Persistence-based Motif Discovery in Time Series

Thibaut Germain, Charles Truong, and Laurent Oudre

***Abstract*—Motif Discovery consists of finding repeated patterns and locating their occurrences in a time series without prior knowledge about their shape or location. Most state-of-the-art algorithms rely on three core parameters: the number of motifs to discover, the length of the motifs, and a similarity threshold between motif occurrences. Setting these parameters is difficult in practice and often results from a trial-and-error strategy.**

In this paper, we propose a new algorithm that discovers motifs of variable length given a single motif length and without requiring a similarity threshold. At its core, the algorithm maps a time series onto a graph, summarizes it with persistent homology - a tool from topological data analysis - and identifies the most relevant motifs from the graph summary. We propose two versions of the algorithm, one requiring the number of motifs to discover and another, adaptive, that infers the number of motifs from the graph summary. Empirical evaluation on 9 labeled datasets, including 6 real-world datasets, shows that both algorithm versions significantly outperform state-of-the-art algorithms.

***Index Terms*—Time Series, Motif Discovery, Persistent Homology, Topological Data Analysis**

I. Introduction

TIME series are time-ordered sequences of real-valued samples. They appear in numerous domains, and their analysis has required developing specific data mining tools [13], [11]. Among them, Motif Discovery algorithms search for repeated patterns and locate their repetitions (also called occurrences) within a time series without prior knowledge about their shape or location. Motifs often provide insights into the process generating the time series. For example, electrocardiograms (ECGs) commonly present a motif corresponding to normal heartbeats. However, with patients suffering from premature heart contractions, the ECGs present a second motif specific to the malfunction [27] (see Figure 1-B). Motifs also summarize long time series with a limited number of patterns. They can be used in downstream analysis, such as classification or anomaly detection, to reduce time complexity, improve performances, or interpret results [43]. Motif Discovery algorithms have been used in various domains such as industry [40], [46], medicine [24], [41], and biology [21], [12].

The mathematical definition of motif is not unique and has led to algorithms based on different criteria such as motif frequency [25], [37], [18] or similarity between motif occurrences [48], [6], [47], [23]. Most of these algorithms rely on three core parameters: the number of motifs to discover, the length of motifs, and a similarity threshold between motif occurrences. These parameters highlight the current limitations of state-of-the-art algorithms. Indeed, the number of motifs to discover is imposed, and few guarantees exist whether the number of motifs is overestimated or underestimated. As well, the first algorithms supposed that all occurrences of all motifs have the same length [25], [2], [18], but more recent algorithms leverage this issue by searching for motifs within a length range [39], [23]. Finally, the similarity threshold is hard to determine as it depends on the variances between occurrences of each repeated pattern. In practice, this parameter is set by trial-and-error [37].

T.Germain, C.Truong, and L.Oudre are with Université Paris Saclay, Université Paris Cité, ENS Paris Saclay, CNRS, SSA, INSERM, Centre Borelli, F-91190, Gif-sur-Yvette, France. Email: {thibaut.germain, ctruong, laurent.oudre}@ens-paris-saclay.fr

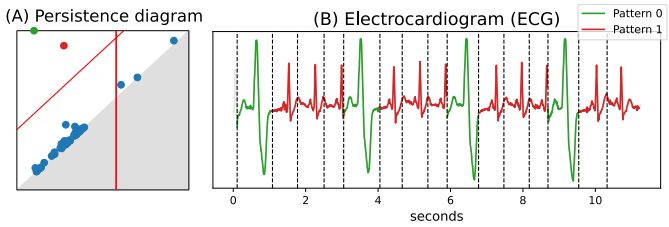


Fig. 1. Time series representation and motif sets discovered by our algorithm on an electrocardiogram (ECG) of a patient suffering from premature ventricular contraction (PVC). **(A) Persistence diagram:** it is a simplified representation of the ECG that shows the existence of two significant motif sets (in green and red). **(B) Electrocardiogram:** pattern 0 (green) represents heartbeats with PVC, and pattern 1 (red) represents normal heartbeats. Vertical dashed lines on the ECG indicate the start location of patterns' occurrences.

With regard to the current limitations, we propose, in this paper, a scalable algorithm that finds motifs of variable length without requiring a similarity threshold. The algorithm is based on a novel criterion: the persistence of motifs. Persistent homology is a central tool in topological data analysis [5] that efficiently tracks topological features at different spatial resolutions. In our context, the algorithm tracks motifs for all similarity thresholds and returns motifs that persist across the largest ranges of scales. Intuitively, the persistence of a motif simultaneously measures the similarity between its occurrences and their dissimilarity with the rest of the time series. The algorithm discovers motifs by mapping a time series onto a graph and creating a summary of the graph from which the most persistent motifs are identified. The algorithm also provides an intuitive visual representation of the graph summary, called persistence diagram, that informs about the number of relevant repeated patterns in a time series (see Figure 1-A). Taking advantage of this representation, we also present an adaptive version of the algorithm that infers the number of motifs to discover from the persistence diagram. Finally, in our experimental evaluation, we show that:

- Both algorithms significantly outperform 6 state-of-the-art algorithms on 9 labeled datasets, including 6 real-world datasets.
- Hyperparameters have limited influence on the algorithms' performances.

- Like state-of-the-art algorithms, the theoretical and empirical time complexity is quadratic in the time series' length.

The paper is organized as follows: In Section II, we review the related work and detail our contributions. In Section III, we present both algorithms. In Section IV, we describe the experimental settings, and in Section V, we review the experimental results.

II. Background

In this section, we recall some definitions related to time series and motif discovery, discuss related work and describe the scientific positioning of our approach.

A. Definitions

First, we recall standard definitions in time series analysis.

Definition 1. *(Time series) A time series of length n is a time-ordered sequence $S = [s_1, \ldots, s_n]$ of n real-valued coefficients.*

Definition 2. *(Subsequence) The subsequence of a time series $S \in \mathbb{R}^n$ of length l and starting at index $i \in [1, \ldots, n-l+1]$ is the sequence $S_i^l = [s_i, \ldots, s_{i+l-1}]$.*

We denote by $\mathbf{S}^l$ the set of all subsequences of length l of a time series S.

Definition 3. *(Overlapping subsequences) Two subsequences $(S_i^l, S_j^{l'})$ of a time series $S \in \mathbb{R}^n$ with $i < j$ overlap if $j \leq i + l$.*

We assume a distance function $d : \mathbb{R}^l \times \mathbb{R}^l \mapsto \mathbb{R}_+$ for the following definitions.

Definition 4. *(r-match) Given a threshold $r > 0$, the subsequences S_i^l and S_j^l of a time series $S \in \mathbb{R}^n$ are r-matching iff $d(S_i^l, S_j^l) < r$.*

Definition 5. *(Distance profile) The distance profile between $C \in \mathbb{R}^l$ and $S \in \mathbb{R}^n$ with $l < n$ is the sequence $[d(C, S_1^l), \ldots, d(C, S_{n-l+1}^l)]$.*

Several algorithms consider the following definitions of motif set:

Definition 6. *(Spherical motif set, [25]) Given a threshold $r > 0$, the spherical motif set associated with a sequence C of length l is the largest set of non-overlapping subsequences of length l of S such that all subsequences are r-matching with C. The sequence C is called the core element of the spherical motif set.*

Definition 7. *(Bi-spherical motif set, [23]) Given a threshold $r > 0$, the bi-spherical motif set associated with a pair of sequences (C_1, C_2) of length l is the largest set of non-overlapping subsequences of length l of S such that all subsequences are r-matching with either C_1 or C_2. The sequences C_1 and C_2 are called the core elements of the bi-spherical motif set.*

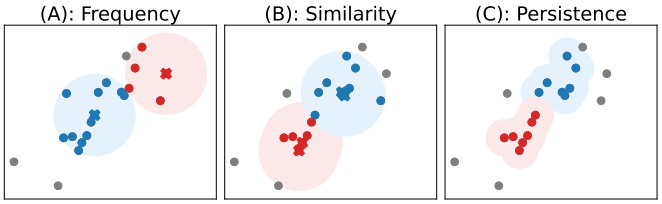


Fig. 2. Three approaches for motif set discovery. Points represent subsequences and algorithms search for two motif sets. Predicted sets are in red and blue, and points in gray are outliers. Cross points are subsequences and ball centers. **(A) Frequency based:** with SetFinder [2], clusters are the largest sets contained in non-overlapping balls of radius r centered on some subsequences. **(B) Similarity based:** with VALMOD [23], clusters are sets included in balls of radius r centered on the two most similar pairs of subsequences. **(C) Persistence based:** with PEPA, clusters are formed incrementally by similarity.

B. Related work

In the literature, motif discovery in time series refers to a number of distinct problems that belong to two primary categories:

- **Motif Pair Discovery**: Identifying the two most similar non-overlapping subsequences in a time series.
- **Motif Set Discovery**: Identifying sets of subsequences that encompass every occurrence of distinct repeated patterns in a time series.

In this paper, we focus on the motif set discovery task. For more information on motif pair discovery, we refer the interested reader to the reviews [43], [37].

Concerning motif set discovery, current algorithms address the problem in two different ways: frequency-based and similarity-based, which are described in the following. As will be seen, these approaches are disjoint and result in different definitions of motif sets, which are illustrated in Figure 2. Major algorithms for motif set discovery are summarized in Table I.

1) Frequency-based algorithms: The frequency-based algorithms aim at identifying sets of subsequences that represent the most frequently repeated patterns. The first motif set discovery algorithm, EMMA [25], follows this frequency-based approach. Given the number of motifs to discover, a fixed subsequence length, and a similarity threshold $r > 0$, the algorithm iteratively finds the largest spherical motif set with radius $r > 0$ and centered on a subsequence of length l of the time series. Additionally, the motif sets are chosen such that the spheres do not overlap, meaning that the core elements are at least a distance of $2r$ from each other. For computational efficiency, subsequences are discretized using Symbolic Aggregate approXimation (SAX) [22]. Subsequences with similar symbols are grouped in sets. The sets are refined in a post-processing step to obtain the final spherical motif sets according to the Euclidean distance. Due to the discretization, the EMMA algorithm provides approximate solutions. However, the more recent algorithm SetFinder [2], returns exact solutions when working with the Euclidean distance or the z-normalized Euclidean distance [8]. The algorithm computes the spherical motif set of all subsequences and selects the largest sets while preserving the non-overlapping constraint between motif sets. Core elements of motif sets for EMMA

TABLE I
VL: VARIABLE LENGTH, **NP:** NUMBER OF PARAMETERS, **COMPLEXITY:** WORST CASE TIME COMPLEXITY. n: TIME SERIES LENGTH, l: SUBSEQUENCE LENGTH, k: NUMBER OF MOTIF' OCCURRENCES, K: NUMBER OF SUBSEQUENCE' NEIGHBORS.

Approach	Algorithm	VL	NP	Complexity
Frequency	EMMA		5	$\mathcal{O}(ln^2)$
	SetFinder		3	$\mathcal{O}(n^3)$
	LearnMotifs		4	$\mathcal{O}(ln)$
	k-Motiflets		4	$\mathcal{O}(kn^2 + nk^2)$
	GrammarViz	✓	5	$\mathcal{O}(ln^2)$
Similarity	STOMP		3	$\mathcal{O}(n^2)$
	VALMOD	✓	5	$\mathcal{O}((l_{max} - l_{min})n^2)$
	MDLC	✓	3	$\mathcal{O}(n^3/l_{min} + (l_{max} - l_{min})n^2)$
Persistence	PEPA	✓	3	$\mathcal{O}(Kn^2)$
	A-PEPA	✓	2	$\mathcal{O}(Kn^2)$

and SetFinder are subsequences of a time series, making motif sets sensitive to noise. LatentMotif [18] addresses this issue by considering an optimization problem that maximizes the total motif frequencies to learn the core elements of the spherical motif sets. The non-overlapping constraint between spheres is relaxed and encoded via a penalty function incorporated in the optimization criteria. Nonetheless, there is no guarantee of an optimal solution as the optimization problem is non-convex.

Previous algorithms have considered the subsequence length as fixed and equal across motif sets, but a time series may contain repeated patterns of variable length. To address this issue, the Grammarviz algorithm [39] has been proposed. The time series is initially discretized using SAX representation as a long sequence of symbols. A grammar induction algorithm Sequitur [31], identifies the longest and most repeated patterns through a hierarchical structure of repeated patterns. While time-efficient, the algorithm is sensitive to the discretization step as repeated pattern identification relies on an exact match of symbolic representations of subsequences. Several variants of Grammarviz have been proposed to address this problem [38], [14], [15].

