Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Comprehensive and efficient discovery of time series motifs*

Lian-hua CHI†[1], He-hua CHI[2], Yu-cai FENG[1], Shu-liang WANG[3], Zhong-sheng CAO[1]

([1]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)

([2]*State Key Laboratory of Software Engineering, Computer School, Wuhan University, Wuhan 430079, China*)

([3]*International School of Software, Wuhan University, Wuhan 430079, China*)

†E-mail: lianhua1221@gmail.com

Received Feb. 16, 2011; Revision accepted June 15, 2011; Crosschecked Nov. 4, 2011

Abstract: Time series motifs are previously unknown, frequently occurring patterns in time series or approximately repeated subsequences that are very similar to each other. There are two issues in time series motifs discovery, the deficiency of the definition of K-motifs given by Lin *et al.* (2002) and the large computation time for extracting motifs. In this paper, we propose a relatively comprehensive definition of K-motifs to obtain more valuable motifs. To minimize the computation time as much as possible, we extend the triangular inequality pruning method to avoid unnecessary operations and calculations, and propose an optimized matrix structure to produce the candidate motifs almost immediately. Results of two experiments on three time series datasets show that our motifs discovery algorithm is feasible and efficient.

Key words: Time series motifs, Definition of K-motifs, Optimized matrix structure, Fast pruning method
doi:10.1631/jzus.C1100037 **Document code:** A **CLC number:** TP391.4

1 Introduction

In recent decades, research concerning time series data mining has received much attention. Time series motifs are the frequently occurring, previously unknown patterns in time series data. By using the motifs, a variety of applications in data mining can be extended, such as discovery of association rules, time series classification, and detection of interesting behaviors.

At present, time series motifs are applied in medicine (Abe *et al.*, 2005), telemedicine (Guyet *et al.*, 2007), weather prediction, etc. Theoretical research on time series motifs began with a pioneering paper (Lin *et al.*, 2002), which formalized the idea of approximately repeated subsequences by introducing time series motifs. Because it is very difficult to compute the exact solution for the motif discovery problem, more and more researchers have proposed fast approximate algorithms to find motifs (Chiu *et al.*, 2003; Tanaka *et al.*, 2005; Guyet *et al.*, 2007; Beaudoin *et al.*, 2008). In another work, a new algorithm was proposed to extract approximate motifs that capture portions of the time series with a similar and eventually symmetric behavior (Ferreira *et al.*, 2006). Recent advances in time series motifs research come from nonlinear-field time series analysis. Time series data has been transformed to a network (Zhang and Small, 2006), and the motifs in the networks are used to distinguish the original time series of different dynamical properties (Xu *et al.*, 2008).

In spite of extensive research in recent years, there are still some deficiencies in time series motifs

* Project supported by the "Nuclear High Base" National Science and Technology Major Project (No. 2010ZX01042-001-003), the National Basic Research Program (973) of China (No. 2007CB310804), and the National Natural Science Foundation of China (No. 61173061)

discovery: (1) Previous studies are based mainly on the original definitions in Lin *et al.* (2002). However, the definition of K-motifs in Lin *et al.* (2002) is not comprehensive, and it can easily lead to the loss of more frequently occurring patterns in time series data. (2) The efficiency of extracting motifs in motifs discovery algorithms is still not high.

To find more frequently occurring patterns in time series, we propose a comprehensive and efficient motifs discovery algorithm. We modify the definition of K-motifs in Lin *et al.* (2002), which helps find more frequently occurring patterns. Meanwhile, we extend the triangular inequality pruning method and propose an optimized matrix structure to improve the efficiency. By case studies, we demonstrate the feasibility and efficiency of our motifs discovery algorithm.

2 Basic principle

Before describing our algorithm, we provide some definitions of the key terms that will be used throughout this paper.

Definition 1 (Time series, Lin *et al.*, 2002) A time series, i.e., a sequence $T = t_1, t_2, \ldots, t_{\mathrm{T\_length}}$, is an ordered set of T_length real valued variables.

The sequence of real-valued values is a typically temporal ordering. Other kinds of well-defined orderings, such as shapes (Ueno *et al.*, 2006) and handwritten text, can be considered as time series. Time series can be very long, sometimes containing billions of observations (Hegland *et al.*, 2001). In this study, we confine our interest to the local of the time series, which are called subsequences.

