Enumerating Hub Motifs in Time Series Based on the Matrix Profile

Genta Yoshimura
yoshimura.genta@aist.go.jp
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan
Mitsubishi Electric Corporation
Kamakura, Japan

Atsunori Kanemura
atsu-kan@aist.go.jp
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan
LeapMind Inc.
Tokyo, Japan

Hideki Asoh
h.asoh@aist.go.jp
National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba, Japan

ABSTRACT

A time series motif is loosely defined as a subsequence in time series that occurs frequently, and the discovery of motifs often precedes analyzing time series data. Algorithms for motif discovery have been proposed with formal definitions of motifs, but they require manual setting of a radius parameter R, which defines the radius of motif clusters. Defining R assumes that subsequences belonging to the same motif are spherically distributed; however, such an assumption does not hold for many real-world datasets. Even if the assumption were correct, adjusting R is not easy, and requiring many trials and errors. In this paper, we propose a hub motif, another formal definition of motif that does not assume the radius parameter R, and develop an algorithm for finding hub motifs. The proposed algorithm, HubFinder, successfully enumerates significant motifs without exhaustive search of R by leveraging the matrix profile, a recently proposed data structure for time series. Experimental results show that HubFinder discovers useful motifs for a synthetic toy example and also for two real datasets from medicine and ergonomics.

CCS CONCEPTS

• **Computing methodologies** → **Motif discovery**; • **Information systems** → *Top-k retrieval in databases*; Summarization.

KEYWORDS

motif discovery, motif set discovery, motif enumeration, time series data mining, matrix profile

ACM Reference Format:
Genta Yoshimura, Atsunori Kanemura, and Hideki Asoh. 2018. Enumerating Hub Motifs in Time Series Based on the Matrix Profile. In *MileTS '19: 5th KDD Workshop on Mining and Learning from Time Series, August 5th, 2019, Anchorage, Alaska, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

MileTS '19, August 5th, 2019, Anchorage, Alaska, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Finding frequent patterns from time series is a fundamental technique for understanding and summarizing temporal structure in time series data. To this end, J. Lin et al. [5] have proposed *time series motif discovery*, which finds representative subsequences called motifs by selecting frequent patterns from all the subsequences extracted from time series. We focus on the *time series motif enumeration problem*, which enumerates multiple motifs in order of significance rather than finding a single motif, because in real applications time series usually include multiple patterns [1, 5, 9, 10].

There are at least two major issues in the existing time series motif enumeration methods (as described in more detail in Section 3.2):

(1) It is not easy to tune a radius parameter R, which defines the radius (size) of motif clusters, because an appropriate value is different for different datasets.
(2) Some datasets do not have a single radius that is appropriate for all the motif clusters, and the existing methods fail to enumerate motifs no matter how they tune the parameter R.

To solve these two issues, we defined a novel concept of time series motifs called *hub motifs* and proposed a motif enumeration method that does not assume the radius parameter R. Since the proposed method does not have the R parameter, we do not need to tune it like the existing methods [1, 5]. Moreover, it can find motifs from differently-sized motif clusters, where the existing methods often fail. The mechanism employed by the proposed method and how it is beneficial will be introduced with an illustrating example in Section 2.

Our main contributions are summarized as follows:

- We defined a novel concept of motifs, which we call *hub motifs* (Section 3.3). It eliminates the assumption that subsequences form clusters of the same radius R.
- We extended the matrix profile [12, 13] to introduce the *sub matrix profile* with an efficient algorithm for calculating it (Section 3.4). It enables us to compute the importance of each subsequence as hub motif efficiently.
- We proposed a novel motif enumeration algorithm, which we call *HubFinder* (Section 3.5). It efficiently enumerates hub motifs from time series without the parameter R based on the sub matrix profile.

- We demonstrated that HubFinder outperforms the existing methods both quantitatively and qualitatively on three datasets including real data (Section 4).

2 FUNDAMENTAL IDEA

We use Figure 1 to explain the fundamental idea behind the proposed method. Let us assume that representative subsequences in a time series are composed of two clusters, both of which are elliptically distributed (Figure 1(a)). The circles and triangles correspond to the subsequences of the first and second clusters, respectively.

The existing methods [1, 5] find motifs by fitting spheres of radius R without overlap (the orange spheres in Figure 1(b) and (c)). The spheres are centered at the time series motifs (the orange markers). As shown in Figure 1(b), the sphere includes not only circles but also triangles wrongly even with the smallest radius R_1 that covers all the subsequences from the first cluster. When R is set smaller, we need to use many spheres to cover a single cluster, which leads to the "discovery" of redundant motifs (Figure 1(c)). In addition, there are some uncovered subsequences (the blue markers) both in Figure 1(b) and (c).

The proposed method, HubFinder, finds motifs that are located in the center of each cluster (the orange marker in Figure 1(d)). We call the proposed motifs *hub motifs* because the extracted motifs behave like hubs in the subsequences space.

HubFinder works as described below. It first computes distances for all pairs of subsequences. Then it evaluates an importance of each subsequence under the assumption that the smaller a sum of the distances between a subsequence and the other subsequences is, the more significant the subsequence is as the hub motif. HubFinder finally enumerates subsequences as hub motifs in descending order of the importance, which is accomplished by a greedy algorithm.

In other words, HubFinder finds a set of motifs so as to minimize a sum of the distances between motifs and the other subsequences (the orange line segments in Figure 1(d)). In this manner HubFinder eliminates the need for the parameter R, and separates subsequences into distinct two clusters successfully as shown in Figure 1(d).

3 METHOD

3.1 Preliminaries

This subsection defines important data types used heavily later in this paper. A *time series* $\mathbf{X}$ is a sequence of V-dimensional real-valued vectors $\boldsymbol{x}_t$: $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{V \times T}$, where T is the length of $\mathbf{X}$.

A *subsequence* $\mathbf{X}_i$ in a time series $\mathbf{X}$ is a consecutive subset of $\mathbf{X}$ starting from position i: $\mathbf{X}_i = [\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+W-1}] \in \mathbb{R}^{V \times W}$ where W is the subsequence length. As will be described later, time series motif enumeration algorithms extract frequent local patterns in time series as subsequences.

The *all-subsequence set* $\mathcal{A}$ of a time series $\mathbf{X}$ is an ordered set of all possible subsequences obtained by sliding a window of length W along $\mathbf{X}$: $\mathcal{A} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$, where $N = T - W + 1$. Time series motif enumeration algorithms return a representative subset of $\mathcal{A}$ whose elements are typical local patterns in time series.

Next we introduce a metric defining a distance between each pair of subsequences in order to discover similarity between subsequences. We assume that the distance is measured by the Euclidean distance between z-normalized subsequences:

$$d(\mathbf{X}_i, \mathbf{X}_j) = \Big(\sum_{k=0}^{W-1} (\tilde{\boldsymbol{x}}_{i+k} - \tilde{\boldsymbol{x}}_{j+k})^2 \Big)^{1/2}, \tag{1}$$

where $\tilde{\mathbf{X}}_i = [\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_{i+1}, \ldots, \tilde{\boldsymbol{x}}_{i+W-1}]$ is a z-normalized subsequence, i.e. zero-mean and unit-variance. Although the Euclidean distance is one of the simplest distances, extensive empirical comparisons [2] have shown that the Euclidean distance is competitive with other measures on various applications.

