A Framework for Discovering Variable-length Motifs in Medical Data Streams

Le Sun¹, Jinyuan He², Chen Wang¹, Jiangang Ma³, Hai Dong³, Yanchun Zhang²

¹School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China
sunle2009@gmail.com, wangchennuist@126.com

²Centre for Applied Informatics, Victoria University, VIC, Australia
jinyuan.he@live.vu.edu.au, {jiangang.ma, yanchun.zhang}@vu.edu.au

³School of Science, RMIT University, VIC, Australia
hai.dong@rmit.edu.au

Received 9 January 2018, Revised 17 January 2018, Accepted 22 January 2018

Abstract. In this paper, we explore two key problems in time series motif discovery: releasing the constraints of trivial matching between subsequence with different lengths and improving the time and space efficiency. The purpose of avoiding trivial matching is to avoid too much repetition between subsequence in calculating their similarities. We describe a limited-length enhanced suffix array based framework (LiSAM) to resolve the two problems. We first convert the continuous time series to the discrete time series using the Symbolic Aggregate approXimation procedure, and then introduce two covering relations of the discrete subsequence: α-covering between the instances of LCP (Longest Common Prefix) intervals and β-covering between LCP intervals to support the motif discovery: if an LCP interval is β-uncovered, its instances form a motif. The βUncover algorithm of LiSAM identifies the β-uncovered l-intervals, in which we introduce two LCP tabs: presuf and nextsuf to support the identification of the α-uncovered instances of an l-interval. Experimental results on Electrocardiogram signals indicate the accuracy of LiSAM on finding motifs with different lengths.

Keywords: motif discovery, suffix array, time series

1 Introduction

Discovering motifs for time series is an important and tough task. It has been proved that the subsequence clustering is meaningless in unsupervised data stream mining area, and the motif grouping in the discrete data stream mining has been applied as a replacement of the subsequence-clustering in the real-time series [1]. In this paper, we focus on two primary issues in the time series motif discovery: reducing the computational complexity and avoiding unexpected repetitions among different motifs and among instances of one motif.

The subsequence trivial matching [2] and the overlapping among different motifs [3] are two types of motif repetition issues in the literature. To avoid trivial matching, some methods assumed that the instances of a motif do not overlap with each other at all [4]. We believe that, however, a more flexible and user-manageable mechanism is necessary to control the numbers and styles of the discovered patterns.

Enhancing the time and space complexity, and at the same time, guarantying an expected accuracy is always one of the top topics in data processing. Some motif discovery researchers used approximate solutions to get an acceptable computational complexity [5]. In this work, we propose an unsupervised Limited-length suffix array based Motif Discovery algorithm (LiSAM) for continuous time series, which is time and space efficient, and supports approximately discovering motifs in different lengths. We first
convert the continuous time series to the discrete time series by using the Symbolic Aggregate approXimation procedure (SAX) [6], and then identify the different-length motifs based on the discrete time series. Our illustration of discrete motif discovery is on the basis of an exact substring matching procedure, however, we can easily embed the existing approximate substring matching methods, such as [7-8], in LiSAM. The distinctive contribution of LiSAM is as below:

- LiSAM can discover motifs in different lengths (e.g., $maxLength$ to $minLength$ provided by users), avoid the unexpected trivial-matching by allowing user-defined overlapping degree (represented as $\alpha$) between the instances of motifs, and support discovering motifs that overlap with each other in a specified degree ($\beta$). It can either be an automatic or semi-automatic algorithm by either manually setting all the parameters or by using default parameters (e.g., set $maxLength = \frac{1}{2} |T|$ is a time series), $minLength = 2$, $\alpha = 0$ and $\beta = 0$).
- We conduct extensive experiments based on both synthetic time series datasets to evaluate the performance of LiSAM. Experimental results show the high accuracy of LiSAM and its applicability in the pattern recognition of data streams such as ECG.