Definition 2 (Subsequence, Lin *et al.*, 2002) Given a time series T of length T_length, a subsequence C of T is a sampling of length C_length ($<$ T_length) of contiguous position from T; that is, $C = t_p, t_{p+1}, \ldots, t_{p+\mathrm{C\_length}-1}$ for $1 \le p \le$ T_length $-$C_length $+ 1$.

By Definition 2 we know that a time series of length C_length has (T_length $-$ C_length $+ 1$) subsequences of length C_length. It is important to provide a definition that can determine if a given subsequence is similar to other subsequences (André-Jönsson and Badal, 1997; Yi and Faloutsos, 2000). These similar subsequences are known as matches (Chiu *et al.*, 2003).

Definition 3 (Match, Lin *et al.*, 2002) Given two subsequences C_i (starting from position i), C_j (starting from position j) and the distance function $D(C_i, C_j)$, we say that C_i matches C_j, if $D(C_i, C_j) < R$. The R is a positive real number (called 'range') which can be assigned by users.

The distance value R used in Definition 3 determines the approximate similarity. The above definition of 'match' is very clear and intuitive, but the definition of motifs should not include the trivial matches (Ferreira *et al.*, 2006).

Definition 4 (Trivial match, Lin *et al.*, 2002) Given a time series T, containing a subsequence C_i beginning at position i and a matching subsequence C_j beginning at j, we say that C_j is a trivial match to C_i if either $i = j$ or there does not exist a subsequence $C_{j'}$ beginning at j' such that $D(C_i, C_{j'}) > R$, and either $i < j' < j$ or $j < j' < i$.

Next, we provide our relatively comprehensive definition of K-motifs.

Definition 5 (K-motifs) Given a time series T, a subsequence length C_length, and a range R, the most significant motif in T (called thereafter 1-motif) is the subsequence C_1 that has the highest count of non-trivial matches. The Kth most significant motif in T (called thereafter K-motif) is the subsequence C_K that has the Kth highest count of non-trivial matches (Lin *et al.*, 2002), and satisfies either condition below. We denote the set of subsequences matching C_K by $M(C_K)$.

(1) $D(C_K, C_i) > 2R$, for all $1 \le i < K$;

(2) $R < D(C_K, C_i) < 2R$, for all $1 \le i < K$, and $D(C_K, C_j) > R$ (C_j is any subsequence in $M(C_i)$).

We can also replace conditions (1) and (2) by a single equivalent condition:

The intersection of $M(C_K)$ with $M(C_i)$ is empty.

Fig. 1 illustrates these two conditions on a simple set of time series projected onto the 2D space.

By Definition 5 we propose a relatively comprehensive definition of K-motifs. Compared with the definition of K-motifs given by Lin *et al.* (2002), we add condition (2).

From Lin *et al.* (2002), one needs to judge whether the distance between two motifs is greater than $2R$. The purpose of this judgment is to avoid the situation in which two motifs share the same subsequences. If two motifs have most of the same subsequences, the two motifs are basically the same.

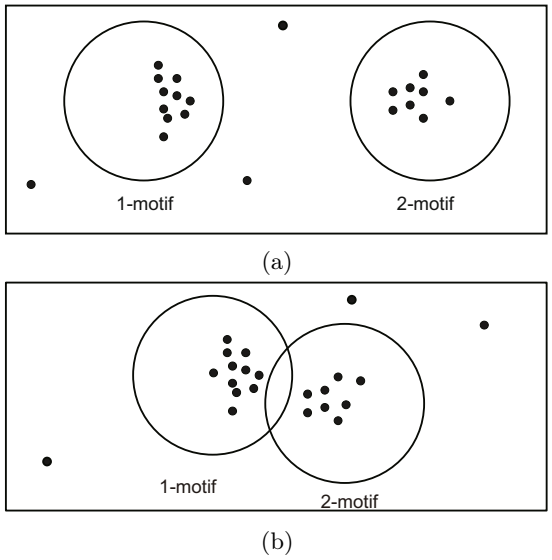

Fig. 1 A visual description of the definition of K-motif. (a) illustrates that the distance between 1-motif and 2-motif is larger than $2R$, and these two motifs will definitely not share the same elements. However, if the distance between two motifs is required only to be between R and $2R$ as in (b), the two motifs also do not share the same elements

In this case, it is meaningless to distinguish these two motifs. From this point of view, we can change some conditions to meet the same requirement. We can reset the distance range. Based on the original scope $2R$, we add a permissible range (R, $2R$). Namely, the intersection of two motifs does not have subsequences (Fig. 1b). This condition also satisfies the requirement that two motifs should not have the same subsequences (Lin *et al.*, 2002). If limited to the definition of K-motifs in Lin *et al.* (2002), we may probably miss some meaningful motifs. The main purpose of improving the theory of the distance range is to find much more frequent patterns.