A *distance profile* $\boldsymbol{D}_i \in \mathbb{R}^N$ is a vector of the z-normalized Euclidean distances between a given subsequence $\mathbf{X}_i$ and each subsequence in an all-subsequences set $\mathcal{A}$. By definition, the i-th location of $\boldsymbol{D}_i$ is zero, and very close to zero around this location. We avoid such "trivial matches" by ignoring an exclusion zone of length $\lfloor W/2 \rfloor$ before and after the location of the query subsequence. Then the distance profile $\boldsymbol{D}_i$ is redefined based on a *non-trivial-subsequences set* $\mathcal{A}_i = \mathcal{A} \setminus \mathcal{N}_i$ rather than the all-subsequences set $\mathcal{A}$, where $\mathcal{N}_i$ is a *trivial matches set*:

$$\mathcal{N}_i = \{\mathbf{X}_j \mid i - \lfloor W/2 \rfloor \leq j \leq i + \lfloor W/2 \rfloor\}. \tag{2}$$

A. Mueen et al. [11] have proposed the MASS algorithm, which calculates the distance profile in only $O(T \log T)$ time. It utilizes the classic Fast Fourier Transform (FFT) algorithm for efficient computation.

Recent works [12, 13] have introduced the matrix profile, a data structure that stores nearest neighbor information for every subsequence in a time series. It offers the solutions to various time series mining tasks, including motif discovery and enumeration. A *matrix profile* $\boldsymbol{P} \in \mathbb{R}^N$ is a vector of the z-normalized Euclidean distance between each subsequence $\mathbf{X}_i$ in the all-subsequences set $\mathcal{A}$ and its nearest neighbor in the non-trivial-subsequences set $\mathcal{A}_i$.

The i-th element in the matrix profile tells us the distance from subsequence $\mathbf{X}_i$ to its nearest neighbor excluding its trivial matches. However, it does not tell where that neighbor is located. This information is stored in a companion vector called the matrix profile index. A *matrix profile index* $\boldsymbol{I} \in \mathbb{Z}^N$ is a vector of integers, which stores the location of the nearest neighbor in $\mathcal{A}_i$ for each subsequence $\mathbf{X}_i$.

There are some efficient algorithms for calculating the matrix profile and matrix profile index such as STAMP [12] and STOMP/GPU-STOMP [13]. We later leverage the STOMP algorithm in order to enumerate representative motifs in time series efficiently.

3.2 Existing Motif Enumeration Methods

Before we introduce our proposed method, we review the existing two motif enumeration methods, SetFinder and ScanMK, which are used as baseline methods in our experiments. They are respectively based on the two motif definitions, range motif and closest-pair motif, which are introduced below.

Time series motifs are subsequences that occur frequently in a time series [5]. There have been proposed several definitions of the time series motifs [1, 5, 10], but they can be divided into two types of definitions: *range motif* and *closest-pair motif*. The difference of two definitions arises from how we regard a subsequence as significant. For example, [10] distinguishes the two definitions by

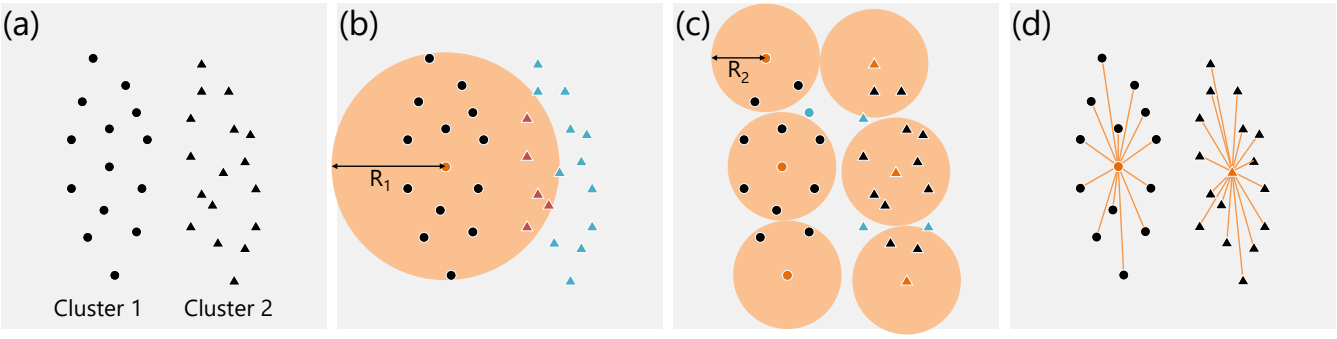

Figure 1: Sketch of the proposed method: (a) Scatter plot of subsequences from a time series. The circles represent subsequences belonging to the first cluster and the triangles are those belonging to the second cluster. (b) The existing methods with a large radius R_1. Some subsequences from another motif cluster (colored in red) are estimated to belong to a wrong cluster. (c) If we use a small radius R_2 ($< R_1$), the existing methods discover redundant motifs. (d) The proposed method finds the two motifs (colored in orange) that are located in the center of motif clusters successfully.

referring the former as the range motif (Definition 5) and the latter as the subsequence motif (Definition 7).

The definition of the range motif is based on the idea that a subsequence is significant if there exist many non-trivial-subsequences within the sphere of radius R. Mathematically, a subsequence $\mathbf{X}_i$ is more significant than $\mathbf{X}_k$ if and only if $|\mathcal{R}(\mathbf{X}_i, R)| > |\mathcal{R}(\mathbf{X}_k, R)|$, where a *range set*

$$\mathcal{R}(\mathbf{X}_i, R) = \{\mathbf{X}_j \in \mathcal{A}_i \mid d(\mathbf{X}_i, \mathbf{X}_j) \leq R\} \tag{3}$$

consists of non-trivial-subsequences whose z-normalized Euclidean distances from $\mathbf{X}_i$ are not greater than R. Then the *range motif* is obtained by

$$\arg\max_{\mathbf{X}_k \in \mathcal{A}} |\mathcal{R}(\mathbf{X}_k, R)|, \tag{4}$$

which corresponds to the most dense spherical region with radius R in subsequences space.

Unlike the range motif, the definition of the closest-pair motif is based on the idea that a subsequence is significant if the z-normalized Euclidean distance to its closest non-trivial-subsequence is small. More formally, a subsequence $\mathbf{X}_i$ is more significant than $\mathbf{X}_k$ if and only if $d(\mathbf{X}_i, \mathbf{X}_j) < d(\mathbf{X}_k, \mathbf{X}_l)$, for all $\mathbf{X}_j \in \mathcal{A}_i$ and $\mathbf{X}_l \in \mathcal{A}_k$. Therefore the *closest-pair motif* is obtained by

$$\arg\min_{\mathbf{X}_k \in \mathcal{A}} \min_{\mathbf{X}_l \in \mathcal{A}_k} d(\mathbf{X}_k, \mathbf{X}_l). \tag{5}$$

Next we move onto enumerating multiple motifs rather than finding a single motif. Although the above two types of motifs are defined as a single motif, they can be easily extended to multiple motifs using a sphere of radius R. *k-motif* $\mathbf{X}^{(k)} \in \mathcal{A}$ is the k-th most significant subsequence in time series $\mathbf{X}$, and the range set of k-motif is *k-motif cluster* $\mathcal{S}^{(k)} = \mathcal{R}(\mathbf{X}^{(k)}, R)$, which is also called *$k$-motif set* as in [1]. If we follow the range motif definition, k-motif is the subsequence which has the highest count of non-trivial matches and satisfies that k-motif cluster $\mathcal{S}^{(k)}$ does not overlap with the previous $(k-1)$ motif clusters, $\mathcal{S}^{(l)}$ for $1 \leq l < k$. This non-overlap condition can be written as $d(\mathbf{X}^{(k)}, \mathbf{X}^{(l)}) > 2R$ for $1 \leq l < k$.