A recent algorithm, k-motiflets [37], focuses on finding the best motif set that contains k non-overlapping subsequences of a fixed length with minimal pairwise distances. Compared to previous algorithms, the similarity threshold is more intuitive and set based on the number of occurrences. In addition, the spheres are not centered on a subsequence but only need to contain k subsequences. The authors also suggest heuristics to determine appropriate numbers of occurrences and subsequence lengths. However, this algorithm focuses on one of the motif sets for a time series with multiple repeated patterns.

Frequency-based algorithms rely on a similarity threshold to determine the radius of spheres enclosing the motif sets. This threshold can be difficult to set and is assumed to be the same for all motif sets. When this assumption does not hold, the motif sets of repeated patterns may contain false occurrences or miss true occurrences.

2) Similarity-based algorithms: The similarity-based algorithms aim at identifying sets of subsequences that represent repeated patterns with minimal variability between occurrences, regardless of frequency. Roughly speaking, a pattern that is repeated only once or twice is detected by similarity-based methods but not by frequency-based methods.

An early similarity-based algorithm, MDLC [34], clusters subsequences of variable length such that the description length of the time series with the clusters is minimal. It is a greedy bottom-up algorithm that, at each iteration, either creates a cluster, adds a subsequence to a cluster, or merges two clusters. The choice of action is based on the number of bits saved. Unlike the more recent algorithms, STOMP [48], [3] and VALMOD [23], MDLC does not require the number of clusters as input; it stops clustering when the time series description length cannot be improved.

STOMP and VALMOD rely on the matrix profile [45], a structure that stores both the index and the distance to the nearest non-overlapping neighbor of all fixed-length subsequences of a time series. Given the number of motifs, a fixed subsequence length, and a similarity ratio $\lambda > 0$, the STOMP algorithm iteratively finds spherical motif sets whose core element corresponds to the left member of the most similar non-overlapping subsequence pair. STOMP also maintains the non-overlapping constraint between all subsequences of all motif sets. To that end, a mask containing all subsequences that overlap with previously selected motif sets is maintained across iterations. The most similar pairs are found with the matrix profile, and the associated spherical motif sets with an efficient algorithm to compute arbitrary distance profiles [29]. The second algorithm, VALMOD, differs from STOMP in two ways: it searches for motif sets with subsequences of variable length within a range $[l_{min}, \ldots, l_{max}]$ and it considers bi-spherical motif sets whose core elements are the most similar subsequence pairs. More precisely, the subsequences have the same length in each motif set, but the length can differ from one set to another. The z-normalized Euclidean distance is divided by the subsequence length to compare subsequences of different lengths. The matrix profile also stores the length of the nearest non-overlapping subsequence. Finally, motif sets are found using the same resolution scheme as STOMP algorithm.

In contrast to frequency-based algorithms, similarity-based algorithms assume that the radius of the balls in which motif sets are contained differs for each set. The radius is proportional to the z-normalized Euclidean distance between the subsequences of the most similar pair associated with the motif set. The proportionality is defined by a similarity ratio $\lambda > 0$. With such a definition, motif sets are sensitive to the nearest neighbor pairs, and small perturbations can lead to different sets.

C. Contributions and scientific positioning

All presented algorithms consider motif sets as collections of subsequences contained in balls whose radius is determined by a similarity threshold. However, setting this parameter is not straightforward as it requires prior knowledge about the similarity between subsequences of repeated patterns. In practice, the threshold is often set by trial and error [37], but this strategy is not tractable for large time series.

Our algorithm, called PersistentPattern (PEPA), takes a different approach. It creates motif sets without any prior

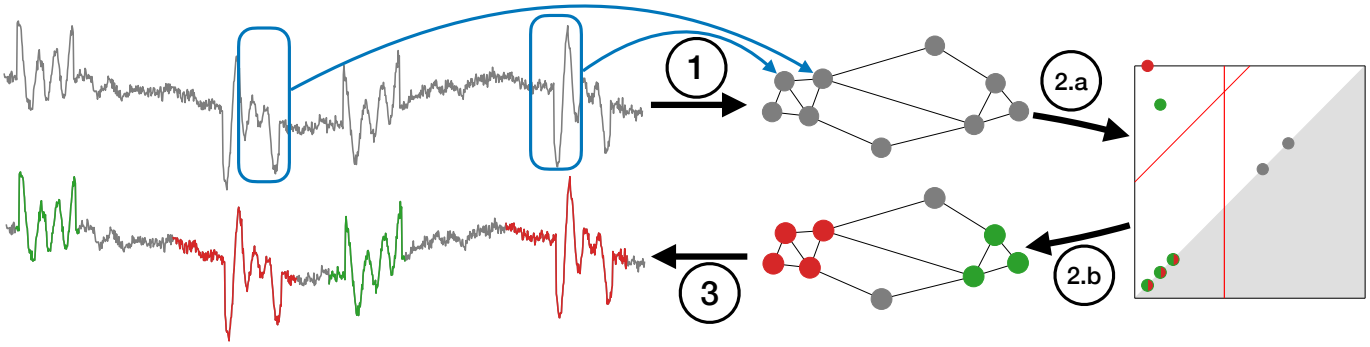


Fig. 3. **(1) From time series to graph:** Nodes are subsequences, and edges depend on the distance between them. **(2) Graph clustering from persistent homology. (a) From graph to persistence diagram:** The graph is summarised by a diagram where each point is a connected subgraph, and its location depends on edge weight. **(b) From persistence diagram to clusters:** Subgraphs associated with motif sets are in the upper left corner. They can be isolated with two thresholds: the red lines. There are two clusters, red/green; the bi-colored points are subgraphs included in the subgraphs of cluster, but their membership cannot be determined from the persistence diagram. The gray points are subgraphs associated with irrelevant parts of the time series. **(3) From clusters to motif sets:** time-adjacent subsequences are merged in each cluster.

knowledge about the similarity between subsequences. Indeed, the algorithm generates motif sets for all possible similarity threshold values, ranging from 0 to $+\infty$. Some motif sets persist across multiple threshold values, and the algorithm selects motif sets with the largest persistence range. In addition, the algorithm does not impose a spherical constraint on the motif sets. Instead, the subsequences are incrementally grouped to form the motif sets based on their similarity. It allows the motif sets to adapt to the local shape of the neighborhood of the subsequences. Ultimately, the algorithm searches for repeated patterns of variable length that are significantly different from each other regardless of their frequency. This approach is illustrated in Figure 2.

All presented algorithms also require the number of motif sets as a parameter. Although difficult to define without prior knowledge, our persistence-based approach provides a simplified representation of a time series from which it is possible to infer the number of motif sets. Thus, we will also present an adaptive version of the main algorithm (A-PEPA) that automatically infers the number of motif sets to be discovered.

Algorithm 1 ComputeGraph

Require: S a time series, l the subsequence length, K the number of neighbor

$n \leftarrow Length(S),\ Graph1 \leftarrow (),\ Graph2 \leftarrow ()$
$DistanceInitialization(S, l)$
for i=1,...,n-l+1 **do**
  $D \leftarrow ComputeDistanceProfile(S_i^l, S)$
  $D[\max(i-l+1,0), \min(i+l-1,n)] \leftarrow +\infty$
  for k=1,...,K **do**:
    $j \leftarrow \arg\min(D)$
    $Graph1.append((i, j, D[j]))$
    $D[\max(j-l+1,0), \min(j+l-1,n)] \leftarrow +\infty$
    if k==1 **then**
      if i==1 **then**
        $d^{prev} \leftarrow D[j]$
      else
        $d^{\max} \leftarrow \max(d^{prev}, D[j])$
        $Graph2.append((i, i+1, d^{\max}))$
        $d^{prev} \leftarrow D[j]$
$Graph \leftarrow Combined(Graph1, Graph2)$
return $Graph$

III. Method

Our approach is based on two main ingredients: a graph that encodes the structural relationships between all subsequences in the time series, and the use of persistent homology (a tool derived from topological data analysis) to isolate and identify the motif sets. The algorithm PEPA can be broken down into three steps illustrated in Figure 3:

1) **From time series to graph:** Transforming a time series into a graph where nodes represent subsequences and edges are weighted with the distance between subsequences. The graph is an adaptation of the k-nearest neighbor graph, which incorporates similarity and time dependence of subsequences.
2) **Graph clustering through persistent homology:** Identifying clusters representing motif sets and separating them from the isolated nodes of the graph representing irrelevant parts of the time series.
3) **From clusters to motif sets:** Merging temporally adjacent subsequences in each cluster to form the variable length motif sets.

A. From time series to graph

This section describes the transformation of a time series $S \in \mathbb{R}^n$ into an undirected weighted graph $\mathcal{G}_S$. For now, assume a distance $d(\cdot,\cdot)$ between subsequences, which will be described later.

Definition 8 (Undirected weighted graph)**.** *An undirected weighted graph $\mathcal{G} = (V, E)$ consists of a set of vertices (also called nodes) V and a set of weighted edges $E \subset \{(x, y, w_{xy}) \mid (x, y) \in V \times V,\ w_{xy} \in \mathbb{R}_+\}$, where w_{xy} is the weight of the edge between nodes x and y.*

In $\mathcal{G}_S$, the set of nodes is composed of all subsequences of length l of S (denoted as $\mathbf{S}^l$) and the edges between

subsequences are defined according to two criteria:

- **Similarity:** each subsequence S_i^l is connected to its K most similar non-overlapping subsequences.
- **Time:** it connects with its time adjacent subsequences (S_{i-1}^l, S_{i+1}^l).

More formally, let N_i^k denote the k-th nearest non-overlapping neighbor of the subsequence S_i^l and d_i^k the distance between subsequences S_i^l and N_i^k. The **similarity** edges are defined by

$$E_1 = \bigcup_{i=1}^{n-l+1} \left\{ \left(S_i^l, N_i^k, d_i^k\right) \mid k = 1, \ldots, K \right\}, \quad (1)$$

and the **time** edges, by

$$E_2 = \bigcup_{i=1}^{n-l} \{(S_i^l, S_{i+1}^l, \max\left(d_i^1, d_{i+1}^1\right)\}. \quad (2)$$

The final graph $\mathcal{G}_S$ is defined as $(\mathbf{S}^l, E_1 \cup E_2)$. Intuitively, low-weight edges connect similar subsequences, while high-weight edges connect less similar ones. The graph $(\mathbf{S}^l, E_1)$ is a variant of the well-known k-nearest neighbor graph, as it connects each subsequence to its k-nearest non-overlapping neighbors. In practice, this graph can split a single motif set into several clusters as subsequences are considered independent and they are not compared with time-adjacent subsequences. In such a situation, the number of discovered motif sets is over-estimated, and a non-trivial post-processing is needed to merge similar clusters. We introduce the time edges set E_2 to prevent this phenomenon: in our graph, $\mathcal{G}_S$, a subsequence is always connected to the subsequences just before S_{i-1}^l and after S_{i+1}^l. As a result, our method assigns overlapping subsequences that represent the same repeated pattern in a single cluster, which limits the over-clustering effect.

Proposition 1. *The graph $\mathcal{G}_S$ associated with the time series $S \in \mathbb{R}^n$ is connected.*

Proof. The graph $(\mathbf{S}^l, E_2)$ is a path graph; thus, it is connected, and by the union of two graphs, $\mathcal{G}_S$ is connected. □

The construction of the graph $\mathcal{G}_S$ relies on a distance between time series. To that end, we introduce a novel distance, which has several advantageous properties, particularly robustness to Linear Trends (LT). It is called the γ-rectified LT-normalized distance. It is based on the LT-normalized (Euclidean) distance which is an extension of the recently proposed LT-normalized (Euclidean) distance [16], and is defined as follows.

Definition 9 (LT-normalized Euclidean distance)**.** *The LT-normalized (Euclidean) distance between $x \in \mathbb{R}^w$ and $y \in \mathbb{R}^w$ is:*

$$d_{LT}(x, y) = \left\| \frac{x - (\alpha_x \mathbf{t} + \beta_x \mathbf{1})}{\|x - (\alpha_x \mathbf{t} + \beta_x \mathbf{1})\|} - \frac{y - (\alpha_y \mathbf{t} + \beta_y \mathbf{1})}{\|y - (\alpha_y \mathbf{t} + \beta_y \mathbf{1})\|} \right\|$$

where $\mathbf{t} = (0, \ldots, w-1)$, $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^w$ and (α_x, β_x) are solutions of the linear regression problem:

$$\arg\min_{(a,b)\in\mathbb{R}^2} \|x - (a\mathbf{t} + b\mathbf{1})\|^2 \quad (3)$$

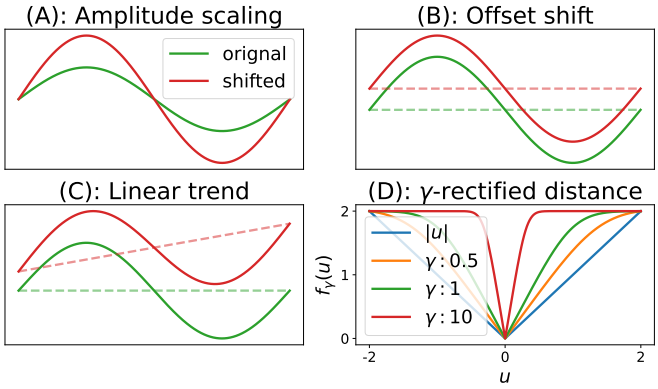


Fig. 4. Shape deformations of time series due to **(A) Amplitude, (B) Offset, (C) Linear** shifts. In green the original time series and in red the shifted time series. The dot lines represent the deformations. **(D) γ-rectified distance:** The blue line represents distance behavior with the absolute value on the range of $[-2, 2]$. The other colors depict γ-rectified distances. As γ increases, the distance stays low on smaller neighborhoods centered on 0 but tends towards 2 outside the neighborhood.