2 Related Work and Background Knowledge

2.1 Related Work

There has been a large amount of effort on exploring approximately accurate and fast motif discovery algorithms in continuous time series. The SAX [9] (Symbolic Aggregate approXimation) method was proposed to symbolize the continuous time series. The SAX method can lower bound the distance between the original time series based on the symbolized time series. Because of its time efficiency, it supports a streaming time series conversion. Based on the SAX, Moskovitch et al. [10] present a classification framework for clustering multivariate time series, which first transforms continuous time series to symbolic time series; then discovers frequent occurrence patterns based on data mining techniques; and at last designs classifiers based on the identified patterns. Floratou et al. [11] concentrated on improving the accuracy of motif discovery in continuous sequential data. They proposed a suffix tree based algorithm FLAME to find different motifs with high accuracy. As motif discovery is an unsupervised process, it is difficult to manually determine the lengths of the motifs in a time series. Against this problem, Yingchareonthawornchai [12] used a compression-based method to discover motifs with variable lengths. The proposed method also supports the motif evaluation and ranking in terms of their importance to the time series. Xie and Wang [13] developed an algorithm ADCMCST that supports the approximate construction of tree networks of wireless sensor networks, in order to balance the node payload and enhance the network lifetime. AdaBoost is a popular classification method in pattern recognition area. Wen et al. [14] introduced an advanced learning schema based on AdaBoost classification algorithm for vehicle detection, which reduces the complexity of time-consuming.

2.2 Background: Enhanced Suffix Array

We briefly introduce the frequently used symbols and the basic concept of the enhanced suffix array in this section. Readers can refer to [15-16] for more details. We first introduce and list the symbols and their definitions in this paper in Table 1.

A suffix array of $S$ is an integer array (suftab) having values $k \in [0, n]$. An enhanced suffix array (ESA) is a suffix array with a number of additional supporting arrays, where two of them (lcptab and bwttab) will be used in this paper. We use an example of $S_{\text{examp}} = aceaceacece$ to describe the ESA that is shown in Table 2. The suftab keeps the starting positions of suffixes of $S$ in ascending lexicographic order. The definition of lcptab is in Table 1. From Table 2, lcptab[0] = 0 and lcptab[n] = 0.

To group the suffixes that have the longest common prefixes, the concept of LCP interval is proposed. We describe below the definition of an LCP interval.
Table 1. Symbols and definitions

<table>
<thead>
<tr>
<th>Concepts</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a continuous time series</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>a finite ordered alphabet</td>
</tr>
<tr>
<td>$\Sigma^*$</td>
<td>strings over $\Sigma$</td>
</tr>
<tr>
<td>$\Sigma^+$</td>
<td>$\Sigma^*$ without null</td>
</tr>
<tr>
<td>$S$</td>
<td>a discrete time series over $\Sigma$ with length $</td>
</tr>
<tr>
<td>$\sim$</td>
<td>$\sim \in \Sigma$, $\sim &gt; \sigma$, $\forall \sigma \in \Sigma$</td>
</tr>
<tr>
<td>$S[i,j]$</td>
<td>substring of $S$ between positions $i$ and $j$</td>
</tr>
<tr>
<td>suftab $[suf]$</td>
<td>suffix array table of $S$</td>
</tr>
<tr>
<td>presuf $[pre]$</td>
<td>the suffix index of the previous position of the current suffix in suftab</td>
</tr>
<tr>
<td>nextsuf $[next]$</td>
<td>the suffix index of the next position of the current suffix in suftab</td>
</tr>
<tr>
<td>$S_{suftab}[i]$</td>
<td>the $i$th suffix of $S$, $i \in [0, n]$</td>
</tr>
<tr>
<td>lcptab $[i]$</td>
<td>Longest common prefix (LCP) of $S_{suftab}[i-1]$ and $S_{suftab}[i]$</td>
</tr>
<tr>
<td>bwttab $[i]$ (bwt)</td>
<td>$S_{[suftab[i]-1]}$, if $suf[i] &gt; 0$; null, if $suf[i] = 0$</td>
</tr>
<tr>
<td>$l_0$ - interval, $l_1$ - $[i, j]$</td>
<td>an LCP interval from index $i$ to index $j$ with length $\ell$</td>
</tr>
<tr>
<td>$l$ - $[i, j]$</td>
<td>singleton interval (SI): $S_{suftab[i]}$</td>
</tr>
<tr>
<td>NSI</td>
<td>non-singleton interval</td>
</tr>
<tr>
<td>$m_0$ - $[i, j]$</td>
<td>$m$-interval: instances of $l_0$ interval forming a motif</td>
</tr>
</tbody>
</table>