Now we show why many more valuable motifs can be discovered. If the distance between the candidate motif circle and each identified motif circle is in (R, $2R$), and these two motifs do not share the same subsequences, this motif should not be ignored. However, in accordance with Lin *et al.* (2002), the distance between any two motif circles is greater than $2R$. In this case, we may miss some probable motifs. These probable motifs may include many more subsequences. Therefore, we can set a smaller distance range to avoid the omission of these valuable motifs.

The above definitions are based on the premise that there is a meaningful method to measure the distance between two subsequences. We use the Euclidean distance to measure the distance between two subsequences. Let $D(C_i, C_j)$ represent the Euclidean distance between subsequences C_i and C_j. Recently, extensive empirical comparisons with more complex measures have shown that the Euclidean distance has better features on a wide variety of areas (Ding *et al.*, 2008). More importantly, the Euclidean distance satisfies the triangle inequality. This is a very important property for this study, one that needs to be satisfied by the distance function.

If the length of the subsequences is C_length, the time complexity of computing the Euclidean distance is O(C_length). In our algorithm, we use a fast triangle inequality pruning method to reduce the computation time. In this method, we can make an initial judgment before computing the distance between any two subsequences. If the distance is greater than the given value R, we do not need to calculate the Euclidean distance between any two subsequences. If it is less, we calculate the actual distance. With the help of this pruning method, we can reduce the time complexity. Then, based on an optimized subsequence matrix structure in which the matrix element is subsequence, we can discover the motifs much more rapidly. We assume that there are seven subsequences C_1–C_7, and set C_1 as the reference subsequence. This sequence number is based on the distances from all other subsequences to C_1. If the distances from C_1 to C_2 and C_1 to C_5 are both less than R, we can set the values of the corresponding array elements in the subsequence matrix.

According to this subsequence matrix, we can effectively extract 1-motif that contains the most similar subsequences. Then we delete the subsequences that belong to 1-motif, and extract 2-motif. The algorithms will be described in the following.

Table 1 summarizes the notations used in this paper.

3 Methods

In this section, we discuss our algorithm in detail. The whole process includes time series data preprocessing, initializing the subsequence matrix, executing the 1-motif discovery algorithm, and executing the K-motif discovery algorithm.

Table 1 A summary of the notations used in this paper

Notation	Description
T	A time series
T_length	The length of T
C	A subsequence of T
C_length	The length of C
R	A positive real number which can be assigned by the user
$D(C_i, C_j)$	The Euclidean distance between subsequences C_i and C_j
lower_bound(C_i, C_j)	The difference between $D(C_1, C_j)$ and $D(C_1, C_i)$

3.1 Time series data preprocessing

We extend the triangular inequality pruning method to avoid unnecessary operations and calculations on time series data preprocessing.

Suppose there are num subsequences and the length of each subsequence is C_length. We precompute the distances between any two subsequences. First, randomly select a subsequence as the reference subsequence marked by C_1 from num subsequences. Then calculate the Euclidean distances from other subsequences to C_1. After that, according to the Euclidean distances to C_1, make a linear arrangement of these subsequences. For example, suppose there are seven subsequences. Select one of the subsequences as the reference subsequence C_1, and then calculate the Euclidean distances from C_1 to the other six subsequences. Array these subsequences according to the Euclidean distances, and the order is C_2, C_3, C_4, C_5, C_6, and C_7 (Fig. 2).

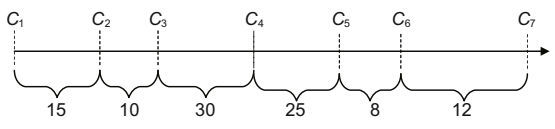

Fig. 2 Differences of Euclidean distances between each pair of consecutive sequences. The marked distances as a condition of pruning are the lower bounds of the adjacent subsequences; the distance between C_1 and C_2 is the real Euclidean distance

Fig. 2 shows the differences of Euclidean distances between each pair of consecutive sequences. In our algorithm, we use lower_bound(C_i, C_j) ($1 < i < j$) to represent the difference between $D(C_1, C_j)$ and $D(C_1, C_i)$. For example, we use lower_bound(C_2, C_3) to represent the difference between $D(C_1, C_3)$ and $D(C_1, C_2)$. According to the triangle inequality principle, the difference of two sides of a triangle is always smaller than the third side. Thus, lower_bound(C_2, C_3) must be smaller than the actual Euclidean distance between C_2 and C_3. If lower_bound(C_2, C_3) is greater than the range R, we do not need to calculate the actual Euclidean distance.