The LT-normalized distance generalizes the Z-normalized distance [44] by removing the linear trend of subsequences and scaling them to unit norm. Thanks to its invariance to offset, linear, and amplitude shifts (see Figure 4-(A-C)), the LT-normalized distance is robust to shape deformations caused by a trend. Also, distances between noisy linear sequences remain high, and distances between noisy occurrences of a complex shape are low. Consequently, our approach, PEPA, and its adaptive version, A-PEPA, focus on repeated patterns that are not linear up to any offset, linear, or amplitude shifts.

Intuitively, the distance should be able to distinguish between subsequences that belong to the same repeated pattern and those that do not. To enforce this behavior, we introduce the γ-rectified LT-normalized Euclidean distance, which produces a soft polarization of the distance values at the limits of the interval $[0, 2]$.

Definition 10. *Given $\gamma > 0$, for any $x \in \mathbb{R}^w$ and $y \in \mathbb{R}^w$, the γ-rectified LT-normalized Euclidean distance, denoted $d_\gamma(\cdot, \cdot)$, is defined by*

$$d_\gamma(x, y) = f_\gamma\left(d_{LT}(x, y)\right), \; f_\gamma(u) = 2\sqrt{\frac{\tanh(\gamma u^2)}{\tanh(4\gamma)}} \quad (4)$$

where $\gamma > 0$ controls the polarization.

Figure 4-D shows the influence of the parameter γ on the function f_γ. For all experiments, the value of γ is identical and fixed to 10.

Algorithm 1 describes the computation of the graph with the γ-rectified LT-normalized Euclidean distance. The algorithm conjointly computes the edges by looping over the distance profile of all subsequences. Thanks to the recursion property of the LT-normalized distance described in [16], each distance profile is computed in $\mathcal{O}(n)$, resulting in a time complexity of $\mathcal{O}(Kn^2)$ for computing the graph. In addition, parallel and GPU computing of the graph is also feasible [16].

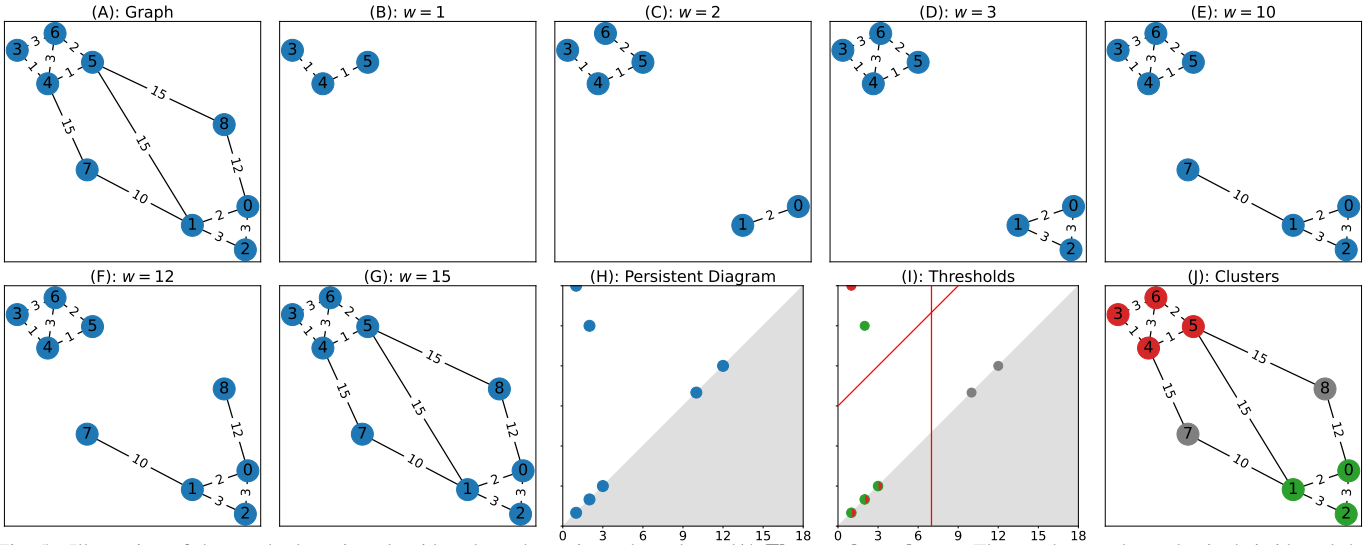


Fig. 5. Illustration of the graph clustering algorithm through persistent homology. **(A) The graph to cluster:** The number on the nodes is their id, and the weights are the distance between nodes. **(B)-(G) The NNVR filtration milestones of the graph:** The edges are added in order of increasing weight. **(H) The corresponding persistence diagram:** The births and deaths of connected subgraphs traced along the filtration are summarized with a 2D scatterplot where births are on the x-axis and deaths are on the y-axis. **(I) The birth and persistence thresholds:** The red and green points in the upper-left corner indicate two clusters. The birth threshold (vertical red line) and the persistence threshold (off-diagonal red line) are set to isolate this region. The bi-colored points are subparts of the cluster, but their membership cannot be determined from the persistence diagram. The gray points are associated with irrelevant parts of the time series. **(J) Clustering result:** Clusters are in red and green; irrelevant nodes are in gray.

B. Graph clustering through persistent homology

The graph $\mathcal{G}_S$ is such that overlapping subsequences from the same motif set are connected with edges of low weight. Therefore, motif sets can be retrieved by searching for connected subgraphs of $\mathcal{G}_S$ with edges of low weight. Persistent homology is well-suited for identifying and isolating such subgraphs. Persistent homology is a central tool in the field of topological data analysis [5], used to track the persistence of topological features of data at multiple scales with respect to a scaling parameter. To summarize the persistence of these features, a 2D scatter plot known as a Persistence Diagram is created, allowing for the identification of noteworthy topological features. For a thorough description of persistent homology and its application in various fields, readers can refer to [10], [33].

In our context, the topological features are the connected subgraphs of $\mathcal{G}_S$, and the scaling parameter is the edge weight. The graph clustering algorithm consists of three steps:

1) Computing the persistence of connected subgraphs
2) Identifying connected subgraphs related to motif sets from the persistence diagram
3) Forming clusters from the chosen connected subgraphs

1) Computing the persistence of connected subgraphs: The persistence of connected subgraphs of $\mathcal{G}_S$ is computed through a sequence of nested graphs. The sequence starts with the empty graph and adds edges one by one, in order of increasing weight until it reaches the final graph. In persistent homology, such sequence is called a filtration. There are several types of filtration, and we have implemented the Nearest Neighbor Vietoris-Rips Filtration (NNVR) [4].

Definition 11 (Nearest Neighbor Vietoris-Rips Filtration)**.** *Let $\mathcal{G}_S = (V, E)$ be the graph of a time series S and $(w_i)_{i=1,\dots,m}$ the edge weights in ascending order. The nearest neighbor Vietoris-Rips filtration is the sequence of nested graphs:*

$$\emptyset = \mathcal{G}_{w_1} \subsetneq \mathcal{G}_{w_2} \subsetneq \ldots \subsetneq \mathcal{G}_{w_{m-1}} \subsetneq \mathcal{G}_{w_m} = \mathcal{G}_S$$

where $\emptyset$ is the empty graph and $\mathcal{G}_{w_i} = (V_{w_i}, E_{w_i})$ such that:

$$V_{w_i} = \left\{ S_x \in V \mid \min_{(S_x, S_y, w_{xy}) \in E} w_{xy} \leq w_i \right\}$$

and

$$E_{w_i} = \{(S_x, S_y, w_{x,y}) \in E \mid w_{xy} \leq w_i\}$$

By convention, when adding an edge in the filtration, the nodes it connects are added first if they are not already in the filtration, then the edge itself is added. If both nodes need to be added, one is arbitrarily added before the other. Alongside the filtration, we keep track of the birth and death dates of connected subgraphs:

- **Birth:** The birth of a connected subgraph occurs when a node is added to the graph; its birth date is equal to the weight of the associated edge.
- **Death:** A connected subgraph dies when an edge connects it to an older connected subgraph. Its death date is equal to the weight of the connecting edge.

By definition, each subsequence is the seed node of a connected subgraph, so they all have a birth date equal to the distance to their nearest non-overlapping neighbors. Since the graph of a time series $\mathcal{G}_S$ is connected, one connected subgraph never dies; its death date is set to $+\infty$. We denote $(b_i, d_i)_{i \in I}$ as the set of birth and death dates of all connected subgraphs traced by the filtration. The persistence of the i^{st} connected subgraph corresponds to its lifetime: $d_i - b_i$. The connected subgraphs are summarized with a 2D-scatterplot called Persistence Diagram, where births are on the x-axis, and deaths are on the y-axis. Each point is counted with

Algorithm 2 ComputePersistence

Require: $\mathcal{G} = (i, j, w_{ij})_{(i,j)\in I}$ a graph, I is sorted by weight.
1: $\mathcal{P} \leftarrow \{\}$ Parent dictionary, $\mathcal{B} \leftarrow \{\}$ Birth dictionary, $\mathcal{D} \leftarrow \{\}$ Death dictionary, $MST \leftarrow ()$ Minimum spanning tree
2: **for** $(i, j) \in I$ **do**
3: $\quad P_1 \leftarrow FindParent(i)$
4: $\quad$ **if** P_1 is empty **then**
5: $\quad\quad \mathcal{P}[i] \leftarrow i,\ \mathcal{B}[i] \leftarrow w_{ij}$
6: $\quad P_2 \leftarrow FindParent(j)$
7: $\quad$ **if** P_2 is empty **then**
8: $\quad\quad \mathcal{P}[j] \leftarrow j,\ \mathcal{B}[j] \leftarrow w_{ij}$
9: $\quad$ **if** $P_1 \neq P_2$ **then**
10: $\quad\quad$ **if** $\mathcal{B}[P_1] < \mathcal{B}[P_2]$ **then**
11: $\quad\quad\quad \mathcal{P}[P_2] \leftarrow P_1,\ \mathcal{D}[P_2] \leftarrow w_{ij}$
12: $\quad\quad$ **else**
13: $\quad\quad\quad \mathcal{P}[P_1] \leftarrow P_2,\ \mathcal{D}[P_1] \leftarrow w_{ij}$
14: $\quad\quad MST.append((i, j, w_{ij}))$
15: **return** $\mathcal{B}, \mathcal{D}, MST$

multiplicity since several connected subgraphs can have the same birth and death dates.

Figure 5 shows milestones of an NNVR filtration on a graph (Figure 5-A to Figure 5-G) and the corresponding persistence diagram (H). When weight $w = 1$, node 3 has killed nodes 4 and 5, so their persistence is zero. With weight $w = 2$, node 0 kills node 1 to form a second independent connected subgraph. Then, nodes 6, 2, 7, and 8 are added and immediately killed for weights $w = 2, 3, 10$, and 12. At weight $w = 15$, the subgraph associated with node 0 is killed by that of node 3 and the graph is complete. The highest point (in red) on the persistence diagram corresponds to the subgraph that never dies.

The birth and death of connected subgraphs are tracked by maintaining a union-find data structure throughout the filtration. The algorithm is equivalent to the Kruskall's algorithm for computing the minimum spanning tree (MST). Algorithm 2 describes the procedure for computing the connected subgraphs persistence from a graph whose edges are ordered by increasing weight. The FindParent command follows the chain of parent pointers from a query node until a root node. This root node represents the connected subgraph to which the query node belongs. The algorithm also stores the MST to efficiently retrieve the clusters later on.