Table 2. An enhanced suffix array

<table>
<thead>
<tr>
<th>index</th>
<th>suf</th>
<th>lcptab</th>
<th>bwt</th>
<th>$S_{suftab[i]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>null</td>
<td>aceaceacece~</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>e</td>
<td>aceacece~</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>e</td>
<td>acece~</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>a</td>
<td>ceacecece~</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>a</td>
<td>ceace~</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
<td>a</td>
<td>ece~</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>e</td>
<td>ce~</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td>c</td>
<td>eaceacece~</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>4</td>
<td>c</td>
<td>eacece~</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>0</td>
<td>c</td>
<td>ece~</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>c</td>
<td>e~</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>0</td>
<td>e</td>
<td>~</td>
</tr>
</tbody>
</table>

Definition 1. Given $S$ and its Enhanced suffix array, an interval $[i, j]$ of index (see Table 2), where $i, j \in [0, n]$ and $i < j$, is a LCP interval with LCP length $\ell$ if the following conditions are satisfied: (1) $lcptab[i] < \ell$; (2) $lcptab[k] \geq \ell$, $\forall k \in [i + 1, j]$; (3) $lcptab[k] = \ell$, if $\exists k \in [i + 1, j]$; (4) $lcptab[j + 1] < \ell$. The LCP interval $[i, j]$ with LCP length $\ell$ can be represented as $l_\ell - [i, j]$.

An LCP interval tree indicates the embedding and enclosing relations between LCP intervals. We describe an example of LCP tree of $S_{examp}$ in Fig. 1. We can see that the root of the LCP tree covers all the suffixes of $S_{examp}$. The child intervals are the intervals embedded in their father intervals. The leaf intervals do not enclose any NSI. A fast traversing procedure for LCP trees is defined in [16]. Note that in this paper we use $l_\ell$ to represent an $l$-interval with LCP length $\ell$, while use $m_\ell$ to represent a motif interval (Def.6) with LCP length $\ell$. In addition, we refer the normal 'LCP intervals' to non-singleton intervals (NSIs).

3 Problem Definition

In this section, we introduce the basic concepts to be used in LiSAM. A continuous time series $T$ is a sequence of real values that have temporal properties. To identify the motifs of a time series, previous work has given different forms of motif definitions [17]. We summarize these definitions and present a comprehensive motif concept in Definition 2.
where we define a general concept of an approximate motif of discrete time series as below.

**Definition 2.** A motif $M$ of a time series $T$ is a set of similar subsequence $SQ = \{sq_1, ..., sq_m\}$ such that $n \geq 2$, and $\forall i, j \in [0, n-1]$, the length of $|sq_i| \geq 2$, $|sq_i \cap sq_j| \leq \alpha$, and $Dis(sq_i, sq_j) \leq \delta$, where $\delta$ is an overlapping threshold to constraint the overlapping length between two subsequence of $M$, $Dis$ is a distance measure, and $d \geq 0$ is a small value to guarantee a certain similarity among subsequence. We call a subsequence of $M$ as an instance of this motif.

**Definition 3.** Given two $l$-intervals $l_i = [i_1, i_2]$ and $l_j = [j_1, j_2]$, $s_{i_2}(k_1 \in [i_1, j_1])$ is an instance of $l_i$, $s_{k_2} \in [j_2, j_2]$ is an instance of $l_j$, $sz_1 = |i_1 - i_2| + 1$: (1) instance $s_{i_2}$ is $\alpha$-covered by $s_{k_2}$ if $l_1 \leq l_2$, $s_{i_2}$ overlaps with $s_{k_2}$ at sub-string $s''$, where $s'' \subset s_{i_2}$ and $s'' \subset s_{k_2}$, and $|s''| > \alpha$, $|s_{i_2}| \geq \alpha \geq \frac{1}{2} |s_{i_2}|$. Or else, $s_{i_2}$ is $\alpha$-uncovered by $s_{k_2}$; (2) Interval $l_i$, is $\beta$-covered by $l_j$, if $h$ instances of $l_i$ are covered by the instances of $l_j$, where $(sz_1 - \beta) < h \leq sz_1$, and $h$ is a pre-defined threshold. Or else, $l_i$ is $\beta$-uncovered (or uncovered) by $l_j$.

From the definition of $l$-interval, an $l_i$-interval can be seen as at least two suffixes that have the LCP of length $l$. Therefore, an $l$-interval can be seen as a pattern of $S$, and the LCPs of the $l$-interval correspond to the occurrences of the pattern. A pattern of $S$ is defined as:

**Definition 4.** Given an alphabet set $\Sigma$ and an approximate time series $S \in \Sigma^*$, a pattern of $S$ is a time series $pt$ that $1 \leq |pt| \leq |S|$, $pt \subset S$, and occurs $k (k \geq 2)$ times in $S$ at positions $\{p_1, ..., p_k\}$, $p_1 \neq ... \neq p_k$, where a position is the start point of an occurrence of $pt$ in $S$.