If $R = 9$, then in a motif, the distances from the central subsequence to other subsequences must be less than 9. In Fig. 2, the lower bound between C_2 and C_3 is 10; thus, the actual distance between C_2 and C_3 must be greater than or equal to 10. Then C_2 and C_3 must not be in the same motif. Also, C_2, C_4, C_5, C_6, and C_7 are not in the same motif, because the lower bounds from C_2 to other subsequences are greater than 10. We extensively leverage the triangle inequality pruning method to preprocess the distances and achieve the pruning quickly. An example is given as follows.

Define $R = 15$ and suppose we want to calculate $D(C_1, C_3)$. We need to judge whether lower_bound(C_2, C_3) is less than R. Fig. 2 shows that lower_bound(C_2, C_3) $= 10 < R$. Thus, we need to calculate the actual Euclidean distance between C_2 and C_3. If the actual Euclidean distance is less than R, we can set the value of the corresponding array element. Next we should calculate $D(C_2, C_4)$. The process is the same as the above description. We should judge whether the value of $D(C_1, C_4) - D(C_1, C_2)$ is less than R. As $D(C_1, C_4) - D(C_1, C_2) = 40 > R$, we do not need to calculate the actual Euclidean distance between C_2 and C_4. We do not need to judge C_2 and C_5, C_2 and C_6, or C_2 and C_7. Obviously, if $D(C_1, C_4) - D(C_1, C_2) = 40 > R$ and $D(C_1, C_5)$, $D(C_1, C_6)$, $D(C_1, C_7)$ are larger than $D(C_1, C_4)$, then $D(C_1, C_5) - D(C_1, C_2)$, $D(C_1, C_6) - D(C_1, C_2)$, and $D(C_1, C_7) - D(C_1, C_2)$ are certainly larger than R. Therefore, we do not need to make any further judgments. Based on this point, we can omit a lot of calculations and achieve a fast pruning. Meanwhile, our algorithm can directly control the loop by the order. Thus, it is easier to implement.

Next, we need to choose the reference subsequence. The selection criterion is such that the reference subsequence chosen can help achieve better and faster pruning. First, we calculate the standard deviations from the reference subsequences to other subsequences, and select the reference subsequence that has the largest standard deviation. According

to Mueen *et al.* (2009), the larger the standard deviation, the larger the lower bound.

3.2 Initializing the subsequence matrix

We propose an optimized matrix structure to produce the candidate motifs almost immediately. Before the matrix is initialized, we need to select the best reference subsequence and guarantee that we can achieve better and faster pruning. Algorithm 1 describes this process.

Algorithm 1 Looking for the best reference

1: **for** $i = 1$ to k **do**
2: $\text{ref}_i \leftarrow$ a randomly chosen subsequence C_i from T
3: **for** $j = 1$ to m **do**
4: Compute the actual distance between ref_i, C_j
5: $D(\text{ref}_i, C_j) \leftarrow$ the distance between ref_i, C_j
6: $D[i][j] \leftarrow D(\text{ref}_i, C_j)$
7: **end for**
8: $S_i \leftarrow$ standard_deviation($D[i]$)
9: **end for**
10: Find the largest standard deviation. The corresponding reference subsequence is the best reference time series. Suppose the result is ref_i
11: Sort $D[i]$ from small to large
12: Dist[] $\leftarrow D[i]$
13: **return** Dist[]

In Algorithm 1, T represents the time series, $D[][]$ stores the distances between reference subsequences and other subsequences, k represents the number of reference subsequences, m represents the number of subsequences, and Dist[] stores a set of distances between the best reference subsequence and other subsequences.

In Algorithm 1, we calculate the standard deviations of $D[i]$ $(1 < i < k)$ (line 8) and select the reference subsequence that has the largest standard deviation (lines 1–9). Then we use a matrix Dist[] to store all distances between the best reference subsequence and other subsequences in ascending order (line 12). The information in Dist[] will be used in the following sections.