2) Identifying connected subgraphs related to motif sets from the persistence diagram: As shown in Figure 6-A, the persistence diagram can be divided into three interpretable regions:

1) **Top-left corner:** Points represent connected subgraphs associated with motif sets.
2) **Lower-left corner:** Points represent connected subgraphs associated with minor variations of the motif sets.
3) **Right side:** Points represent connected subgraphs associated with irrelevant parts of the time series.

On one hand, the graph associated with a time series is constructed so that non-repeating or noisy linear subsequences are far from any other subsequence. Thus, connected subgraphs composed of these subsequences have a late birth and

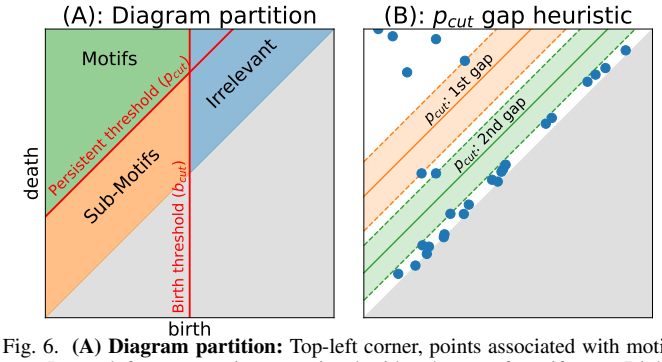


Fig. 6. **(A) Diagram partition:** Top-left corner, points associated with motif sets. Lower-left corner, points associated with subparts of motifs sets. Right side, irrelevant parts of the time series. **(B) Persistence threshold heuristic:** The orange line is the persistence threshold of the largest gap, and the green line is the threshold of the second largest gap.

are located in the right part of the persistence diagram. On the other hand, subsequences that overlap any occurrence of a repeated pattern are close to each other and far from all other subsequences. The connected subgraphs associated with repeated patterns have early births and late deaths. They are located in the top-left corner of the persistence diagram. Finally, the lower-right corner corresponds to connected subgraphs associated with minor variations of repeated patterns.

The region associated with motif sets can be isolated with a vertical line and an off-diagonal line as presented in Figure 6-A. The vertical line corresponds to a threshold on birth dates (b_{cut}). It defines the difference between irrelevant subsequences and subsequences belonging to motif sets. This threshold is computed with Otsu's method [32], an algorithm introduced in image processing to transform a grayscale image into a black-and-white image. The off-diagonal line corresponds to a threshold on the persistence (p_{cut}). Connected subgraphs associated with motif sets have a persistence greater than this threshold. To discover N motif sets, the persistence threshold is set to the average between the N-th and $(N+1)$-th most persistent connected subgraphs whose birth dates are less than the birth threshold.

3) Create clusters from the selected connected subgraphs: The clusters are computed by maintaining a union-find data structure throughout the filtration of the minimum spanning tree (MST) of $\mathcal{G}_S$. The MST is computed in the first clustering step (Section III-B1), and it is the smallest graph that contains all information about the birth and the death of all the connected subgraphs traced by the filtration of $\mathcal{G}_S$. The algorithm that computes the clusters is similar to the ComputePersistence algorithm (Algorithm 2) with two modifications. Specifically, the edges connecting two connected subgraphs with persistence higher than the persistence threshold are not considered when going through the filtration. This is done by changing the condition of the if-loop Line 9 to $(P_1 \neq P_2) + ((w_{ij} - \mathcal{B}[P_1] \leq p_{cut}) * (w_{ij} - \mathcal{B}[P_2] \leq p_{cut}))$. Second, after the main for-loop, the parent dictionary is updated, and the nodes whose birth date exceeds the birth threshold are removed. Ultimately, the algorithm returns N clusters from the parent dictionary, each composed of subsequences of length l of S.

Algorithm 3 ComputeMotifSets

Require: S time series, I indexes of subsequences in clusters, $\mathcal{C}$ cluster dictionary, $\mathcal{B}$ birth dictionary, l subsequence length.

$\mathcal{O} \leftarrow \{\}$ Occurrence dictionary, $IDX \leftarrow ()$ Index to keep, $M \leftarrow ()$ Motif sets, $p\_idx \leftarrow I[0] - 1$ Previous index, $o\_id \leftarrow 1$ Occurrence ID
for $i \in sort(I)$ **do**
  if $(i - 1 \neq p\_idx)$ or $(\mathcal{C}[i] \neq \mathcal{C}[p\_idx])$ **then**
    $o\_id \leftarrow o\_id + 1$
  $\mathcal{O}[i] \leftarrow o\_id$
$I' \leftarrow BirthOrderedIndex(I, \mathcal{B})$ ▷ *order the index list I by increasing order of birth date.*
for $i \in I'$ **do**
  $IndexToKeep = True$
  $J \leftarrow OverlappingIndex(i, IDX, l)$ ▷ *select index in IDX overlapping with i.*
  if J *is not empty* **then**
    for $j \in J$ **do**
      if $O[i] \neq O[j]$ **then**
        $IndexToKeep = False$
        break
  if $IndexToKeep$ *is* $True$ **then**
    $IDX.append(i)$
$M \leftarrow MotifSetsFromSubsequences(S, IDX, \mathcal{C}, l)$
return M

C. From clusters to motif sets

Recall that a motif set is a set of non-overlapping subsequences of possibly varying length where elements of each motif should represent occurrences of the same repeated pattern. Our clustering approach produces clusters of overlapping subsequences, which must be refined to create a motif set. In particular, we merge specific overlapping subsequences to form a single one representing an occurrence. To that end, subsequences with the latest birth date are removed until the non-overlapping constraint is satisfied when merging subsequences into a single one per occurrence. The refined clustering is then a motif set.

Algorithm 3 shows the procedure for computing the motif sets from clusters. The first for-loop computes the occurrence membership of all subsequences. Two subsequences belong to the occurrence if they are in the same cluster like all temporally consecutive subsequences between them. The second for-loop refines the clusters to enforce the non-overlapping constraint. If a subsequence overlaps with at least one subsequence of another occurrence with an earlier birth date, it is removed from its cluster. Finally, the motif sets are formed by merging the temporally consecutive subsequences in each cluster.

D. Adaptive algorithm: A-PEPA

In this section, we present an adaptive version of the PEPA algorithm called A-PEPA, which infers the number of motif sets from the persistence diagram. The only difference between PEPA and A-PEPA is the computation of the persistence threshold.

Algorithm 4 AdaptivePersistenceThreshold

Require: $\mathcal{B}$ birth dictionary, $\mathcal{D}$ death dictionary, b_{cut} birth threshold, M number of gap

$P \leftarrow ()$ persistence list, $p_{cut} \leftarrow 0$ persistence threshold
for $i = 1, \ldots, n - l + 1$ **do**
  if $\mathcal{B}[i] \leq b_{cut}$ **then**
    $P.append(\mathcal{D}[i] - \mathcal{B}[i])$
$P \leftarrow Sort(P)$
for $i = 1, \ldots, M$ **do**
  $j \leftarrow \arg\max(P),\ p_{cut} \leftarrow (P[j+1] - P[j])/2$
  $P \leftarrow P[0 : j + 1]$
return p_{cut}

The PEPA algorithm isolates motif sets with a birth threshold and a persistence threshold based on the number of motif sets to discover. The adaptive version of the algorithm, A-PEPA, infers the persistence threshold by looking at successive gaps in persistence as shown in Figure 6-B. A large gap indicates that repeated patterns (points above the gap) significantly differ from all other patterns in the time series (points below the gap). Depending on the application, the second or higher-order gap may be more interesting than the largest; some variations of more persistent repeated patterns should be considered as different motif sets. Algorithm 4 shows the procedure for computing the adaptive persistence threshold. In practice, we set the adaptive persistence threshold to the second-largest persistence gap.

E. Time complexity and parameter tuning

The time complexity of PEPA and A-PEPA is in $\mathcal{O}(Kn^2)$, where n is the length of the time series, and K is the number of nearest neighbors.

Indeed, the graph $\mathcal{G}_S$ is computed in $\mathcal{O}(Kn^2)$ by following the procedure of the STOMP algorithm [48]. The graph clustering algorithm is in $\mathcal{O}(Kn\log(Kn))$ in the worst case because the algorithm 2 requires maintaining a union-find data structure over $\mathcal{G}_S$ which has Kn edges. The computation of both thresholds is in $\mathcal{O}(n)$, and the algorithm that computes the clusters from the selected connected subgraphs is in $\mathcal{O}(n\log(n))$ since it requires maintaining a union-find data structure of the MST of $\mathcal{G}_S$ which has n edges. The algorithm 3 is in $\mathcal{O}(n\log(n))$ in the worst case because it requires sorting the subsequences by increasing order of birth dates. The bottleneck of PEPA and A-PEPA is the computation of the graph in $\mathcal{O}(Kn^2)$.

F. Parameter tuning

The PEPA algorithm has three parameters:

- The number of motif sets to discover: $N \in \mathbb{N}^*$. Note that this number is empirically estimated when using A-PEPA.
- Two parameters linked to the graph construction: the length of subsequences $l \in \mathbb{N}^*$ and the number of nearest neighbors $K \in \mathbb{N}^*$.

Like other motif discovery algorithms, setting the number of motif N depends on expert knowledge. However, with PEPA,

this number can be deducted through the persistence diagram, Figure 6, and the motifs sets can be updated in $\mathcal{O}(n\log(n))$.

Empirical results (Section V-C2) shows that PEPA and A-PEPA are not sensitive to the number of neighbors K when it exceeds 5 (the relative error to the optimal is less than 1%). As this parameter influences the algorithms' computational time, we advise setting it to 5.

Experiment on the influence of the window length l (Section V-C1) shows that PEPA and A-PEPA retrieve motifs whose length are at most twice the window length. In practice, we recommend setting it to the length of the smallest motif.

IV. Experimental settings

This section describes the datasets, performances metrics and implementation details we used for our experimental evaluation. For reproducibility, the source code and all datasets are available on our webpage [1].

A. Datasets

We conducted our experiments on 9 labeled datasets constructed from real and synthetic time series. Table II presents the main characteristics of the datasets for the motif set discovery problem. The datasets are described in more detail in the following paragraphs.

1) Synthetic datasets: We have generated datasets based on three scenarios of increasing complexity:

(S-1) **single**: There is 1 pattern of length 100 that repeats 50 times.

(S-2) **fixed**: There are 5 patterns of length 100. For each pattern, the number of occurrences is sampled uniformly between 2 and 10.

(S-3) **variable**: There are 5 patterns with length uniformly sampled between 100 and 200. For each pattern, the number of occurrences is sampled uniformly between 2 and 10.

All time series are generated using the same protocol: occurrences of the N repeated patterns are randomly placed on top of a random walk, and Gaussian noise is added to the resulting time series. The amplitude of the random walk (resp. Gaussian noise) is set to 0.2 (resp. 0.1). The interval between two consecutive occurrences is also uniformly sampled over $[10, 90]$ for the single/fixed scenarios and $[20, 180]$ for the variable-length scenario. Given a length of $l_0 \in \mathbb{N}^*$ and a fundamental frequency of $4Hz$, a pattern is generated as the sum of the sine function of the l_0 first harmonics, with the phases and the amplitudes are uniformly sampled over $[-\pi, \pi]$ and $[-1, 1]$.

2) Real datasets: We used four real datasets already used in the litterature:

(R-1) **mitdb-1** [17], [27]: The MIT-BIH Arrhythmia Database contains 48 half-hour recordings of two-channel ambulatory electrocardiograms (ECGs) sampled at $360Hz$. Cardiologists annotated the heartbeats according to 19 categories[1]. We divided all recordings into time series of 1 minute and kept the first channel. We selected time series of healthy subjects (id: 100, 101, 103, 117, 122, according to [35]) that contains only normal heartbeats, and randomly selected 100 time series.

(R-2) **mitdb-2**: We randomly selected 100 one-minute time series in from MIT-BIH dataset. This dataset is more challenging than the previous one as it contains unhealthy heartbeats. Each time series has 1 to 4 patterns, each with several occurrences.

(R-3) **mitdb800** [19]: This database includes 78 half-hour ECG recordings sampled at $120Hz$ with heartbeat annotations (19 categories). We divide all recordings into three-minute time series and keep the first channel. We randomly select 100 time series, and the number of repeated patterns varied between 1 and 4.

(R-4) **ptt-ppg** [26]: Pule-Transit-Time PPG dataset consists of time series recorded with multiple sensors (sampled at $500Hz$) from healthy subjects performing physical activities. Heartbeats are also annotated. We randomly select a hundred 40-second long signals from the photoplethysmogram (PPG) first channel during the "run" activity.

