618-018-0596-4

[2] Alexandr Andoni and Piotr Indyk. 2006. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *FOCS*. 459–468. https://doi.org/10.1109/FOCS.2006.49

[3] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008), 117–122. https://doi.org/10.1145/1327452.1327494

[4] David Arnu, Edwin Yaqub, Claudio Mocci, Valentina Colla, Marcus Neuer, Gabriel Fricout, Xavier Renard, Christophe Mozzati, and Patrick Gallinari. 2017. A reference architecture for quality improvement in steel production. In *Data Science–Analytics and Applications: Proceedings of the 1st International Data Science Conference–iDSC2017*. Springer, 85–90.

[5] Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Michael Vesterli. 2019. PUFFINN: Parameterless and Universally Fast FInding of Nearest Neighbors. In *ESA (LIPIcs)*, Michael A. Bender, Ola Svensson, and Grzegorz Herman (Eds.), Vol. 144. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 10:1–10:16. https://doi.org/10.4230/LIPIcs.ESA.2019.10

[6] Arvind Balasubramanian, Jun Wang, and Balakrishnan Prabhakaran. 2016. Discovering Multidimensional Motifs in Physiological Signals for Personalized Healthcare. *IEEE Journal of Selected Topics in Signal Processing* 10, 5 (2016), 832–841. https://doi.org/10.1109/JSTSP.2016.2543679

[7] Eugen Berlin and Kristof Van Laerhoven. 2012. Detecting leisure activities with dense motif discovery. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (Pittsburgh, Pennsylvania) *(UbiComp '12)*. Association for Computing Machinery, New York, NY, USA, 250–259. https://doi.org/10.1145/2370216.2370257

[8] Eugen Berlin and Kristof Van Laerhoven. 2012. Detecting leisure activities with dense motif discovery. In *ACM Conference on Ubiquitous Computing*. ACM, New York, NY, USA, 250–259. https://doi.org/10.1145/2370216.2370257

[9] Michael R. Berthold and Frank Höppner. 2016. On Clustering Time Series Using Euclidean Distance and Pearson Correlation. arXiv:1601.02213 [cs.LG] https://arxiv.org/abs/1601.02213

[10] Carmelo Cassisi, Marco Aliotta, Andrea Cannata, Placido Montalto, Domenico Patanè, Alfredo Pulvirenti, and Letizia Spampinato. 2012. Motif Discovery on Seismic Amplitude Time Series: The Case Study of Mt Etna 2011 Eruptive Activity. *Pure and Applied Geophysics* 170, 4 (2012), 529–545. https://doi.org/10.1007/s00024-012-0560-y

[11] Matteo Ceccarello and Johann Gamper. 2022. Fast and Scalable Mining of Time Series Motifs with Probabilistic Guarantees. *Proc. VLDB Endow.* 15, 13 (Sept. 2022), 3841–3853. https://doi.org/10.14778/3565838.3565840

[12] Tobias Christiani. 2019. Fast Locality-Sensitive Hashing Frameworks for Approximate Near Neighbor Search. In *SISAP Proceedings*. Springer-Verlag, Berlin, Heidelberg, 3–17. https://doi.org/10.1007/978-3-030-32047-8_1

[13] Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. 2020. Confirmation Sampling for Exact Nearest Neighbor Search. In *SISAP Proceedings*. Springer-Verlag, Berlin, Heidelberg, 97–110. https://doi.org/10.1007/978-3-030-60936-8_8

[14] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of SCG*. ACM, New York, NY, USA, 253–262. https://doi.org/10.1145/997817.997857

[15] C. S. Daw, C. E. A. Finney, and E. R. Tracy. 2003. A review of symbolic analysis of experimental data. *Review of Scientific Instruments* 74, 2 (02 2003), 915–930. https://doi.org/10.1063/1.1531823

[16] B.L.R. (ed.) De Moor. 2024. *DaISy: Database for the Identification of Systems, Cutaneous potential recordings of a pregnant woman, 96-012*. Department of Electrical Engineering, ESAT/STADIUS, KU Leuven, Belgium. Retrieved September 3, 2024 from http://homes.esat.kuleuven.be/~smc/daisy/

[17] B.L.R. (ed.) De Moor. 2024. *DaISy: Database for the Identification of Systems, Data from an industrial evaporator, Process Industry Systems, 96-010*. Department of Electrical Engineering, ESAT/STADIUS, KU Leuven, Belgium. Retrieved September 3, 2024 from http://homes.esat.kuleuven.be/~smc/daisy/

[18] Dieter De Paepe, Diego Nieves Avendano, and Sofie Van Hoecke. 2020. Implications of Z-Normalization in the Matrix Profile. In *Pattern Recognition Applications and Methods*, Maria De Marsico, Gabriella Sanniti di Baja, and Ana Fred (Eds.). Springer International Publishing, Cham, 95–118.