All existing fixed-length motif enumeration methods [1, 5] utilize a sphere whose radius R is constant. They first find the 1-motif $\mathbf{X}^{(1)}$, the most significant motif among $\mathcal{A}$ according to each motif definition. Then, they iteratively find the k-motif $\mathbf{X}^{(k)}$, the k-th significant motif from *k-candidate set*

$$C^{(k)} = \bigcap_{l=1}^{k-1} \{\mathbf{X}_i \in \mathcal{A} \mid d(\mathbf{X}_i, \mathbf{X}^{(l)}) > 2R\} \tag{6}$$

until $C^{(k)}$ becomes empty. According to this manner, A. Bagnall et al. [1] proposed SetFinder and ScanMK, which are respectively based on the definition of range motif and closest-pair motif. The detail of SetFinder and ScanMK are described in Algorithm 3 in Appendix A.1 and Algorithm 4 in Appendix A.2 respectively.

The existing methods have two defects caused by the radius R. First, it is not easy to adjust the parameter R because the appropriate parameter changes in accordance with the target dataset. Unfortunately, most real applications of the motif discovery require the unsupervised setting where true annotations are not available. In this case we can not even know which R is appropriate, and therefore it is impossible to tune R.

Second, to make matters worse, there are cases where the existing methods fail to enumerate motifs successfully no matter how finely the parameter R is tuned. The motivating example in the introduction section (Figure 1) sketches this situation. Later in the experimental section, we will show that such a case can be easily made and actually occurs in real datasets.

In order to solve these two problems, we define a novel time series motif *hub motif*, and propose a novel motif enumeration algorithm *HubFinder*, which works without the parameter R in the following section.

3.3 Definition of Hub Motif

We first introduce an *anchor set* $\mathcal{H} \subset \mathcal{A}$, whose subsequences are located at the local minima of the matrix profile $\boldsymbol{P}$. Mathematically, a subsequence $\mathbf{X}_i$ is an *anchor*, i.e. $\mathbf{X}_i \in \mathcal{H}$, if and only if $\boldsymbol{P}[i] \leq \boldsymbol{P}[j]$ for all $j \in \mathcal{N}_i$. Each anchor is regarded as the candidate of motifs because the distance between an anchor and its closest subsequence is shorter than any distances between its trivial matches and their closest subsequences.

The definition of the hub motif is based on the idea that a subsequence is significant if a sum of z-normalized Euclidean distances to anchors is small. Mathematically, a subsequence $\mathbf{X}_i$ is more significant than $\mathbf{X}_k$ if and only if $\sum_{\mathbf{X}_j \in \mathcal{H}} d(\mathbf{X}_i, \mathbf{X}_j) < \sum_{\mathbf{X}_j \in \mathcal{H}} d(\mathbf{X}_k, \mathbf{X}_j)$. The *hub motif* is obtained by

$$\arg\min_{\mathbf{X}_k \in \mathcal{A}} \sum_{\mathbf{X}_j \in \mathcal{H}} d(\mathbf{X}_k, \mathbf{X}_j). \tag{7}$$

It is called "hub" motif because it is located in the central part of the anchors as in Figure 1(b).

Next, in order to enumerate multiple hub motifs without using the radius R, we define a *cost function* $f(\mathcal{B})$ for a set of subsequences $\mathcal{B} \subset \mathcal{A}$:

$$f(\mathcal{B}) = \sum_{\mathbf{X}_i \in \mathcal{H}} \min_{\mathbf{X}_j \in N_i, \mathbf{X}_k \in \mathcal{B}} d(\mathbf{X}_j, \mathbf{X}_k) \tag{8}$$

Our aim is to find a *set of K hub motifs* $\mathcal{B}^{(K)}$ which minimizes the cost function $f(\mathcal{B})$ over the anchor set $\mathcal{H}$:

$$\mathcal{B}^{(K)} = \arg\min_{\mathcal{B} \subset \mathcal{H} \text{ s.t. } |\mathcal{B}|=K} f(\mathcal{B}) \tag{9}$$

In this paper, the search space is restricted to $\mathcal{H}$ instead of $\mathcal{A}$, and the maximum number of motifs K is set to $K = W$.

3.4 Sub Matrix Profile

In order to compute the cost function (8) efficiently, we introduce a sub matrix profile. *Sub matrix profile* $\boldsymbol{P}^{(\mathcal{B})}$ is a vector of the z-normalized Euclidean distance between each subsequence $\mathbf{X}_i$ in a set of subsequences $\mathcal{B} \subset \mathcal{A}$ and its nearest neighbor in the non-trivial-subsequences $\mathcal{A}_i$.

The sub matrix profile can be computed efficiently along with a *sub matrix profile index* $\boldsymbol{I}^{(\mathcal{B})} \in \mathbb{Z}^N$, which stores the location of the nearest neighbor in $\mathcal{B}$ for each subsequence $\mathbf{X}_i$. Algorithm 1 shows how to compute $\boldsymbol{P}^{(\mathcal{B})}$ and $\boldsymbol{I}^{(\mathcal{B})}$ for a tentative set of motifs $\mathcal{B}$. ElementWiseMin compares two vectors $\boldsymbol{P}$ and $\boldsymbol{D}_i$, and returns a new vector containing the element-wise minima. This is almost the same as STAMP algorithm [12] except that it computes the matrix profile and matrix profile index on the subset $\mathcal{B} \subset \mathcal{A}$ instead of the all-subsequences set $\mathcal{A}$.

Algorithm 1 SubMatrixProfile

Input: Tentative set of motifs $\mathcal{B}$
Output:
Sub matrix profile $\boldsymbol{P}^{(\mathcal{B})}$
Sub matrix profile index $\boldsymbol{I}^{(\mathcal{B})}$
1: $\boldsymbol{P} \leftarrow \infty$
2: $\boldsymbol{I} \leftarrow \mathbf{0}$
3: **for all** $\mathbf{X}_i \in \mathcal{B}$ **do**
4: $\quad \boldsymbol{P}, \boldsymbol{I} \leftarrow \text{ElementWiseMin}(\boldsymbol{P}, \boldsymbol{I}, \boldsymbol{D}_i, i)$
5: **end for**
6: **return** $\boldsymbol{P}, \boldsymbol{I}$

Using the obtained sub matrix profile, the cost function (8) can be rewritten as

$$f(\mathcal{B}) = \sum_{\mathbf{X}_i \in \mathcal{H}} \min_{\mathbf{X}_j \in N_i} \boldsymbol{P}^{(\mathcal{B})}[j], \tag{10}$$

which can be computed efficiently than the original form.

In fact, the idea of the cost function and sub matrix profile was inspired by the fast convergence property of STAMP algorithm. As seen in Figure 5 of [12], the matrix profile converges very quickly to its true value with the random order updates. This suggests that only small fraction of the all-subsequences set $\mathcal{A}$ can approximate the true matrix profile. The optimization problem (9) aims to find a small subset of subsequences $\mathcal{B}$ whose sub matrix profile $\boldsymbol{P}^{(\mathcal{B})}$ is close to the matrix profile $\boldsymbol{P}$ around its local minima.