(R-5) **refit** [30]: The original dataset provides aggregate and individual appliance load curves at 8-second sampling intervals from 20 houses in the United Kingdom, recorded over two years. We selected 10 houses and aggregated recordings of the appliances available: dishwasher, food mixer, washing machine, and tumble dryer. The recordings were down-sampled to 32-second intervals and divided into time series of one week. We kept 10 time series for each house in which the appliances were not used simultaneously. This resulted in a dataset of 100 univariate time series with a maximum of 3 motif sets.

(R-6) **arm-coda** [7] is a dataset of 240 multivariate time series collected using 34 Cartesian Optoelectronic Dynamic Anthropometers (CODA) placed on the upper limbs of 16 healthy subjects, each of whom performed 15 predefined movements such as raising their arms or combing their hair. Each sensor records its position in 3D space. To construct the dataset, we kept the left (resp. right) forearm sensor of id 29 (resp. 20) and 5 of the predefined movements: 0,1,4,6,8 (resp. 0,1,4,5,7). We selected the first two occurrences of all movements in the x and y dimensions. Then, the occurrences of the 5 movements

[1]https://archive.physionet.org/physiobank/annotations.shtml

TABLE II
N NUMBER OF REPEATED PATTERNS, IF $< k$, THERE ARE AT MOST k PATTERNS. μ_l AVERAGE PATTERN LENGTH, σ_l STANDARD DEVIATION OF PATTERN LENGTH, MIN/MAX MINIMUM/MAXIMUM PATTERN LENGTH, n TIME SERIES LENGTH, # NUMBER OF TIME SERIES.

Type	Name	N	μ_l	σ_l	min/max	n	#
real	(R-1) mitdb-1	1	320	60	215/461	20k	100
	(R-2) mitdb-2	< 4	280	70	69/496	20k	100
	(R-3) mitdb800	< 4	95	25	24/165	20k	100
	(R-4) ptt-ppg	1	325	45	201/461	20k	100
	(R-5) refit	< 3	100	20	47/143	20k	100
	(R-6) arm-coda	5	525	105	272/886	8k	64
synthetic	(S-1) single	1	100	0	100/100	8k	100
	(S-2) fixed	5	100	0	100/100	3k	100
	(S-3) variable	5	150	30	100/200	4k	100

were randomly placed along the time axis for each subject, sensor, and dimension. The distance between two consecutive occurrences is sampled uniformly over $[50, 450]$. A Gaussian noise with a signal-to-noise ratio of 0.01 was added to all time series. This resulted in a dataset of 64 univariate time series.

B. Performance metrics

Motif discovery in time series is an unsupervised event detection task. Like other time series event-based tasks, we evaluate performance with precision, recall, and f1-score metrics [42]. However, compared to supervised tasks, the computation of these metrics requires the additional step of pairing real and predicted motif sets. This step is a two-level assignment problem: predicted motif sets must be assigned to real motif sets, and predicted occurrences must be assigned to real ones between paired motif sets. The optimal pairings maximize the total overlapping length between real and predicted motif sets, and they can be efficiently computed with the Hungarian matching algorithm [20], [36]. The precision, recall, and f1-score computation rely on the optimal pairings and a threshold $\tau \in [0, 1]$ that controls the overlapping ratio. Any metric's score is the average of the individual metric score between paired motif sets; the averaging can be macro or weighted. For precision (resp. recall), a motif occurrence is counted as a true positive if the ratio between the overlap length and the predicted (resp. real) occurrence length is greater than the threshold τ. This threshold is set to 50% for all experiments. The resolution of the motif sets assignment problem and the metrics' computation are well-detailed in supplementary materials.

We also rank methods according to the f1-score and compute critical difference diagrams [9]. The associated test significance level is set to 0.05. We use Friedman's test to reject the null hypothesis, and we compute the critical differences using Nemenyi post-hoc test.

C. State-of-the-art methods and implementation details

The evaluation was performed on a server with Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz, and 250 GB of RAM. We compared PEPA and A-PEPA with SetFinder (SF)[2], LatentMotif (LM)[18], Grammarviz (GM)[38], MDLC (MC)[34], STOMP (SM)[48], and VALMOD (VM)[23]. For fairness, we implemented all of the algorithms in Python except Grammarviz. Indeed, they all rely on a fast computation of the distance profiles, and we implemented a common structure based on the algorithms [48].

VALMOD algorithm efficiently computes all matrix profiles within a subsequence length range thanks to STOMP algorithm and a pruning strategy. Therefore, STOMP algorithm provides a lower bound of VALMOD scalability performances. For simplicity, we implemented a greedy version of VALMOD algorithm that does not consider the pruning strategy. Predicted motif sets remain identical, and we use STOMP algorithm as a lower bound for VALMOD scalability performance.

For Grammarviz, we used an implementation in JAVA provided by the authors [38].

V. Experimental Evaluation

Our experimental evaluation has four components:

1) A qualitative evaluation on physiological signals, (in Section V-A).
2) A quantitative comparison of PEPA and A-PEPA with 5 state-of-the-art algorithms on several labeled real and synthetic datasets (in Section V-B).
3) Several experiments to show the influence of the main parameters of PEPA and A-PEPA: the subsequence length, the number of nearest neighbors, and the persistence threshold heuristic for A-PEPA (in Section V-C).
4) A scalability experiment (in Section V-D).

A. Qualitative evaluation

In this section, we illustrate the visual interpretability of our algorithm and its ability to detect meaningful patterns in two types of physiological data.

ECG data. Electrocardiograms of patients suffering from premature ventricular contractions (PVCs) contain a typical pattern for normal heartbeats and another typical pattern for heartbeats with PVCs. Several ECGs in the mitdb800 database correspond to patients suffering from PVCs, and we ran the adaptive algorithm on a 16-second portion of one of them. We set the window length to 500ms and the number of neighbors to 5. The persistence diagram, Figure 7-A (left), suggests that two motif sets have been properly isolated with the birth and persistence thresholds. Figure 7-A (right) shows the motif sets; the first set corresponds to heartbeats with PVC, and the second corresponds to normal heartbeats. Figure 7-A (middle) shows that all heartbeats are detected and well classified except one normal heartbeat. Illustrations of motifs sets discovered with other motif discovery algorithms can be found in supplementary material.

EEG data. During the second stage of sleep, the brain activity slows down, except for short bursts of activity that help resist awakening by external stimuli. On an electroencephalogram (EEG), these short bursts of activity fall into two categories: the K-complexes and the sleep spindles [28]. A K-complex is the succession of high-voltage positive and negative peaks that last for about 600ms and occur every 1 or 2 minutes. Sleep spindles correspond to 11 to 16 Hz voltage oscillations and last for about 0.5 to 1.5 seconds. We ran the adaptive algorithm on the EEG of a patient in the second stage of sleep [28]. It is a single-channel EEG sampled at 100hz, and we selected a 500-second window. We set the window length to 1 second and the number of neighbors to 5. The persistence diagram, Figure 7-B (left), shows that the algorithm has detected two motif sets. The first motif set gathers K-complexes, and the second set corresponds to sleep spindles, Figure 7-B (right).

In both cases, the algorithm has detected patterns that account for the patients' physiological state. The persistence diagrams ensure the relevance of these patterns because they significantly detach from the rest of the time series.

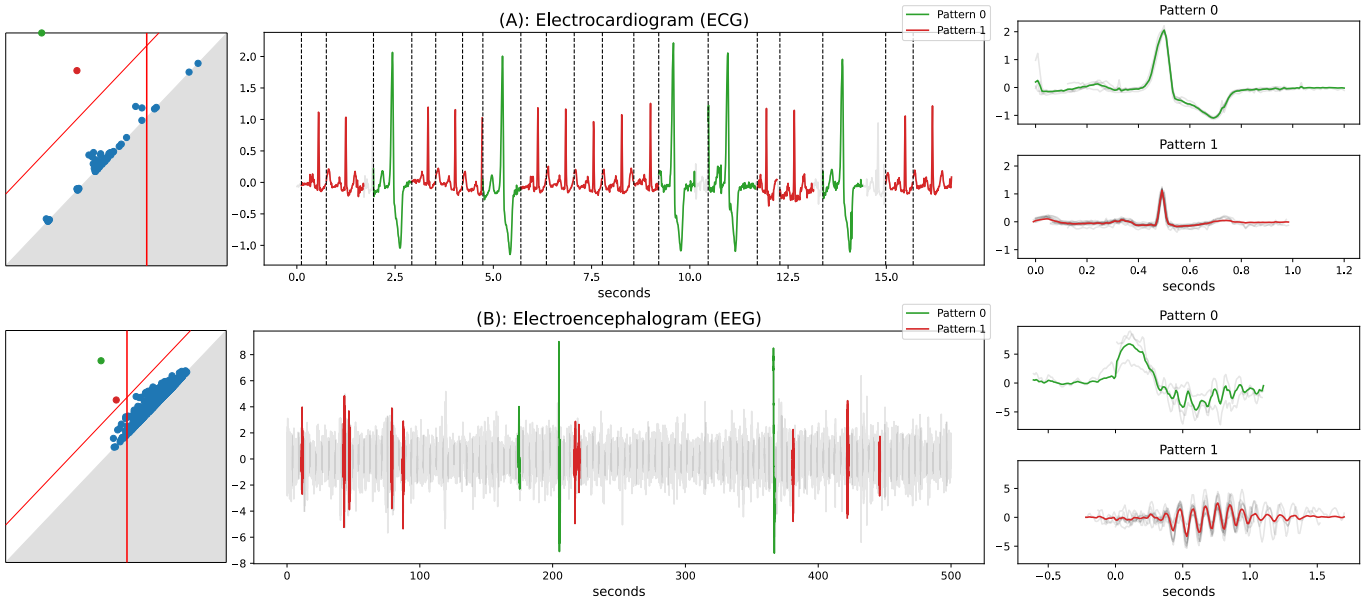


Fig. 7. (left): persistence diagrams, (middle): time series with colored motif sets, (right): motif sets with barycenters. **(A) Electrocardiogram:** ECG of a patient with premature ventricular contraction (PVC). The persistence diagram shows two significant motif sets; pattern 0 represents heartbeats with PVC, and pattern 1 represents normal heartbeats. Vertical dashed lines on the time series plot indicate the start location of the pattern occurrences. **(B) Electroencephalogram:** single-channel EEG of a patient in a second stage of sleep. The persistence diagram indicates two significant motif sets; pattern 0 represents K-complexes, and pattern 1 represents sleep spindles. Both patterns represent short bursts of brain activity that help resist awakening by external stimuli.

B. Comparison with state-of-the-art

In this experiment, we evaluate the performance of PEPA and A-PEPA with 5 state-of-the-art algorithms on two tasks of increasing complexity:

- **occurrence detection:** Ability to localize pattern occurrences regardless of their motif set membership.
- **motif set discovery:** Ability to localize pattern occurrences and classify them according to their motif set membership.

For PEPA and A-PEPA, the window length is set to the average pattern length minus its standard deviation, and the number of neighbors is set to 5 for each dataset. For SetFinder, LatentMotif, STOMP, and VALMOD, the window length is set to the average pattern length, and the radius is set with a gridsearch on each dataset. For Grammarviz, the window length is set to the average pattern length; the radius, the alphabet, and the word size are set with a gridsearch on each dataset. Parameters settings can be found in supplementary materials.

For the occurrence detection task, results are shown in Table III, and the critical difference diagram in Figure 8. We make several comments:

- PEPA and A-PEPA are the best-performing methods, with a mean rank significantly higher than other methods.
- Our approach has a relatively low f1-score on the refit data (0.31 only). However, it is still better than other methods by a margin. Motifs in refit are similar to square waves, and normalized Euclidean distances have difficulties fully recovering such patterns.
- On ptt-pgg, methods based on the Z-normalized distance have low recalls (0.43 at best), contrary to PEPA and A-PEPA, which have markedly higher recall (0.62 and 0.66) thanks to the LT-normalized distance. Indeed, motifs in PPG signals are significantly affected by the trend induced by subjects' motions.

For the motif set discovery task, results are shown in Table IV and Figure 9:

- Again, the mean rank of PEPA and A-PEPA are significantly better than other methods. As the number of motifs is known, PEPA performs better than A-PEPA. Therefore, if a good calibration of PEPA is possible, it should be preferred over the adaptive scheme.
- Overall, f1-scores are lower on the motif discovery task because pattern occurrences must be classified, not just localized.
- Unlike A-PEPA, MDLC performances drop significantly from occurrence detection to motif set discovery. MDLC groups detected occurrences in too many sub-clusters, whereas A-PEPA better estimates the number of motifs as depicted in Section V-C3.

C. Influence of the parameters

In this section, we evaluate the influence of three parameters: the window length, the number of neighbors, and the persistence threshold heuristic in A-PEPA.