In the above definition, we define that a pattern should occur at least twice in a time series. From the definition of $l$-interval, an $l_i$-interval is composed of at least two suffixes that have the LCP of length $l$. Therefore, an $l$-interval can be seen as a pattern of $S$, and the LCPs of the $l$-interval correspond to the occurrences of the pattern. However, the requirement on the minimum occurrence times of a pattern varies in different situations. For example, in a very long $S$ (e.g., $\geq$ 10 thousands), the element that repeats a small number of times (e.g., $< 10$ times) is meaningless for the time series analysis. Therefore, we define a general concept of an approximate motif of discrete time series as below.

**Definition 5.** Assume $u = S[a, b] (a \leq b)$ is an instance of an $l$-interval $l_i = [i, j]$ of $S$. Given a lower bound $mint (mint \geq 2)$ of the pattern occurrences, if $e = j - i + 1 \geq mint$, and $l_i$ is uncovered by any other $l$-intervals of $S$, it is an approximate motif of $S$, represented as $mf = \langle \ell ; P = \{p_1, ..., p_\ell\} \rangle$, where $\ell = b - a + 1 (l \geq 1)$ is the length of $mf$, $p_i$ is the start index of the occurrences of $u$ in $S$, and $\ell$ is the size of the motif $mf$.

In the following description, a motif of $S$ refers to an approximate motif. The relation between an $l$-interval and a motif of $S$ is defined as an $m$-interval.

**Definition 6.** For an $l$-interval $l_i = [i, j]$ of $S$, if the instances of $l_i$ is one-to-one matched to the occurrences of a motif $mf = \langle \ell ; suftab[i], ..., suftab[j] \rangle$, then $l_i$ is an $m$-interval, represented as $m_i = [i, j]$.

In the following sections, we refer an $m$-interval to a motif.
4 Limited-length Suffix-array-based Motif Discovery

4.1 Identify $\beta$-uncovered $l$-intervals for Discrete Time Series

In this section, we first discuss the determination of the $\beta$-uncovered intervals with an assumption that $\alpha = 1$ for the $\alpha$-covering relation between instances. In section 4.2, we introduce the $\alpha$-covered algorithm and illustrate how to interactively perform the $\beta$- and $\alpha$-covered algorithms to identify the motifs.

In ESA, identifying LCP intervals is a bottom-up traversing process. When an LCP interval is being processed, its child intervals have been identified, so the child intervals can support the determination of $\beta$-covering of the LCP interval. We distinguish the case of an LCP interval having a single character (the singleChar interval) with the case that the interval is comprised of more than one character (the multiChar interval). We give Lemma 1 to identify the $\beta$-uncovered multiChar intervals.

**Lemma 1.** Given an multiChar LCP interval $l_i = [i, j]$, its child intervals $\Theta$, and the lower bound of the occurrence times of motifs $minT \geq 2$, let $\lambda = j - i + 1$, $l_i$ is $\beta$-uncovered by other $l$-intervals if any of the following conditions is satisfied:

1. $|\Theta| = 0$, so the characters after the LCP subsequences of $l_i$, are pair-wise different, i.e., $bwttab[i] \neq \ldots \neq bwttab[j]$;
2. $|\Theta| = 0$, and $\exists \sigma_1 \ldots \neq \sigma_\gamma \in bwttab[i \ldots j]$, $minT + 1 \leq \gamma \leq \lambda$;
3. $|\Theta| > 0$, $\exists l_i \in \Theta$ and $\lambda_0 = z_i - w_i + 1 \geq minT$, and $\exists r_1 \ldots r_k \in [w_i, z_i]$ and $h_1 \ldots h_k \in [i, j]$ but $\notin [w_i, z_i]$ that $bwttab[r_1] \neq bwttab[h_1], \ldots, bwttab[r_k] \neq bwttab[h_k], k \geq minT$;
4. $|\Theta| > 1$, $\exists m_i - [w_i, z_i], \ldots, m_i - [w_i, z_i] \in \Theta$, $k \geq minT$, and $m_i, \ldots, m_i$ are $\beta$-uncovered.