3.5 Proposed Method: HubFinder

For the hub motif definition, *k-motif* $\mathbf{X}^{(k)}$ $(k = 1, 2, \ldots, K)$ can be obtained by sorting the subsequences in $\mathcal{B}^{(K)}$ in order of significance. $\mathbf{X}^{(1)}$ corresponds to the most significant hub motif. *k-motif cluster* $\mathcal{S}^{(k)}$ is a set of subsequences which are related to the k-motif $\mathbf{X}^{(k)}$:

$$\mathcal{S}^{(k)} = \{\mathbf{X}_i \in \mathcal{H}^{(K)} \mid d(\mathbf{X}_i, \mathbf{X}^{(k)}) < \min_{j \neq k} d(\mathbf{X}_i, \mathbf{X}^{(j)})\}, \tag{11}$$

where $\mathcal{H}^{(K)} \subset \mathcal{A}$ corresponds to the local minima of the sub matrix profile $\boldsymbol{P}^{(\mathcal{B})}$ for $\mathcal{B} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}\}$. Each subsequence in $\mathcal{S}^{(k)}$ is more close to the k-motif $\mathbf{X}^{(k)}$ than the other motifs.

Our proposed method *HubFinder* (Algorithm 2) minimizes the cost function (10) in greedy manner to enumerate the hub motifs and hub motif clusters. It consists of three parts: (i) Refinement (ii) Sorting and (iii) Association. It first extracts K candidate subsequences of hub motifs, then sorts them in order of significance, and finally associates each motif with its motif cluster.

In the refinement part, the distance profile $\boldsymbol{D}_i$ is computed using STOMP-based algorithm for each subsequence $\mathbf{X}_i$, which is used for detecting the position of anchors $\mathcal{H}$ and the calculation of the sub matrix profile $\boldsymbol{P}^{(\mathcal{B})}$ and cost function $f(\mathcal{B})$. SlidingWindow($\mathbf{X}, W$) enumerates all-subsequences set $\mathcal{A}$ by sliding windows of length W. STOMPSTEP($\boldsymbol{D}_{j-1}, \mathbf{X}$) computes $\boldsymbol{D}_j$ from the previous distance profile $\boldsymbol{D}_{j-1}$, which corresponds to each iteration of STOMP algorithm. According to the line 14, subsequences located at local minima of the matrix profile are detected as anchors. Then each anchor $\mathbf{X}_i \in \mathcal{H}$ is appended to the tentative set of motifs $\mathcal{B}$. If the number of subsequences in $\mathcal{B}$ exceeds K, the most insignificant subsequence is removed from $\mathcal{B}$. These procedures refine the anchor set $\mathcal{H}$ to obtain K motifs efficiently.

In the sorting part, the refined K motifs $\mathbf{X}_i \in \mathcal{B}$ are sorted in order of significance. The most insignificant motif is removed from $\mathcal{B}$ and appended to the ordered list of motifs $\mathbf{M}$ iteratively. The first k motifs in $\mathbf{M}$ constitute $\mathcal{B}^{(k)}$ for $k \leq K$.

In the association part, K motifs are associated with motif clusters. The motif clusters consist of the local minima of the sub matrix profile $\boldsymbol{P}^{(\mathcal{B})}$. Each subsequence in motif clusters is related to the closest motif using the sub matrix profile index $\boldsymbol{I}^{(\mathcal{B})}$.

4 EXPERIMENTAL RESULTS

We introduce the evaluation metrics for the motif enumeration task, and then show experimental results with a synthetic toy example and two real datasets from medicine and ergonomics.

Algorithm 2 HubFinder

Input:
Time series $\mathbf{X}$
Subsequence length W
Number of motifs K
Output:
Ordered list of motifs $\mathbf{M}$
Ordered list of motif clusters $\mathcal{S}$

(i) Refine the anchors to extract motifs
$\mathcal{A} \leftarrow \text{SlidingWindow}(\mathbf{X}, W)$
$\mathcal{B} \leftarrow \emptyset$
$\mathcal{H} \leftarrow \emptyset$
$\boldsymbol{D}_1 \leftarrow \text{MASS}(\mathbf{X}_1, \mathbf{X})$
$\boldsymbol{P} \leftarrow \infty$
for all $\mathbf{X}_i \in \mathcal{A}$ **do**
  for all $\mathbf{X}_j \in \mathcal{N}_i$ **do**
    if $\boldsymbol{P}[j] = \infty$ **then**
      $\boldsymbol{D}_j \leftarrow \text{STOMPSTEP}(\boldsymbol{D}_{j-1}, \mathbf{X})$
      $\boldsymbol{P}[j] \leftarrow \min \boldsymbol{D}_j$
    end if
  end for
  if $\boldsymbol{P}[i] \leq \boldsymbol{P}[j]$ for all $j \in \mathcal{N}_i$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathbf{X}_i\}$
  end if
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{X}_i\}$
  if $|\mathcal{B}| > K$ **then**
    $\mathbf{X}_j \leftarrow \arg\min_{\mathbf{X}_k \in \mathcal{B}} f(\mathcal{B} \setminus \{\mathbf{X}_k\})$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\mathbf{X}_j\}$
  end if
end for
(ii) Sort the motifs in order of significance
for $k = 1$ to K **do**
  $\mathbf{X}_i \leftarrow \arg\min_{\mathbf{X}_j \in \mathcal{B}} f(\mathcal{B} \setminus \{\mathbf{X}_j\})$
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\mathbf{X}_i\}$
  $\mathbf{M}[K - k + 1] \leftarrow \mathbf{X}_i$
end for
(iii) Associate the motifs with motif clusters
$\mathcal{B} \leftarrow \{\mathbf{M}[1], \mathbf{M}[2], \ldots, \mathbf{M}[K]\}$
for $k = 1$ to K **do**
  $\mathcal{S}[k] \leftarrow \emptyset$
end for
for all $\mathbf{X}_i \in \mathcal{A}$ **do**
  if $\mathbf{X}_i = \arg\min_{\mathbf{X}_j \in \mathcal{N}_i} \boldsymbol{P}^{(\mathcal{B})}[j]$ **then**
    $k \leftarrow \boldsymbol{I}^{(\mathcal{B})}[i]$
    $\mathcal{S}[k] \leftarrow \mathcal{S}[k] \cup \{\mathbf{X}_i\}$
  end if
end for
return $\mathbf{M}, \mathcal{S}$

We implemented the proposed HubFinder as well as the existing methods, SetFinder and ScanMK. Python codes for all experiments are available at https://github.com/intellygenta/HubFinder.

4.1 Evaluation metrics

The motif enumeration task is similar to the clustering task to some extent. Their objectives are to find representative patterns within a dataset in an unsupervised manner. Thus we adopt a metric for the clustering task. Purity is one of the most popular metrics for the clustering task:

$$\text{Purity}(\Omega, \Psi) = \frac{1}{\sum_j |\psi_j|} \sum_k \max_j |\omega_k \cap \psi_j|, \tag{12}$$

where $\Omega = (\omega_1, \omega_2, \ldots, \omega_K)$ is the set of clusters and $\Psi = (\psi_1, \psi_2, \ldots, \psi_L)$ is the set of actual clusters.