1) Influence of the window length: Our approach is tested on dataset (S-2), where all patterns have the same length. We run each algorithm with the window length parameter ranging from 50% to 150% of the pattern length. All other parameters are identical to those defined previously. For VALMOD and MDLC, the minimum/maximum window lengths are 50%/150%.

The results are shown in Figure 10. PEPA performs better than other methods for all metrics and most window lengths.

TABLE III
OCCURRENCE DETECTION. SF: SETFINDER, GM: GRAMMARVIZ, LM: LATENTMOTIF, SM: STOMP, VM: VALMOD, MC: MDLC

dataset	algorithm / metric	SF	GM	LM	SM	VM	MC	PEPA	A-PEPA
(S-1) single	f1-score	**0.98**	0.07	0.27	0.71	0.86	0.85	0.96	0.97
	precision	**0.99**	0.69	0.59	0.98	0.97	0.81	0.96	0.98
	recall	**0.97**	0.04	0.19	0.58	0.78	0.89	0.95	0.96
(S-2) fixed	f1-score	0.49	0.20	0.50	0.82	0.82	0.66	**0.90**	0.89
	precision	0.59	0.57	0.68	0.82	0.78	0.60	0.94	**0.94**
	recall	0.47	0.13	0.41	0.84	0.85	0.74	**0.87**	0.85
(S-3) variable	f1-score	0.49	0.02	0.48	0.86	0.76	0.76	**0.95**	0.95
	precision	0.81	0.12	0.74	0.87	0.72	0.75	0.97	**0.97**
	recall	0.37	0.01	0.37	0.86	0.83	0.78	**0.94**	0.93
(R-1) mitdb-1	f1-score	0.34	0.01	0.11	0.58	0.67	**0.77**	0.71	0.75
	precision	0.78	0.20	0.96	**0.97**	0.95	0.91	0.91	0.92
	recall	0.28	0.01	0.06	0.46	0.64	**0.68**	0.60	0.67
(R-2) mitdb-2	f1-score	0.64	0.03	0.30	0.58	0.68	0.67	0.86	**0.87**
	precision	0.93	0.45	**0.97**	0.97	0.97	0.86	0.94	0.95
	recall	0.53	0.02	0.19	0.44	0.59	0.55	0.80	**0.80**
(R-3) mitdb800	f1-score	0.75	0.13	0.40	0.56	0.70	0.49	**0.89**	0.89
	precision	0.90	0.96	0.87	0.97	**0.98**	0.92	0.96	0.97
	recall	0.67	0.07	0.28	0.43	0.59	0.34	**0.84**	0.84
(R-4) ptt-ppg	f1-score	0.49	0.01	0.12	0.49	0.52	0.71	0.73	**0.75**
	precision	0.91	0.11	0.92	**0.97**	0.87	0.85	0.96	0.97
	recall	0.38	0.00	0.07	0.36	0.43	0.61	0.62	**0.66**
(R-5) refit	f1-score	0.07	0.14	0.12	0.00	0.01	0.07	**0.31**	0.29
	precision	0.06	0.21	0.17	0.00	0.01	0.05	**0.23**	0.22
	recall	0.17	0.19	0.11	0.02	0.04	0.14	**0.56**	0.51
(R-6) arm-coda	f1-score	0.25	0.00	0.44	0.51	0.28	0.54	**0.62**	0.59
	precision	0.24	0.02	0.54	0.47	0.45	0.50	**0.61**	0.59
	recall	0.38	0.01	0.41	0.58	0.28	0.63	**0.66**	0.62

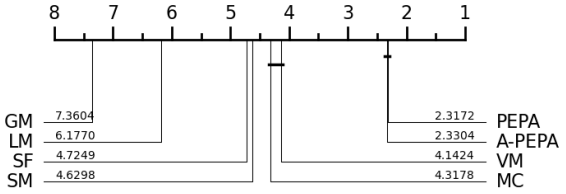


Fig. 8. Occurrence detection critical difference diagram. The rank is based on the f1-score. PEPA and A-PEPA perform significantly better than any other algorithm. Performances between PEPA and A-PEPA are not significantly different.

TABLE IV
MOTIF SET DISCOVERY. SF: SETFINDER, GM: GRAMMARVIZ, LM: LATENTMOTIF, SM: STOMP, VM: VALMOD, MC: MDLC

dataset	algorithm / metric	SF	GM	LM	SM	VM	MC	PEPA	A-PEPA
(S-1) single	f1-score	**0.98**	0.07	0.27	0.71	0.86	0.34	0.96	0.84
	precision	**0.99**	0.69	0.59	0.98	0.97	0.99	0.96	0.98
	recall	**0.97**	0.04	0.19	0.58	0.78	0.21	0.95	0.78
(S-2) fixed	f1-score	0.34	0.14	0.41	0.66	0.55	0.61	**0.84**	0.81
	precision	0.28	0.20	0.45	0.71	0.54	0.70	**0.86**	0.85
	recall	0.45	0.11	0.41	0.67	0.63	0.57	**0.85**	0.81
(S-3) variable	f1-score	0.32	0.02	0.39	0.72	0.43	0.75	**0.80**	0.80
	precision	0.31	0.02	0.49	0.74	0.42	**0.86**	0.81	0.82
	recall	0.34	0.01	0.37	0.75	0.54	0.71	**0.82**	0.81
(R-1) mitdb-1	f1-score	0.34	0.01	0.11	0.58	0.67	0.20	**0.71**	0.45
	precision	0.78	0.20	0.96	0.97	0.95	**0.98**	0.91	0.96
	recall	0.28	0.01	0.06	0.46	**0.64**	0.12	0.60	0.31
(R-2) mitdb-2	f1-score	0.49	0.02	0.24	0.41	0.51	0.31	**0.68**	0.59
	precision	0.66	0.31	0.73	0.73	0.77	**0.82**	0.75	0.78
	recall	0.44	0.01	0.17	0.32	0.47	0.24	**0.65**	0.53
(R-3) mitdb800	f1-score	0.35	0.06	0.23	0.25	0.33	0.08	**0.46**	0.41
	precision	0.42	0.53	0.49	0.51	0.52	**0.71**	0.50	0.53
	recall	0.34	0.04	0.19	0.22	0.31	0.05	**0.45**	0.38
(R-4) ptt-ppg	f1-score	0.49	0.01	0.12	0.49	0.52	0.19	**0.73**	0.50
	precision	0.91	0.11	0.92	0.97	0.87	**0.97**	0.96	0.96
	recall	0.38	0.00	0.07	0.36	0.43	0.11	**0.62**	0.37
(R-5) refit	f1-score	0.08	0.10	0.10	0.00	0.01	0.07	0.17	**0.20**
	precision	0.07	**0.20**	0.13	0.00	0.01	0.14	0.14	0.18
	recall	0.16	0.16	0.09	0.02	0.04	0.06	**0.35**	0.33
(R-6) arm-coda	f1-score	0.28	0.00	0.39	0.30	0.14	**0.53**	0.32	0.32
	precision	0.21	0.00	0.40	0.31	0.18	**0.55**	0.30	0.31
	recall	0.54	0.00	0.44	0.41	0.19	**0.63**	0.45	0.45

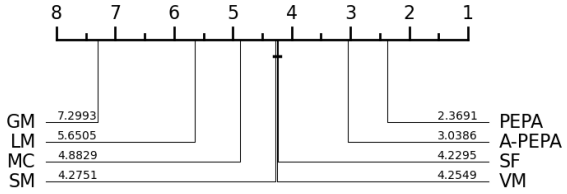


Fig. 9. Motif set discovery critical difference diagram. The rank is based on the f1-score. PEPA performs significantly better than any other algorithm. The second best performer is A-PEPA. Other algorithms perform significantly worse.

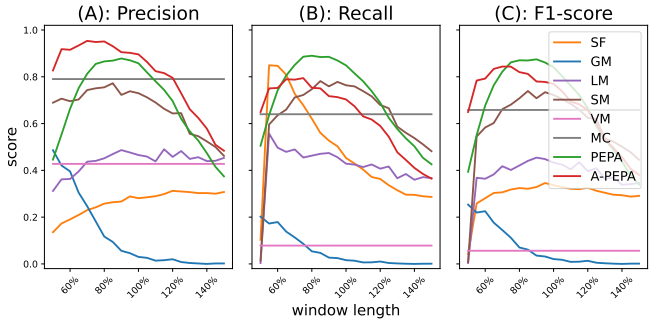


Fig. 10. Precision, recall, and f1-score of all algorithms as a function of window length. The experiment is run on the fixed dataset, and the window length is expressed as a percentage of the pattern length. The performance in f1-score of PEPA and A-PEPA is similar. They outperform all other algorithms in their best configuration over a wide range of window lengths.

Its f1-score is stable and close to the maximum for window lengths between 80% and 100%, proving its robustness to the window length parameter. A-PEPA has the best precision, but its recall drops as the number of motifs tends to get overestimated. However, its f1-score remains high and similar to PEPA, showing its robustness to the window length parameter. In light of this result, we recommend underestimating the length of the true patterns when using PEPA and A-PEPA.

2) Influence of the number of neighbors: Setting the number of neighbors is mandatory to compute the K-nearest neighbor graph (Section III-A). Theoretically, a larger number of neighbors leads to better performance but higher computation time and storage. In this experiment, we measured the relative errors between f1-scores obtained for different numbers of neighbors and the f1-scores obtained with the whole graph on all datasets. The number of neighbors ranges from 1 to 15 for PEPA and A-PEPA, and other parameters remain identical.

Results shown in Figure 11 prove that the number of neighbors has little influence on the performance of PEPA and A-PEPA. Regardless of the algorithm and for more than 5 neighbors, the average relative error does not exceed 1%, the average error is less than 0.05%, and the standard deviation is less than 0.2%. In practice, we recommend setting the number of neighbors to 5; it leads to good performance while maintaining low computational time and storage.

3) Influence of the persistence threshold heuristic in A-PEPA: In this experiment, we evaluate the ability of A-PEPA to detect the exact number of motif sets, and we compare its performances with the other adaptive method, MDLC, by measuring the error between the real and predicted number of motif sets on all datasets presented in Section IV-A. We set the persistence threshold heuristic of A-PEPA on the second-largest gap. It enforces A-PEPA to consider variations of the

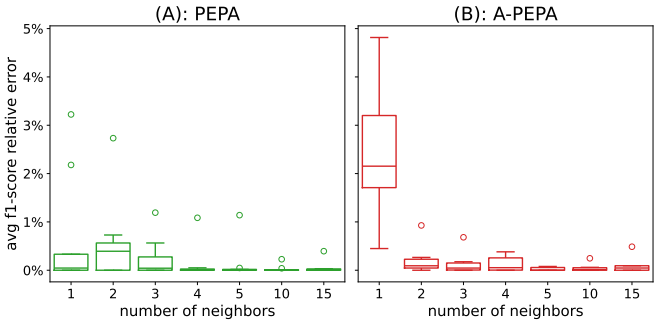


Fig. 11. Average f1-score relative error per dataset as a function of the number of neighbors for PEPA and A-PEPA and on all datasets. The baselines correspond to the scores obtained on the whole graph. For both algorithms, the relative error is less than 1% for more than 5 neighbors and never exceeds 9%.

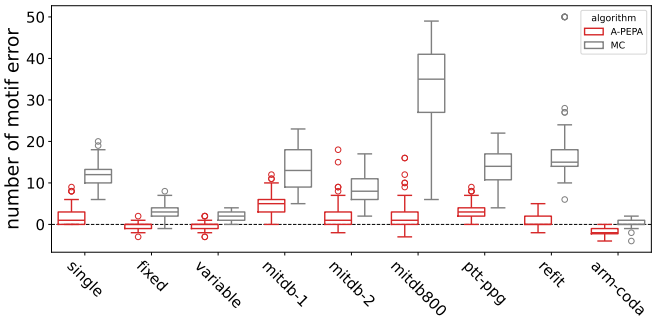


Fig. 12. Boxplots of the error in estimating the number of motif sets with A-PEPA for all datasets.

most persistent subgraphs as potential motif sets. The results are shown in Figure 12.

On all datasets except arm-coda (R-6), A-PEPA better estimates the number of motifs compared to MDLC. The average absolute mean error is 2.1 for A-PEPA and 12.0 for MDLC, with a standard deviation of 2.4 vs 10.8. It reflects the performance drop observed in Section V-B when MDLC clusters detected occurrences, whereas A-PEPA better retrieves the number of motif sets thanks to the persistent diagram.

Congruently with its heuristic settings (second most persistent gap), A-PEPA overestimates the number of motifs on single motif datasets: single (S-1), mitdb-1 (R-1), ptt-ppg (R-4). The number of motif sets is also greatly overestimated in rare cases for mitdb-2 (R-2), mitdb800 (R-3); the second gap leads to the inclusion of many sub-motif sets. However, the estimation is more accurate for datasets with a larger number of motif sets and shows less variability.