**Proof of Lemma 1.**

1. $|\Theta| = 0$, so the characters after the LCP subsequences of $l_i$, are pair-wise different, i.e., $S[suftab[i] + 1 \ldots j] \neq S[suftab[i] + 1]$. Meanwhile, $\lambda = minT$ and $bwttab[i] \neq \ldots \neq bwttab[j]$. So the instances of $l_i$ are not covered by any longer repeated sequences in $S$. Hence, $l_i$ is $\beta$-uncovered.

2. if $\gamma > minT$, then at least $minT + 1$ characters in $bwttab[i, j]$ are different (assume $bwttab[k_1] \neq bwttab[k_2]$); and as $\Theta = 0$, the $k_i$th and $k_2$th LCP subsequences are not covered by any longer subsequences of its child intervals. So $l_i$ is $\beta$-uncovered.

3. assume $l_i$ have one child interval $c_0$, where $\lambda_0 \geq minT$, $i \leq w_0 \leq z_0 \leq j$ and $\lambda > minT$. (a)Assume $\lambda - \lambda_0 = 0$, then $l_i = c_0$, $c_0$ is not a child interval of $l_i$. Assumption (a) is not true. (b) Assume $\lambda - \lambda_0 < minT$, then there are $\lambda - minT$ instances of $l_i$ covered by the instances of $c_0$, so interval $l_i$ is covered by interval $c_0$, and $l_i$ is not a motif. Assumption (b) is not true. (c) as $\lambda - \lambda_0 \geq minT$, then there are at least $minT$ instances of $l_i$ that are not covered by the instances of $c_0$. In addition, $\exists \sigma_i \neq \ldots \neq \sigma_\gamma \in bwttab[i \ldots j]$, $minT < \gamma \leq \lambda$, based on the proof of (3), $l_i$ is $\beta$-uncovered.

4. if $k = minT$, as $m_i, \ldots, m_i$ are $k$ motifs, the subsequences in all of the $minT$ intervals are pairwise different, so the interval $l_i$, where $\ell < l_i, \ldots, \ell_{\text{max}}$, cannot be covered by any of $\{m_i \ (as \ \forall |m_i| \geq minT, t \in [1, k], t 
eq 1, \ldots, \ell_{\text{max}} \}$, that is the interval $l_i$ cannot be individually covered by any of its $k$ child motifs. So $l_i$ is $\beta$-uncovered.

For singleChar intervals, the problem of determining their motif property is to avoid finding a shorter singleChar motif covered by a longer singleChar motif. Lemma 2 shows how to determine if a singleChar interval is $\beta$-uncovered.

**Lemma 2.** Given a singleChar interval $l_i - [i, j]$ that its LCP subsequence, i.e., $S[suftab[i], suftab[i] + 1 \ldots j]$, is only comprised of one character (assume $\sigma$),

1. if $l_i$ does not have child intervals, i.e., $|\Theta| = 0$ and $\exists \sigma_i \neq \ldots \neq \sigma_\gamma \in bwttab[i \ldots j]$, $minT + 1 \leq \gamma \leq \lambda$, then $l_i$ is $\beta$-uncovered;
2. if $|\Theta| > 0$ and $\theta - [w, z] \in \Theta$, that $\exists \sigma_i \neq \ldots \neq \sigma_\gamma \neq \sigma$ and $\sigma_{1 \ldots, j} \in bwttab[w \ldots z]$, where $z' - w' + 1 \geq 2$, $\lambda > 0$, $[w' \ldots z'] \subset [i \ldots j]$ and $[w' \ldots z']$ is $\beta$-uncovered by $[w \ldots z]$;
Proof of Lemma 2.

(1) As \( l_i \) does not have child intervals, \( l_i \) cannot be covered by an interval comprising LCP subsequences of \( u = S[suftab[k_1], ..., suftab[k_1] + \ell - 1] \), where \( k_1 \in [i, j] \), \( \ell > \ell' \). In addition, as \( \exists \sigma_i \neq \ldots \neq \sigma_r \sigma_{1 \ldots r} \in bwttab[i ... j] \), \( \min T + 1 \leq r \leq \lambda \), \( l_i \) cannot be covered by an interval comprising LCP subsequences of \( u = S[suftab[k_2] - 1, ..., suftab[k_2] - 1 + \ell'' - 1] \), where \( k_2 \in [i, j] \), \( \ell'' > \ell' \). So \( l_i \) is a \( \beta \)-uncovered.