In our problem setting, both of the sets are the subset of the all-subsequence set, i.e. $\omega_k \subset \mathcal{A}$ and $\psi_l \subset \mathcal{A}$. The set of clusters ω_k is written as $\omega_k = \{\mathbf{X}_i \in \mathcal{A} \mid |i-j| < W, |i-j| < \min_{\mathbf{X}_{j'} \in \mathcal{S}^{k'}, k' \neq k} |i-j'|), \mathbf{X}_j \in \mathcal{S}^{(k)}\}$. In other words, a subsequence $\mathbf{X}_i$ does not belong to any set of clusters if it does not overlap with any subsequence in motif clusters. Otherwise it belongs to k-th cluster ω_k if its most overlapping subsequence in motif clusters is one of $\mathcal{S}^{(k)}$.

4.2 Synthetic data

We prepared two subsequences of length $W = 32$: the single-period triangular and sine waves. Both subsequences were z-normalized and synthesized into a univariate time series $\mathbf{X}$. These two subsequences were arranged alternately as shown in Figure 2. *i.i.d.* Gaussian noise $\mathcal{N}(0, \sigma_1)$ and $\mathcal{N}(0, \sigma_2)$ were added on the triangular and sine subsequences respectively. We set $\sigma_2 = 0.1$ and $\sigma_1 = \sigma_2/2 = 0.05$, and left space of length $W/2$ between subsequences, which were added *i.i.d.* Gaussian noise $\mathcal{N}(0, 1)$. Finally the total length of the synthesized time series became $T = 9616$ by repeating 100 times for both subsequences.

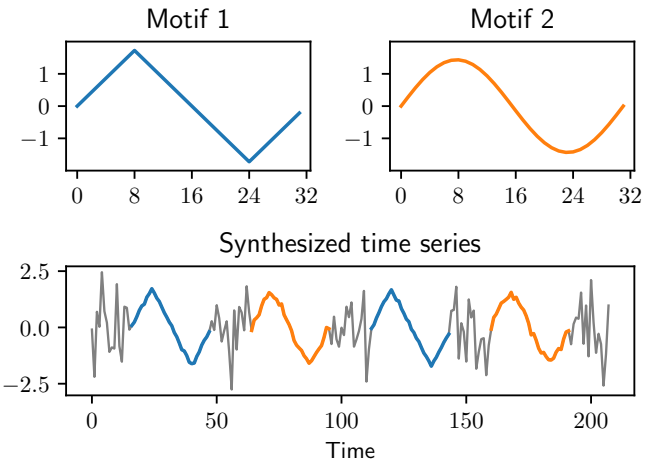

Figure 2: Two motifs and synthesized time series. (Top) The blue and orange subsequences are the first and second subsequences respectively. (Bottom) An example of the synthesized time series. The blue and orange colored subsequences show the true position of the motifs.

ScanMK, SetFinder, and HubFinder were used for enumerating two motifs in the synthesized time series. The parameter of motif

length W was set to its actual value $W = 32$ for each method. The radius R for ScanMK and SetFinder was found from the candidates $\{0.01, 0.02, \ldots, 2.00\}$ because it was not known in advance. The number of motifs K for HubFinder was set to $K = W = 32$, but arbitrary $k < K$ motifs can be obtained because it returns an ordered list of motifs which is sorted in order of significance.

Let us compare the purity between three methods. Figure 3 shows that HubFinder achieves 100% purity and outperforms ScanMK and SetFinder when the number of motifs is fixed to the actual value $K = 2$. The maximum purity of ScanMK is 58.5% for $R^* = 0.96$, and that of SetFinder is 69% for $R^* = 0.83$. This means that the existing methods can not achieve high purity no matter how finely you tune the parameter R. Furthermore, the existing methods seem to be sensitive to the parameter R and the "sweet spot" is restricted to a narrow range of R.

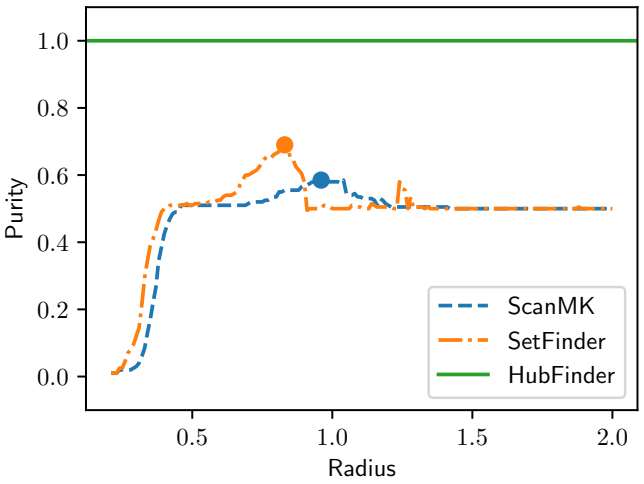

Figure 3: Dependency of the purity on the radius R for $K = 2$. The purity for HubFinder does not depend on R because it does not need the parameter R. The optimal radii for ScanMK and SetFinder are represented as circle markers.

Figure 4 shows that HubFinder achieves 100% purity for all $K \geq 2$. In contrast, although the purities for ScanMK and SetFinder grow as K increases, they can never reach 100%. The performance curves of the existing methods are interrupted in the middle because their algorithms terminate there. Their break points differ depending on R, which also implies that the existing methods are sensitive to the radius R.

Extracted motifs can be checked in Figure 5. Each methods successfully finds the triangular motif and its motif cluster (blue subsequences). However, the existing methods fail to extract the sine motif cluster, while HubFinder succeeds in capturing all of them (orange subsequences).

Figure 6 shows the multi-dimentional scaling (MDS) plot, which visualizes the similarity of the z-normalized subsequences of two clusters in two dimensional subsequence space. The triangle and circle markers represents the triangular and sine subsequences respectively. It can be seen from the upper left figure that the motif subsequences form two clusters. The variation of the sine

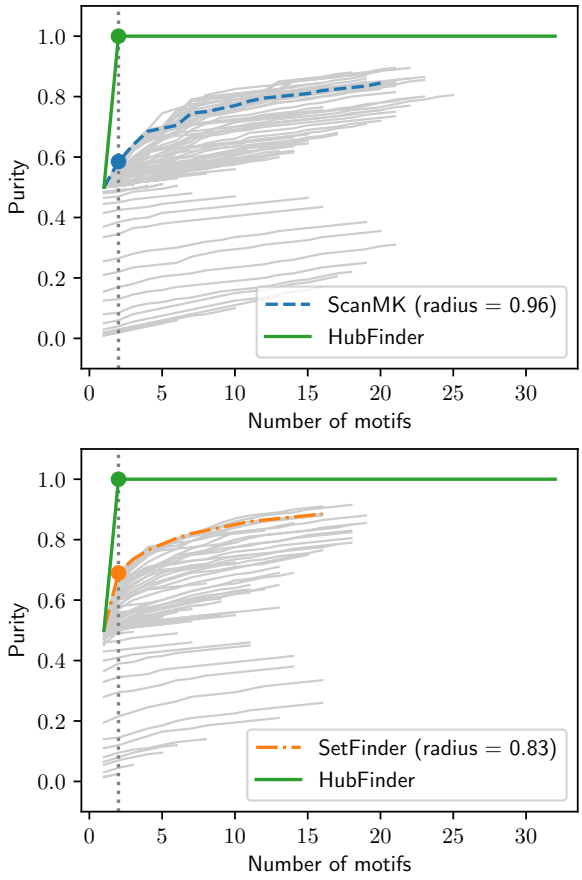

Figure 4: Dependency of the purity on K. The actual number of motifs is depicted as the vertical dotted line at $K = 2$. HubFinder's result is expressed as the green line for both figures. (Upper) ScanMK's purity for the optimal radius $R^* = 0.96$ is expressed as the blue dashed line, while the results for other radii are depicted as gray lines. (Lower) SetFinder's purity for the optimal radius $R^* = 0.83$ is expressed as the orange dotted-dashed line.

cluster is twice as large as the triangular one as expected from the experimental setting, $\sigma_2 = 2\sigma_1$.