In practice, we recommend setting the persistence threshold heuristic to the largest or second largest gap. Alternatively, the relevance of the threshold can be verified with the persistence diagram, and recomputing of motif sets is done in $\mathcal{O}(n \log(n))$ in the worst case according to Section III-E.

D. Scalability

In this experiment, we evaluated the scalability with the time series length of PEPA and state-of-the-art algorithms.

We evaluated the algorithm runtime on a dataset consisting of synthetic time series of the following lengths: 10K, 50K, 100K, 500K, and 1M. We generated 10 time series for each length following the fixed scenario (S-2) and only modified

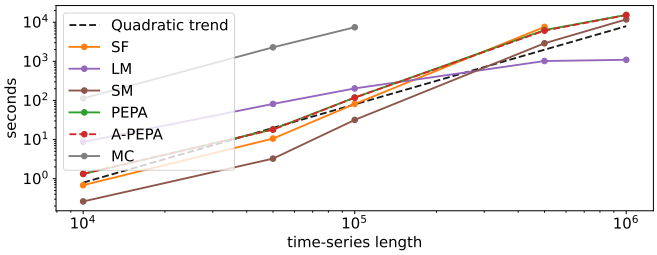


Fig. 13. Algorithms scalability with the length of the time series. SF: SetFinder, LM: LatentMotif, SM: STOMP.

the space between successive occurrences. We did not consider Grammarviz, since it is implemented in Java, and VALMOD, since we implemented a greedy version that does not consider its pruning strategy. Nevertheless, the performance of STOMP is a lower bound of the performance of VALMOD since it runs STOMP as an initialization. We considered the algorithms SetFinder, LatentMotif, STOMP, MDLC, PEPA, and A-PEPA. The parameters of each algorithm were identical to those defined in the benchmark for the fixed dataset. The timeout was set to 24 hours. The average runtimes per length are shown in Figure 13. MDLC is the worst time-performing algorithm and times out after 100K. SetFinder does not scale with the length of the time series and times out after 500K. On the other hand, LatentMotif is the fastest for large lengths (>300K), but it is slow for small lengths (<50K). This trend is due to the optimization scheme that limits the number of distance profiles computed. PEPA A-PEPA and STOMP scale according to their quadratic time. PEPA and A-PEPA performances are identical. STOMP is slightly faster than PEPA, but its advantage decreases as the length of the time series increases. Indeed, STOMP has to recompute some distance profiles to create the motif sets, while PEPA creates motif sets from the precomputed graph.

VI. Acknowledgments

This work was supported by grants from Région Ile-de-France (DIM MathInnov). Charles Truong is funded by the PhLAMES chair of ENS Paris-Saclay.

References

[1] https://github.com/thibaut-germain/Persistent-Pattern-Discovery.
[2] A. Bagnall, J. Hills, and J. Lines. Finding motif sets in time series. *arXiv preprint arXiv:1407.3685*, 2014.
[3] A. V. Benschoten, A. Ouyang, F. Bischoff, and T. Marrs. Mpa: a novel cross-language api for time series analysis. *Journal of Open Source Software*, 5(49):2179, 2020.
[4] A. Bois, B. Tervil, and L. Oudre. Persistence-based clustering with outlier-removing filtration. *Frontiers in Applied Mathematics and Statistics*, 10:1260828, 2024.
[5] J.-D. Boissonnat, F. Chazal, and M. Yvinec. *Geometric and topological inference*, volume 57. Cambridge University Press, 2018.
[6] Y. Chen, K. Chen, and M. A. Nascimento. Effective and efficient shape-based pattern detection over streaming time series. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):265–278, 2010.
[7] S. W. Combettes, P. Boniol, A. Mazarguil, D. Wang, D. Vaquero-Ramos, M. Chauveau, L. Oudre, N. Vayatis, P.-P. Vidal, A. Roren, and M.-M. Lefèvre-Colau. Arm-coda: A dataset of upper-limb human movement during routine examination. https://www.ipol.im/pub/pre/494/.
[8] D. De Paepe, D. N. Avendano, and S. Van Hoecke. Implications of z-normalization in the matrix profile. In *International Conference on Pattern Recognition Applications and Methods*, pages 95–118. Springer, 2019.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

[10] H. Edelsbrunner, J. Harer, et al. Persistent homology-a survey. *Contemporary mathematics*, 453(26):257–282, 2008.

[11] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34, 2012.

[12] T. Feng and S. S. Narayanan. Discovering optimal variable-length time series motifs in large-scale wearable recordings of human bio-behavioral signals. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7615–7619. IEEE, 2019.

[13] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[14] Y. Gao and J. Lin. Efficient discovery of time series motifs with large length range in million scale time series. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1213–1222. IEEE, 2017.

[15] Y. Gao and J. Lin. Hime: discovering variable-length motifs in large-scale time series. *Knowledge and Information Systems*, 61:513–542, 2019.

[16] T. Germain, C. Truong, and L. Oudre. Lt-normalized euclidean distance, a novel distance invariant to linear trend for time series data mining. http://www.laurentoudre.fr/publis/ICDE2024.pdf.

[17] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[18] J. Grabocka, N. Schilling, and L. Schmidt-Thieme. Latent time-series motifs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(1):1–20, 2016.

[19] S. D. Greenwald, R. S. Patil, and R. G. Mark. *Improved detection and classification of arrhythmias in noise-corrupted electrocardiograms using contextual information*. IEEE, 1990.

[20] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[21] N. K. Lee, F. L. Azizan, Y. S. Wong, and N. Omar. Deepfinder: An integration of feature-based and deep learning approach for dna motif discovery. *Biotechnology & Biotechnological Equipment*, 32(3):759–768, 2018.

[22] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15:107–144, 2007.

[23] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh. Matrix profile x: Valmod-scalable discovery of variable-length motifs in data series. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1053–1066, 2018.

[24] B. Liu, J. Li, C. Chen, W. Tan, Q. Chen, and M. Zhou. Efficient motif discovery for large-scale time series in healthcare. *IEEE Transactions on Industrial Informatics*, 11(3):583–590, 2015.

[25] J. Lonardi and P. Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

[26] P. Mehrgardt, M. Khushi, S. Poon, and A. Withana. Pulse transit time ppg dataset. *PhysioNet*, 10:e215–e220, 2022.

[27] G. B. Moody and R. G. Mark. The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, 20(3):45–50, 2001.

[28] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

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

[30] D. Murray, L. Stankovic, and V. Stankovic. An electrical load measurements dataset of united kingdom households from a two-year longitudinal study. *Scientific data*, 4(1):1–12, 2017.

[31] C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.

[32] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[33] C. S. Pun, S. X. Lee, and K. Xia. Persistent-homology-based machine learning: a survey and a comparative study. *Artificial Intelligence Review*, 55(7):5169–5213, 2022.

[34] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Mdl-based time series clustering. *Knowledge and information systems*, 33:371–399, 2012.

[35] L. Saclova, A. Nemcova, R. Smisek, L. Smital, M. Vitek, and M. Ronzhina. Reliable p wave detection in pathological ecg signals. *Scientific Reports*, 12(1):6589, 2022.

[36] S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11225–11234, 2021.

[37] P. Schäfer and U. Leser. Motiflets: Simple and accurate detection of motifs in time series. *Proceedings of the VLDB Endowment*, 16(4):725–737, 2022.

[38] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein. Grammarviz 3.0: Interactive discovery of variable-length time series patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(1):1–28, 2018.

[39] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner. Grammarviz 2.0: a tool for grammar-based pattern discovery in time series. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part III 14*, pages 468–472. Springer, 2014.

[40] H. Shao, M. Marwah, and N. Ramakrishnan. A temporal motif mining approach to unsupervised energy disaggregation: Applications to residential and commercial buildings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 1327–1333, 2013.

[41] H. Sivaraks, C. A. Ratanamahatana, et al. Robust and accurate anomaly detection in ecg artifacts using time series motif discovery. *Computational and mathematical methods in medicine*, 2015, 2015.

[42] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich. Precision and recall for time series. *Advances in neural information processing systems*, 31, 2018.

[43] S. Torkamani and V. Lohweg. Survey on time series motif discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2):e1199, 2017.

[44] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26:275–309, 2013.

[45] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. 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)*, pages 1317–1322. Ieee, 2016.

[46] Y. Zhang, F. Gan, and X. Chen. Motif difference field: An effective image-based time series classification and applications in machine malfunction detection. In *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, pages 3079–3083. IEEE, 2020.

[47] Y. Zhu, A. Mueen, and E. Keogh. Matrix profile ix: Admissible time series motif discovery with missing data. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2616–2626, 2019.

[48] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh. 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)*, pages 739–748. IEEE, 2016.

Thibaut Germain PhD student at Centre Borelli, a research lab of Ecole Normale Supérieure Paris-Saclay (France). Co-supervised by Charles Truong and Laurent Oudre, his PhD focuses on pattern recognition and representation in time-series.

Charles Truong Researcher at Centre Borelli, a reseach lab of Ecole Normale Supérieure Paris-Saclay (France). His research focuses on machine learning for time series with special attention to the problem of change point detection in multivariate signals.

Laurent Oudre Full professor at Centre Borelli, a research lab of Ecole Normale Supérieure Paris-Saclay (France). He leads a team of more than ten young researchers and has worked for about fifteen years on signal processing, pattern recognition, and machine learning for time series. His scientific projects mainly focus on AI applications in health and industry, often with a strong interdisciplinary component.

APPENDIX

Motif discovery in time series is an unsupervised event detection task. Like other time series event-based tasks, we evaluate performance with precision, recall, and f1-score metrics [42]. However, compared to supervised tasks, the computation of these metrics requires the additional step of pairing real and predicted motif sets. In what follows, we propose a resolution of the motif sets assignment problem and detail the metrics' computation.

A. Motif sets assignment problem

Pairing real and predicted motifs sets is a two-level assignment problem: predicted motif sets must be assigned to real motif sets, and predicted occurrences must be assigned to real ones between paired motif sets. We compute all pairings simultaneously by maximizing the total overlapping between real and predicted motif sets. Technically, let $R = (R_i)_{1 \leq i \leq |R|}$ the real motif sets such that $R_i = (R^s_{i,u}, R^e_{i,u})_{1 \leq u \leq |R_i|}$ is the list of starting and ending sample location of occurrences of the i^{th} motif. Likewise, we define the predicted motif sets $\left((P^s_{j,v}, P^e_{j,v})_{1 \leq v \leq |P_j|}\right)_{1 \leq j \leq |P|}$ and Σ_N the permutation group of the sequence $(1, \ldots, N)$. Note that we do not enforce the number of motif sets and occurrences to be identical between real and predicted labels. The total overlapping between real and predicted motif sets is defined by:

$$total\_overlapping(R, P) = \max_{(\sigma,\sigma') \in \Sigma_{|R|} \times \Sigma_{|P|}} \sum_{i=1}^{\min(|R|,|P|)} C(R_{\sigma(i)}, P_{\sigma'(i)}) \tag{5}$$

where:

$$C(R_i, P_j) = \max_{(\pi,\pi') \in \Sigma_{|R_i|} \times \Sigma_{|P_j|}} \sum_{u=1}^{\min(|R_i|,|P_j|)} overlap(R_{i,\pi(u)}, P_{j,\pi'(u)}) \tag{6}$$

and:

$$overlap(R_{i,u}, P_{j,v}) = \max(\min(R^e_{i,u}, P^e_{j,v}) - \max(R^s_{i,u}, P^s_{j,v}), 0) \tag{7}$$

Optimal pairings, $(\sigma, \sigma') \in \Sigma_{|R|} \times \Sigma_{|P|}$ and $\left\{(\pi_{i,j}, \pi'_{i,j}) \ | \exists u \ s.t \ (i,j) = (\sigma(u), \sigma'(u)), \ \pi_{i,j} \in \Sigma_{|R_i|}, \ \pi'_{i,j} \in \Sigma_{|P_i|}\right\}$, can be efficiently retrieved with the the Hungarian matching algorithm [20], [36].

B. Metrics computation

Precision, recall, and f1-score computations rely on the optimal pairings and a threshold $\tau \in [0, 1]$ that controls the overlapping ratio. The metrics average elementary metrics computed between paired motif sets; it can be a macro average with weights $w_i = 1/|R|$ or a weighted average with weights $w_i = |R_i| / \sum_{j=1}^{|R|} |R_j|$. In what follows, (σ, σ') is the optimal pairing between the motif sets of R and P, (π, π') is the optimal pairing between occurrences of R_i and P_j, and $\mathbb{1}$ is the indicator function.