(2) Assume \( u = S[suftab[i] \ldots suftab[j] + \theta - 1] \) is the prefix of \( l_i \), and \( u' = S[suftab[w] \ldots suftab[w] + \theta - 1] \) is the prefix of \( l_0 \), and assume \( \exists \sigma_i \in bwttab[w \ldots z] \) and \( \exists \sigma_2 \in bwttab[w', z'] \) that \( \sigma_i \neq \sigma \) and \( \sigma_2 \neq \sigma \), then (1) any child interval \( l_0 \) cannot cover \( l_i \) since \( z' - w' + 1 \geq 2 \); (2) we prove that under condition 2 in Lemma 2, if \( l_i \) is a singleChar interval with LCPs like \( \mu = x_i, ..., x_{\lambda} \), then not \( \exists l_0 \) (the strings of its singleChar LCP \( \mu = x_i, ..., x_{\lambda} (\theta \neq \ell) \) that cover \( l_i \). Assume exist such \( l_0 \) then the strings of the LCP of \( l_0 \) include all the strings whose prefixes with length \( \theta \) are \( u \), i.e., \( \exists k(z - w + 1) \) subsequences \( u \subset S \), and there must be \( \eta(=k \times (\theta - 1)) \) bwttabs that \( bwttab[r_1] = ..., bwttab[r_{\lambda}] = \sigma, \eta = z'-w'+1 + k + \eta = j-i+1 \); which means there must not exist \( \sigma_i, ..., \lambda \neq \sigma, \lambda > \theta \) in bwttab[w', z']. This is contradicting with condition 2 of Lemma2, so the second statement (2) is correct. Combining statements (1) and (2), the singleChar interval \( l_i \) is \( \beta \)-uncovered given condition 2 of Lemma 2.

5 Performance Evaluation

In this section, we present the experimental results to show the efficiency of LiSAM. Our experiments are conducted on a windows 64-bit system with 3.2GHz CPU and 4 GB RAM, and is implemented by Java.

We extract patterns from six different ECG data streams, repeat each pattern 30 times and insert the repeated patterns to Gaussian white noise data streams separately. The information of the extracted motifs, which represents the overlapping degree between the inserted pattern \( (p_i) \) and the discovered pattern \( (d_j) \):

\[
old = \sum_i \sum_{d} \frac{overlap(p_i, d_j)}{\text{length(plantedPattern)}}
\]

(1)
The old values for each of the simulated ECG time series are shown in Table 4. We can see that the proposed motif discovery algorithm can identify the inserted patterns with very high accuracy (all over 0.9). We compare the shapes of the planted patterns and the discovered motifs in each of the six time series in Fig. 2. In addition, we use the average pair-wise distances among instances (represented as inDis) of a motif to measure the dis-similarity degree of the instances of one discovered motif (e.g., motif m), which is calculated as:

\[
inDis(m) = \frac{\sum_{i,j} \text{dis}(m_i, m_j)}{m\text{.len} \times m\text{.size}} \tag{2}
\]

where \(m_i\) and \(m_j\) represent the \(i\)th and \(j\)th instances of \(m\); and \(m\text{.len}\) is the length of this motif; \(m\text{.size}\) is the number of its instances, and \(\text{dis}\) is the Euclidean distance function. The average \(\text{inDis}\) value of each time series is shown in Table 4, and the distance distribution of each instance pair of the most frequent motif for each dataset is shown in Fig. 3. We can see that the instances of one motif for each datasets are very close to each other, all of which have less than 0.1 average instance dissimilarities.

6 Conclusion and Future Work

In this paper, we proposed an algorithm LiSAM to resolve two important problems in discovering approximate time series motif: releasing the constraints of trivial matching between subsequences with different lengths and improving the time and space efficiency. We proposed two covering relations: \(\alpha\)-covering between instances of \(l\)-intervals and \(\beta\)-covering between \(l\)-intervals to support the motif discovery. We use the LiSAM algorithm to identify the \(\beta\)-uncovered \(l\)-intervals, and we introduced two LCP tabs: \(\text{presuf}\) and \(\text{nextsuf}\) to support the identification of the \(\alpha\)-uncovered instances of an \(l\)-interval.
A Framework for Discovering Variable-length Motifs in Medical Data Streams

Fig. 3. Distance distribution of instance pairs of the most frequent motif for six datasets

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (Grants No. 61702274) and the Natural Science Foundation of Jiangsu Province (Grants No. BK20170958).

References

[9] M. Gupta, J. Gao, C.C. Aggarwal, J. Han, Outlier detection for temporal data: a survey, IEEE Transactions on Knowledge and Data Engineering 26(9)(2014) 2250-2267.