For all methods each motif cluster is colored with the same color, while unallocated subsequences are uncolored. The upper right figure shows that HubFinder with $K = 2$ succeeds in finding two motif clusters perfectly, whereas the lower two figures show that ScanMK and SetFinder fail. When the parameter R is set to the optimal value R^* for the existing methods, they find a motif cluster for the triangular subsequences (blue markers) as in the lower figures, however, it also includes some sine subsequences.

In addition, motif clusters for the sine subsequences (colored markers other than blue markers) are too small to capture the entire cluster of the sine motif. This is because two clusters have different variation and no fixed radius can capture both clusters at the same time. The existing methods can never deal with such datasets because they are built on the assumption that each motif

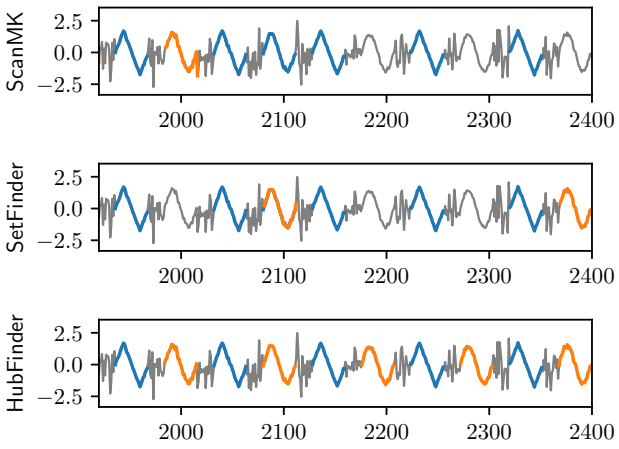

Figure 5: Visualization of the extracted motifs and motif clusters. The gray line is the original time series. The blue subsequences belong to 1-motif or its motif cluster, while the orange ones correspond to 2-motif.

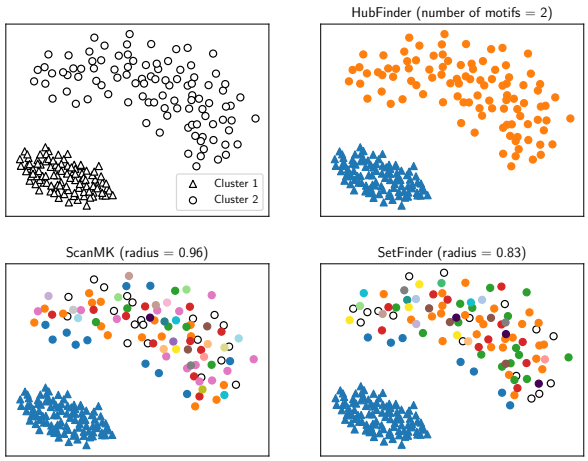

Figure 6: MDS plot of the z-normalized subsequences: (Upper left) The triangle and sine subsequences are plotted as the triangle and circle markers respectively. (Upper right) HubFinder with $K = 2$. (Lower left) ScanMK with the optimal radius $R^* = 0.96$. (Lower Right) SetFinder with the optimal radius $R^* = 0.83$.

cluster has the same radius. Although those findings are based on a synthetic dataset, differently-sized clusters often appear in real datasets, and a practical motif discovery needs to deal with it.

The running times of three methods are depicted in Figure 7. The number of repetitions for the triangular and sine subsequences was changed between $\{100, 200, \ldots, 1000\}$, which leads to the time series length $T = \{9616, 19216, \ldots, 96016\}$. The average of 10 trials for each length is shown in the figure. The experiments were performed on an Intel Core i7-8700K CPU with 3.70GHz and 64GB memory.

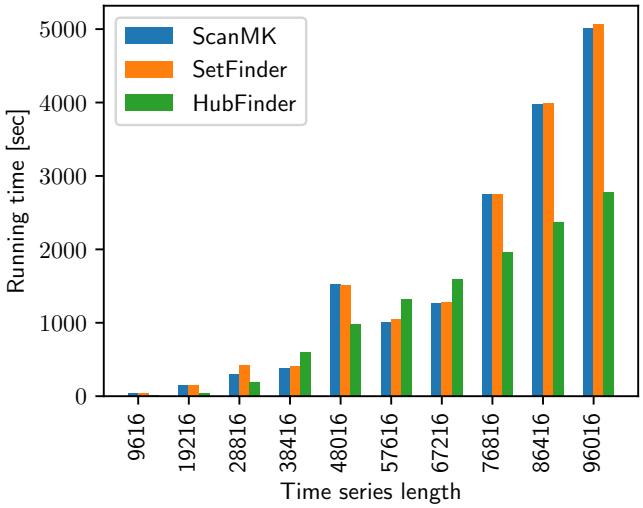

Figure 7: Running times on the synthetic time series.

It turned out that HubFinder is faster than the existing methods for large time series lengths. This is due to the time complexity of HubFinder, which is $O(T^2)$ with the aid of STOMP-based computation, while that of the existing methods unstably grow with $O(T^2 \log T)$. In addition, unlike the existing methods, HubFinder does not need multiple trials for tuning the parameter R, which reduces the total running time in real situations.

4.3 ECG data

MIT-BIH Arrhythmia Database[1] contains 48 half-hour excerpts of two-channel ambulatory ECG recordings [8], which is published on the PhysioNet [4].

We selected subject 106 because there are many premature ventricular contractions (PVCs) in her data. According to the annotations, it has 1507 normal beats and 520 PVCs. The upper signal, i.e. the modified limb lead II (MLII) was used as a univariate input time series. The motif length was set to $W = 128$ so as to capture the broad QRS complex and the lack of P wave, which are the charasteristics of PVCs. The radius parameter for ScanMK and SetFinder was changed among $R = \{0.1, 0.2, \ldots, 10.0\}$.

Figure 8 is the purity evaluation results for three methods, when the number of motifs is fixed to its actual number $K = 2$. It is clear that HubFinder outperforms the existing methods for each radius R: the purity of HubFinder is 99.4%, which is better than the purity 83.4% of SetFinder with the optimal radius $R^* = 8.2$, and 92.6% of ScanMK with $R^* = 6.4$. Moreover, the existing methods are sensitive to the parameter R as in the case with the synthetic data (Figure 3).

[1] https://www.physionet.org/physiobank/database/mitdb/

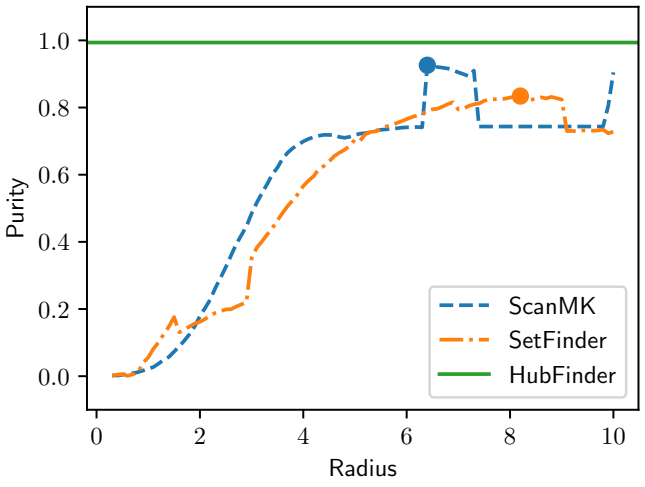

Figure 8: Dependency of the purity on the radius R for $K = 2$.