1) Precision:

$$precision(R, P; \tau) = \sum_{i=1}^{\min(|R|,|P|)} w_{\sigma(i)} * elementary\_precision(R_{\sigma(i)}, P_{\sigma'(i)}; \tau)$$

$$elementary\_precision(R_i, P_j; \tau) = \frac{1}{|P_i|} \sum_{u=1}^{\min(|R_i|,|P_j|)} \mathbb{1}\left(overlap(R_{i,\pi(u)}, P_{j,\pi'(u)}) \geq \tau(P^e_{i,\pi'(u)} - P^s_{i,\pi'(u)})\right)$$

2) Recall:

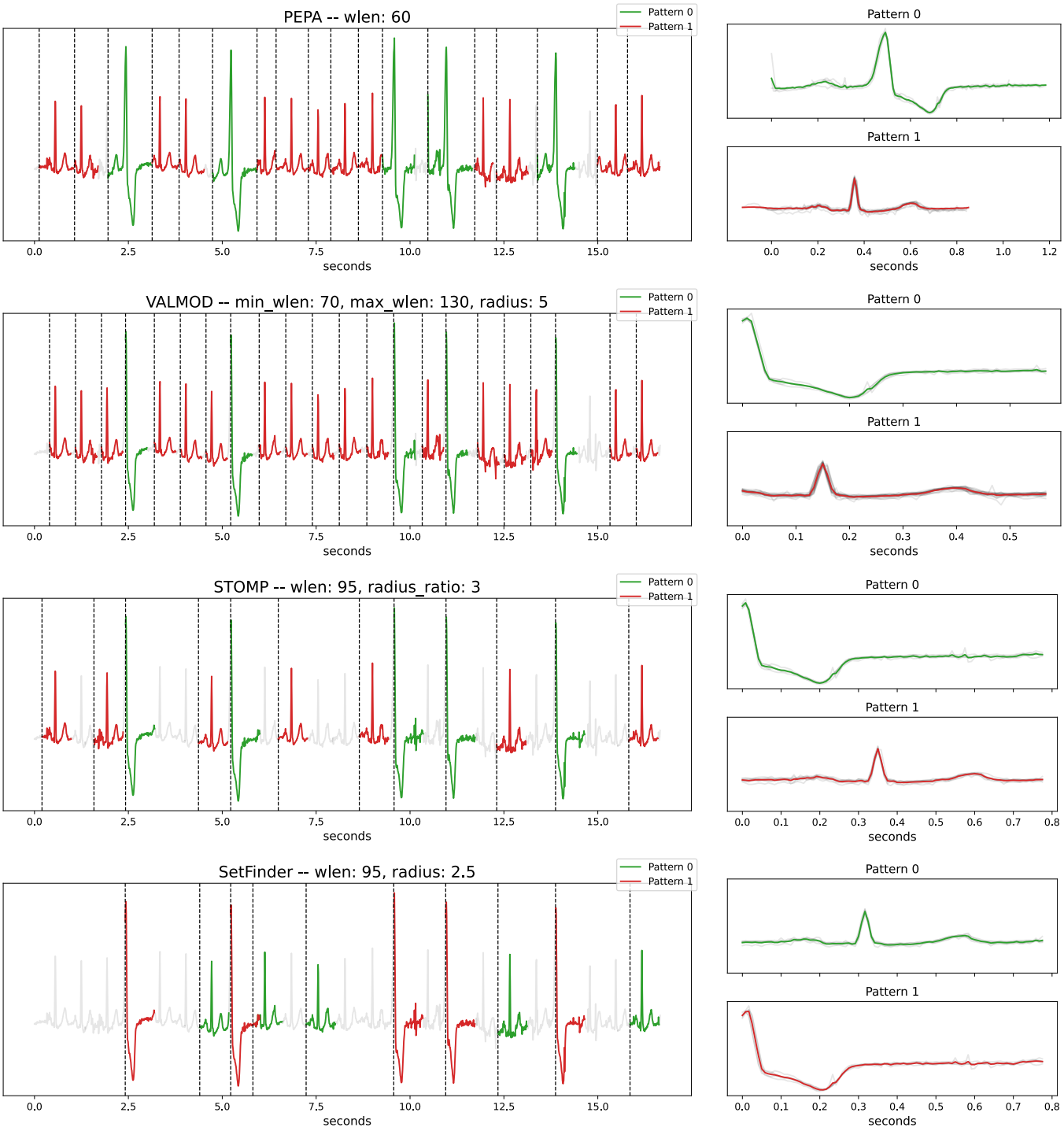
$$recall(R_i, P_j) = \sum_{i=1}^{\min(|R|,|P|)} w_{\sigma(i)} * elementary\_recall(R_{\sigma(i)}, P_{\sigma'(i)}; \tau)$$

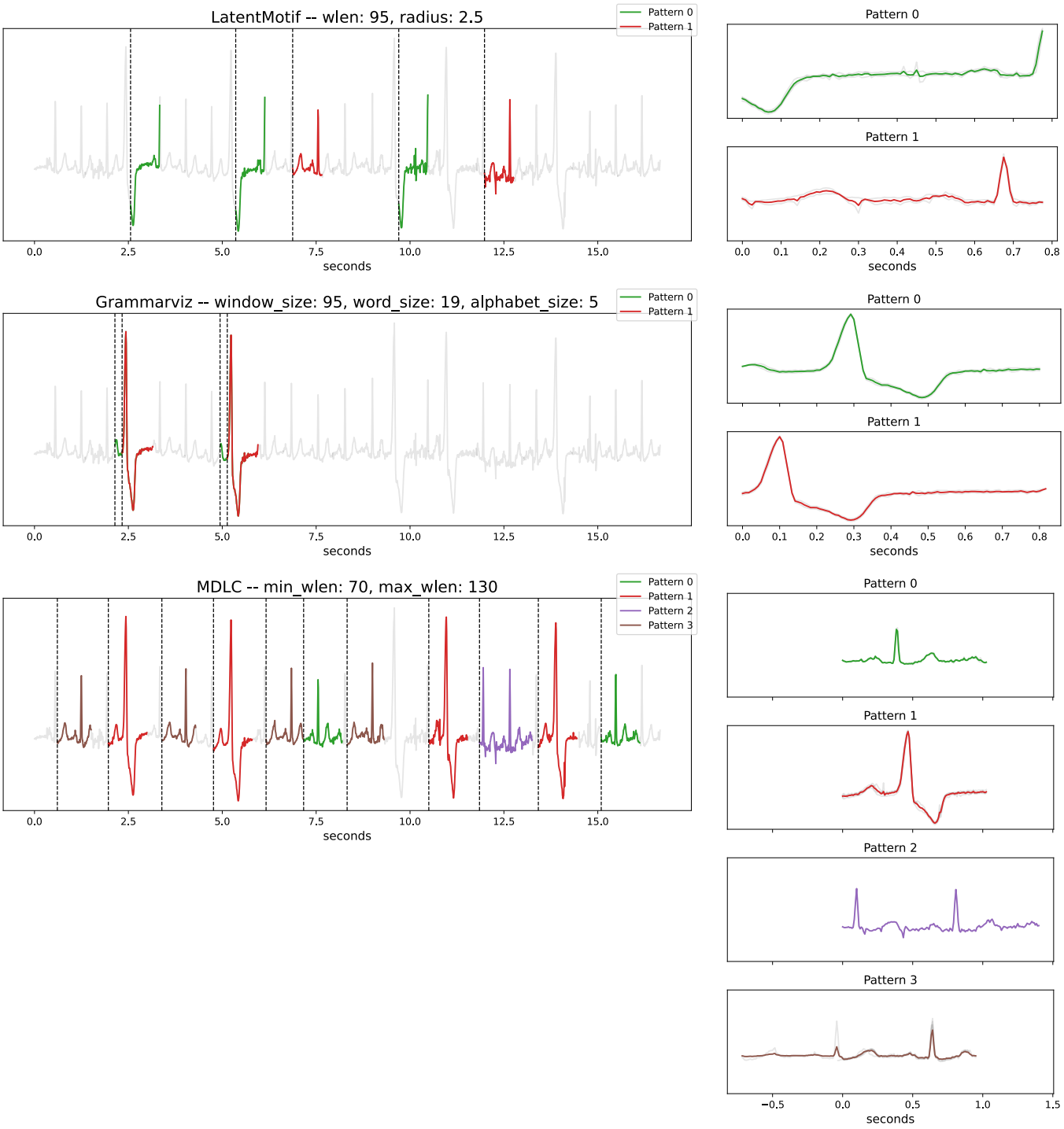
$$elementary\_recall(R_{\sigma(i)}, P_{\sigma'(i)}; \tau) = \frac{1}{|R_i|} \sum_{u=1}^{\min(|R_i|,|P_j|)} \mathbb{1}\left(overlap(R_{i,\pi(u)}, P_{j,\pi'(u)}) \geq \tau(R^e_{i,\pi'(u)} - R^s_{i,\pi'(u)})\right)$$

3) F1-score:

$$fscore(R, P; \tau) = \frac{2 * precision(R, P; \tau) * recall(R, P; \tau)}{precision(R, P; \tau) + recall(R, P; \tau)}$$

Electrocardiograms of patients suffering from premature ventricular contractions (PVCs) contain a typical pattern for normal heartbeats and another typical pattern for heartbeats with PVCs. Several ECGs in the mitdb800 [19] database correspond to patients suffering from PVCs, and we ran different motif discovery algorithms on a 16-second portion of one of them. Following figures present the motifs discovered for PEPA (our method), VALMOD [23], STOMP [48], SetFinder [2], LatentMotif [18], GrammarViz [38], and MDLC [34]. PEPA and VALMOD are the best-performing algorithms. They retrieve both motifs and all occurrences except one. However, compared to PEPA, VALMOD is not able to fully recover the motif associated with PVC.




LatentMotif -- wlen: 95, radius: 2.5
Pattern 0
Pattern 1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
seconds
Pattern 0
Pattern 1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
seconds
Grammarviz -- window_size: 95, word_size: 19, alphabet_size: 5
Pattern 0
Pattern 1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
seconds
Pattern 0
Pattern 1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
seconds
MDLC -- min_wlen: 70, max_wlen: 130
Pattern 0
Pattern 1
Pattern 2
Pattern 3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
seconds
Pattern 0
Pattern 1
Pattern 2
Pattern 3
−0.5
0.0
0.5
1.0
1.5
seconds

The following tables present the parameter settings of all methods involved in the comparative study (Section V.B). Methods are gathered by approach. For conciseness, only parameters, except the distance, that are modified across datasets are displayed; other parameters remain at their default settings and can be found on the Github repository [2].

C. Frequency-based algorithms

	SetFinder			LatentMotif			Grammarviz		
dataset	radius	wlen	distance	radius	wlen	distance	alphabet_size	window_size	word_size
mitdb800	30	95	Z	10	95	Z	4	95	19
ptt-ppg	5	320	Z	5	320	Z	5	320	64
variable	5	150	Z	5	150	Z	5	150	30
arm-coda	10	520	Z	10	520	Z	5	500	100
mitdb-1	5	320	Z	5	320	Z	5	320	80
single	5	100	Z	5	100	Z	5	100	10
refit	5	100	Z	5	100	Z	5	100	10
fixed	5	100	Z	5	100	Z	5	100	10
mitdb-2	5	280	Z	5	280	Z	5	280	70

D. Similarity-based algorithms

	STOMP			VALMOD				MDLC		
dataset	radius_ratio	wlen	distance	radius_ratio	min_wlen	max_wlen	distance	min_wlen	max_wlen	distance
mitdb800	3	95	Z	3	80	120	Z	80	120	Z
ptt-ppg	3	320	Z	3	260	401	Z	260	401	Z
variable	3	150	Z	3	100	201	Z	100	201	Z
arm-coda	2	520	Z	2	400	601	Z	400	601	Z
mitdb-1	3	320	Z	3	220	401	Z	220	401	Z
single	3	100	Z	3	95	105	Z	100	101	Z
refit	2	100	Z	2	80	121	Z	80	121	Z
fixed	2	100	Z	2	100	101	Z	100	101	Z
mitdb-2	3	280	Z	3	200	351	Z	200	351	Z

E. Persistence-based algorithms

	PEPA		A-PEPA	
dataset	wlen	distance	wlen	distance
mitdb800	60	LT	60	LT
ptt-ppg	260	LT	260	LT
variable	100	LT	100	LT
arm-coda	400	LT	400	LT
mitdb-1	220	LT	220	LT
single	100	LT	100	LT
refit	80	LT	80	LT
fixed	100	LT	100	LT
mitdb-2	180	LT	180	LT

[2]Github: https://github.com/thibaut-germain/Persistent-Pattern-Discovery