[19] Rakshitha Godahewa, Christoph Bergmeir, Geoff Webb, Rob Hyndman, and Pablo Montero-Manso. 2021. *Oikolab Weather Dataset*. Zenodo. https://doi.org/10.5281/zenodo.5184708

[20] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation* 101, 23 (2000), e215–e220. https://doi.org/10.1161/01.CIR.101.23.e215 arXiv:https://www.ahajournals.org/doi/pdf/10.1161/01.CIR.101.23.e215

[21] Espen A. F. Ihlen, Aner Weiss, Jorunn L. Helbostad, and Jeffrey M. Hausdorff. 2015. The Discriminant Value of Phase-Dependent Local Dynamic Stability of Daily Life Walking in Older Adult Community-Dwelling Fallers and Nonfallers. *BioMed Research International* 2015, 1 (2015), 402596. https://doi.org/10.1155/2015/402596 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2015/402596

[22] Maria Inês Silva and Roberto Henriques. 2020. Exploring time-series motifs through DTW-SOM. In *IJCNN*. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207614

[23] Sean M. Law. 2019. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *Journal of Open Source Software* 4, 39 (2019), 1504. https://doi.org/10.21105/joss.01504

[24] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of Massive Datasets* (2nd ed.). Cambridge University Press, USA.

[25] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *SIGMOD*. ACM, New York, NY, USA, 2–11. https://doi.org/10.1145/882082.882086

[26] Bo Liu, Jianqiang Li, Cheng Chen, Wei Tan, Qiang Chen, and MengChu Zhou. 2015. Efficient Motif Discovery for Large-Scale Time Series in Healthcare. *IEEE Transactions on Industrial Informatics* 11, 3 (2015), 583–590. https://doi.org/10.1109/TII.2015.2411226

[27] Bo Liu, Huaipu Zhao, Yinxing Liu, Suyu Wang, Jianqiang Li, Yong Li, Jianlei Lang, and Rentao Gu. 2020. Discovering multi-dimensional motifs from multi-dimensional time series for air pollution control. *Concurrency and Computation: Practice and Experience* 32, 11 (2020), e5645. https://doi.org/10.1002/cpe.5645 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5645 e5645 CPE-19-1130.R1.

[28] Lucia Luzi, Giovanni Lanzano, Chiara Felicetta, Maria Chiara D'Amico, Emilio Russo, Sara Sgobba, Francesca Pacor, and ORFEUS Working Group 5. 2020. Engineering Strong Motion Database (Version 2.0). https://doi.org/10.13127/ESM.2

[29] David Minnen, Charles Isbell, Irfan Essa, and Thad Starner. 2007. Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery. In *ICDM*. 601–606. https://doi.org/10.1109/ICDM.2007.52

[30] Fabian Mörchen and Alfred Ultsch. 2005. Optimizing time series discretization for knowledge discovery. In *SIGKDD*. ACM, New York, NY, USA, 660–665. https://doi.org/10.1145/1081870.1081953

[31] David Murray, Lina Stankovic, and Vladimir Stankovic. 2017. An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study. *Scientific data* 4 (2017), 160122. https://doi.org/10.1038/sdata.2016.122

[32] NOAA Pacific Islands Fisheries Science Center. 2021. PIPAN 10kHz Data. https://doi.org/10.25921/Z787-9Y54 [Access date: 2024-10].

[33] Xavier Renard. 2017. *Time series representation for classification : a motif-based approach*. Theses. Université Pierre et Marie Curie - Paris VI. https://theses.hal.science/tel-01922186

[34] Kexin Rong, Clara E. Yoon, Karianne J. Bergen, Hashem Elezabi, Peter Bailis, Philip Levis, and Gregory C. Beroza. 2018. Locality-sensitive hashing for earthquake detection: a case study of scaling data-driven science. *Proc. VLDB Endow.* 11, 11 (July 2018), 1674–1687. https://doi.org/10.14778/3236187.3236214

[35] Anita Sant'Anna and Nicholas Wickström. 2011. Symbolization of time-series: An evaluation of SAX, Persist, and ACA. In *CISP*, Vol. 4. 2223–2228. https://doi.org/10.1109/CISP.2011.6100559

[36] Patrick Schäfer and Ulf Leser. 2024. Discovering Leitmotifs in Multidimensional Time Series. *Proc. VLDB Endow.* 18, 2 (2024), 377–389.

[37] Yoshiki Tanaka, Kazuhisa Iwamoto, and Kuniaki Uehara. 2005. Discovery of Time-Series Motif from MultiDimensional Data Based on MDL Principle. *ML* 58 (02 2005), 269–300. https://doi.org/10.1007/s10994-005-5829-2

[38] Alireza Vahdatpour, Navid Amini, and Majid Sarrafzadeh. 2009. Toward unsupervised activity discovery using multi-dimensional motif detection in time series. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (Pasadena, California, USA) *(IJCAI'09)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1261–1266.

[39] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for Similarity Search: A Survey. arXiv:1408.2927 [cs.DS] https://arxiv.org/abs/1408.2927

[40] Peng Xiao, Xingyu Cai, and Sanguthevar Rajasekaran. 2019. Efficient Algorithms for Finding Edit-Distance Based Motifs. In *Algorithms for Computational Biology*, Ian Holmes, Carlos Martín-Vide, and Miguel A. Vega-Rodríguez (Eds.). Springer International Publishing, Cham, 212–223.

[41] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. 2017. Matrix Profile VI: Meaningful Multidimensional Motif Discovery. In *ICDM*. 565–574. https://doi.org/10.1109/ICDM.2017.66

[42] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 1317–1322. https://doi.org/10.1109/ICDM.2016.0179

[43] Sheng Zhong and Abdullah Mueen. 2024. MASS: distance profile of a query over a time series. *Data Min. Knowl. Discov.* 38, 3 (2024), 1466–1492. https://doi.org/10.1007/s10618-024-01005-2

[44] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Abdullah Mueen, Philip Brisk, and Eamonn Keogh. 2016. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 739–748. https://doi.org/10.1109/ICDM.2016.0085