4.4 Human motion data

MotionSense Dataset[2] includes three dimensional accelerometer time series of human motion [7]. A total of 24 subjects performed six activities: downstairs, upstairs, walking, jogging, sitting, and standing.

We applied ScanMK, SetFinder, and HubFinder for each subject to find motifs and motif clusters in the human motion. Downstairs, upstairs, walking, and jogging activities were chosen among the six activities because the other two activities may have less movement than these four activities. That is, the trial number 11, 12, 15 and 16 were selected and concatenated into a three dimensional time series for each subject. The motif length was set to $W = 64$ and the radius for ScanMK and SetFinder was changed among $R = \{0.1, 0.2, \ldots, 20.0\}$. For each subject the purity was computed for $K = 4$, which is the true number of the activities.

Figure 9 shows that HubFinder outperforms the existing methods, where almost all markers are plotted on the top-left triangle region except subject 5 for SetFinder.

We elaborate on the result for subject 5, where HubFinder loses to SetFinder when $K = 4$. According to Figure 10, although the purity of HubFinder for $K = 4$ is less than one of SetFinder, it finally exceeds for $K > 5$. This may imply that she has more variation in her action and the number of motifs $K = 4$ is not enough for her.

Figure 11 shows the position of the extracted motifs and motif clusters of subject 23 for three methods. The existing methods extract two motifs from the same activity, while the proposed method successfully finds motifs from all four activities. In addition, it is interesting to note that HubFinder seems to capture the walking activities at landings of stairs because the motif for the walking activity also appears in the downstairs and upstairs activities.

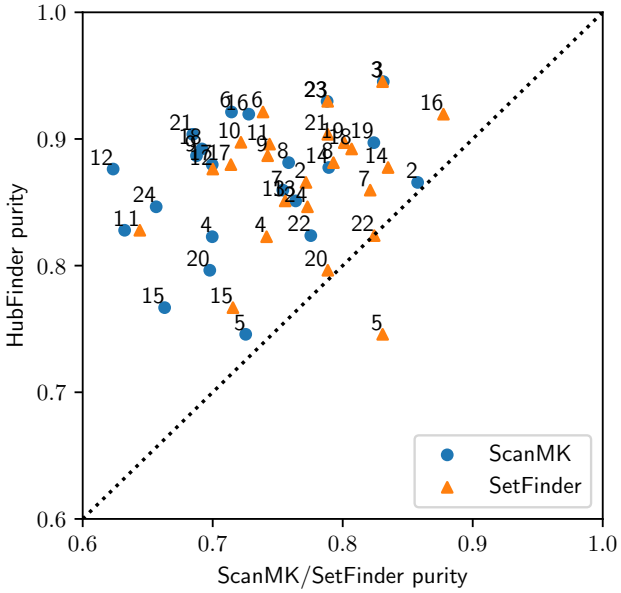

Figure 9: Comparison of the purity between the existing methods and HubFinder for $K = 4$. X-axis and y-axis are the purity of the existing methods and HubFinder respectively. The numbers nearby markers represent the number of subjects.

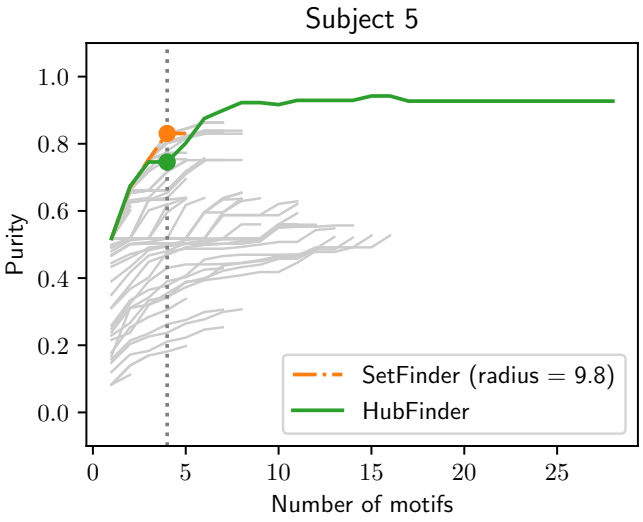

Figure 10: Dependency of the purity of Subject 5 on the number of motifs.

5 DISCUSSION AND CONCLUSION

We proposed hub motifs and developed an algorithm for finding hub motifs, which gets rid of the radius parameter R in order to solve the two issues in the existing motif enumeration methods (shown in Section 1). Experimental results (such as Figures 3, 8, and 9) show that the proposed HubFinder outperforms the existing methods for all three experiments in terms of the quantitative evaluation metrics of the purity. Unlike the existing methods, HubFinder enumerates useful motifs for a synthetic toy example and also for two real datasets without exhaustive search of R.

[2]https://github.com/mmalekzadeh/motion-sense/

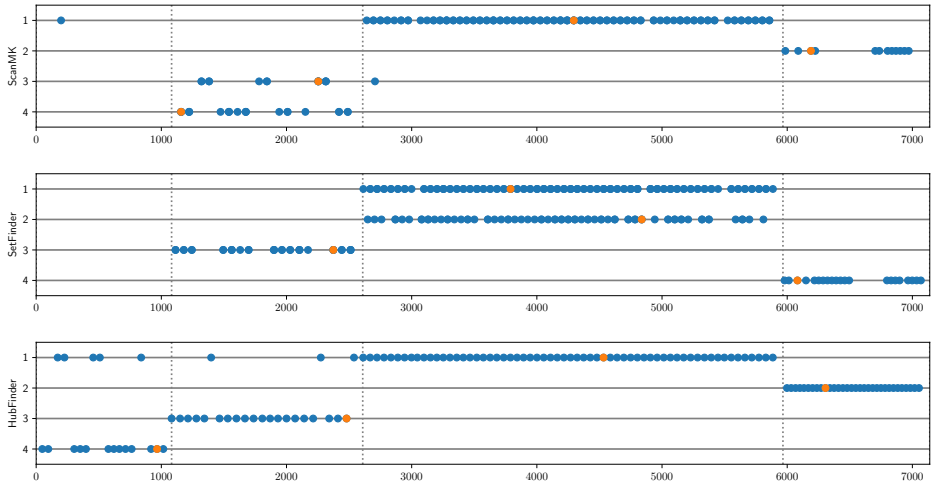

Figure 11: Extracted motifs and motif clusters of subject 23 for three methods. The vertical dotted lines separate different activities. The downstairs, upstairs, walking, and jogging activities are lined up from left to right. The horizontal lines represent the extracted motifs, on which the position of motifs and motif clusters are depicted using the orange and blue circles respectively.

Not only finding useful motifs without R, HubFinder has many other advantages: HubFinder is scalable to time series length T because its time complexity is $O(T^2)$. This is better than the existing methods, whose time complexity is $O(T^2 \log T)$. In addition, HubFinder is adaptive not only to univariate time series but also to multivariate ones, as shown in Section 4.4. These properties are useful in real applications because real time series often have large length T or multiple sensors ($V > 1$). Furthermore, HubFinder is incremental, which means that the tentative set of motifs $\mathcal{B}$ is updated in incremental manner as shown in Algorithm 2. Therefore, it can deal with the streaming data, which is also often the case with real time series.

Finally we discuss future directions. HubFinder asks us to tune the motif length parameter W. There are some motif discovery methods which finds variable length motifs [3, 6, 9]. HubFinder may be able to enumerate variable length motifs instead of the fixed length W by applying the concepts of such methods.

Another direction is to utilize extracted motifs for other time series mining tasks such as classification, forecasting, clustering, segmentation, and anomaly detection. Various features can be extracted from the obtained hub motifs, e.g. the number of occurrences of each motif, the sequential order of motifs, the variation from motif, and so on. We think these features are more abstract and understandable than time series signal itself, therefore machine learning models trained with them can be more generalized and interpretable than the existing models.

ACKNOWLEDGMENTS

We are grateful to Dr.Keogh for providing access to the codes and papers related to the matrix profile. We wish thank Dr.Mueen for publishing the MASS code. We also would like to thank G.B. Moody, R.G. Mark, M. Malekzadeh, R.G. Clegg, A. Cavallaro, and H. Haddadi for providing valuable datasets, MIT-BIH Arrhythmia Database and MotionSense Dataset. This study was supported in part by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

REFERENCES

[1] Anthony Bagnall, Jon Hills, and Jason Lines. 2014. Finding Motif Sets in Time Series. *arXiv preprint arXiv:1407.3685* (2014).
[2] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.
[3] Yifeng Gao and Jessica Lin. 2018. Efficient Discovery of Variable-length Time Series Motifs with Large Length Range in Million Scale Time Series. *arXiv preprint arXiv:1802.04883* (2018).
[4] Ary L. Goldberger, Luis A. N. Amaral, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, and H. Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101, 23 (2000), e215–e220.
[5] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. 2002. Finding Motifs in Time Series. In *SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 53–68.
[6] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. 2018. Matrix profile X: VALMOD—Scalable discovery of variable-length motifs in data series. In *ACM SIGMOD Conference on Management of Data (SIGMOD)*. 1053–1066.
[7] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. 2018. Protecting sensory data against sensitive inferences. In *Workshop on Privacy by Design in Distributed Systems (W-P2DS)*.
[8] George B Moody and Roger G Mark. 2001. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine* 20, 3 (2001), 45–50.
[9] Abdullah Mueen and Nikan Chavoshi. 2015. Enumeration of time series motifs of all lengths. *Knowledge and Information Systems* 45, 1 (2015), 105–132.
[10] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. 2009. Exact discovery of time series motifs. In *SIAM International Conference on Data Mining (SDM)*. 473–484.
[11] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. 2017. The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.
[12] 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 *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 1317–1322.

[13] 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 *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 739–748.

A EXISTING MOTIF ENUMERATION ALGORITHMS

A.1 SetFinder

Algorithm 3 shows the detailed SetFinder algorithm. It consists of two steps: It first counts the number of non-trivial-subsequences within radius R for each subsequence in $\mathcal{A}$, and then iteratively finds motifs and corresponding motif clusters according to the counts sorted in descending order. Sphere$(\mathbf{D}_i, R)$ returns $\mathcal{R}(\mathbf{X}_i, R)$ by extracting $\mathbf{D}_i \leq R$.

In this paper we utilized the MASS algorithm to speed up the original SetFinder algorithm.

Algorithm 3 SetFinder

Input:
Time series $\mathbf{X}$
Subsequence length W
Radius R
Output:
Ordered list of motifs $\mathbf{M}$
Ordered list of motif clusters $\mathcal{S}$

(i) Count the number of subsequences within radius R
$\mathcal{A} \leftarrow$ SlidingWindow$(\mathbf{X}, W)$
for all $\mathbf{X}_i \in \mathcal{A}$ **do**
  $\mathbf{D}_i \leftarrow$ MASS$(\mathbf{X}_i, \mathbf{X})$
  $\boldsymbol{U}[i] \leftarrow |$Sphere$(\mathbf{D}_i, R)|$
end for
(ii) Find motifs and motif clusters in order of significance
$k \leftarrow 1$
$C \leftarrow \mathcal{A}$
while $|C| > 0$ **do**
  $\mathbf{X}_i \leftarrow \arg\max_{\mathbf{X}_j \in C} \boldsymbol{U}[j]$
  if $\boldsymbol{U}[i] = 0$ **then**
    break
  end if
  $\mathbf{D}_i \leftarrow$ MASS$(\mathbf{X}_i, \mathbf{X})$
  $\mathbf{M}[k] \leftarrow \mathbf{X}_i$
  $\mathcal{S}[k] \leftarrow$ Sphere$(\mathbf{D}_i, R)$
  $k \leftarrow k + 1$
  $C \leftarrow C \setminus (\{\mathbf{X}_i\} \cup \mathcal{N}_i \cup$ Sphere$(\mathbf{D}_i, 2R))$
end while
return $\mathbf{M}, \mathcal{S}$

A.2 ScanMK

Algorithm 4 depicts the detailed ScanMK algorithm. It iterates the process of finding closest pairs and their matches, adding them to a motif cluster and removing them and their trivial matches from the candidate set after each iteration.

In this paper we modified the original algorithm by replacing the MK algorithm with STAMP-based computation in order to find the closest-pair motifs efficiently.

Algorithm 4 ScanMK

Input:
Time series $\mathbf{X}$
Subsequence length W
Radius R
Output:
Ordered list of motifs $\mathbf{M}$
Ordered list of motif clusters $\mathcal{S}$

(i) Compute the closest-pair distances
$\mathcal{A} \leftarrow$ SlidingWindow$(\mathbf{X}, W)$
$\boldsymbol{P} \leftarrow \infty$
$\boldsymbol{I} \leftarrow \mathbf{0}$
for all $\mathbf{X}_i \in \mathcal{A}$ **do**
  $\mathbf{D}_i \leftarrow$ MASS$(\mathbf{X}_i, \mathbf{X})$
  $\boldsymbol{P}, \boldsymbol{I} \leftarrow$ ElementWiseMin$(\boldsymbol{P}, \boldsymbol{I}, \boldsymbol{D}_i, i)$
end for
(ii) Find motifs and motif clusters in order of significance
$k \leftarrow 1$
$C \leftarrow \mathcal{A}$
while $|C| > 0$ **do**
  $\mathbf{X}_i \leftarrow \arg\max_{\mathbf{X}_j \in C} \boldsymbol{P}[j]$
  if $\boldsymbol{P}[i] > R$ **then**
    break
  end if
  $j \leftarrow \boldsymbol{I}[i]$
  $\mathbf{D}_i \leftarrow$ MASS$(\mathbf{X}_i, \mathbf{X})$
  $\mathbf{D}_j \leftarrow$ MASS$(\mathbf{X}_j, \mathbf{X})$
  $\mathbf{M}[k] \leftarrow \mathbf{X}_i$
  $\mathcal{S}[k] \leftarrow$ Sphere$(\mathbf{D}_i, R) \cap$ Sphere$(\mathbf{D}_j, R)$
  $k \leftarrow k + 1$
  $C \leftarrow C \setminus (\{\mathbf{X}_i\} \cup \mathcal{N}_i \cup$ Sphere$(\mathbf{D}_i, 2R))$
  $C \leftarrow C \setminus (\{\mathbf{X}_j\} \cup \mathcal{N}_j \cup$ Sphere$(\mathbf{D}_j, 2R))$
end while
return $\mathbf{M}, \mathcal{S}$