With the preprocessing on the distances between any two subsequences, we can construct the subsequence matrix. First, we set the values of all matrix elements as 0. Then judge whether the lower bound between any two subsequences is less than R. If it is greater than R, the corresponding matrix element value is still 0. If less, then we need to calculate the actual Euclidean distance $D(C_i, C_j)$. If the actual distance is less than R, then the corresponding matrix element value is set as $D(C_i, C_j)$. If greater, the corresponding matrix element value is still 0.

There is a problem concerning the large number of subsequences. Supposing there are 5×10^7 subsequences, the matrix will be very large and occupy much storage space. According to the symmetry of the Euclidean distance, $T[][]$ is also a symmetric matrix. When most of element values in the matrix are 0, $T[][]$ is a sparse matrix. It would use much space to store unnecessary information. One of the simplest solutions is to store only the actual distances that are less than R, and the corresponding information. Particularly, if R is very small, this method can save much storage space. Based on this optimization, we use a compressed storage structure, the triple sequence table.

Let subr store the line number, col the column number, and dist the corresponding Euclidean distance in row subr and column col. We have dist $<$ R. The triple sequence table Euc_dist stores the values of subr, col, and dist. m represents the number of subsequences. C_count[i] stores the number of subsequences whose distances to C_i are less than R.

Algorithm 2 describes the process of initializing the triple sequence table Euc_dist.

Algorithm 2 Initializing the triple sequence table

1: Initialize C_count$[1, 2, \ldots, m] \leftarrow 0$
2: **for** $i = 1$ to m **do**
3: **for** $j = i + 1$ to m **do**
4: Compute the lower bound between C_i, C_j
5: lower_bound$(C_i, C_j) \leftarrow |\text{Dist}[j] - \text{Dist}[i]|$
6: **if** lower_bound$(C_i, C_j) > R$ **then**
7: Break // This can reduce space dramatically // and is an important feature of our algorithm
8: **else**
9: Compute the actual distance between C_i, C_j
10: **if** the actual distance is smaller than R **then**
11: Assign related information to Euc_dist
12: C_count[i]++
13: C_count[j]++
14: **end if**
15: **end if**
16: **end for**
17: **end for**

In Algorithm 2, we use the triple sequence table to store the useful information that will be used for discovering motifs. The triple sequence table can

help save the storage space. Based on the symmetry, if $D(C_i, C_j)$ is smaller than R, $D(C_j, C_i)$ is certainly smaller than R. Because $D(C_i, C_j)$ is equal to $D(C_j, C_i)$, we assign only one value. Define an array C_count, and C_count[i] represents the number of subsequences whose distances to C_i are smaller than R. In the following algorithm, C_count will be used directly to make judgments.

Next, we explain lines 2–7 in Algorithm 2. We have calculated the distances between other subsequences and the reference subsequence. Thus, according to the values of these distances, the subsequences can be arranged in ascending order, followed by C_2, C_3, C_4, ... According to the illustration in Section 3.1, when lower_bound(C_i, C_j) $> R$, stop computing lower_bound(C_i, C_k) ($j < k < m$) and skip inner loop. This can reduce search time dramatically.

3.3 Executing the 1-motif discovery algorithm

We present an algorithm to discover 1-motif based on the matrix described in Section 3.2. Based on the values of the array C_count, we look for the elements with the largest value in C_count, and add the subsequences whose corresponding distance values in Euc_dist are not 0 into 1-motif. Of course, we may encounter the situation in which there are several elements with the largest value in C_count. For this situation, there exists a method that chooses the motif whose matches have the lower variance (Chiu *et al.*, 2003). This method, however, was not verified in Chu *et al.* (2003). In future work, we will verify the validity of this method, and try to find a better method to solve this problem.

Algorithm 3 describes the process of discovering 1-motif. In Algorithm 3, 1-motif_center represents the center of 1-motif. First, we look for the position where the value is the maximum in C_count (lines 2–6). Then we use the corresponding subsequence as the center of 1-motif (lines 7–8), and find all the subsequences whose distances to this center subsequence are smaller than R (lines 9–18). Finally, we regard all subsequences as 1-motif.

Algorithm 3 Discovering 1-motif

```
1: Initialize max ← 1
2: for i = 2 to m do
3:    if C_count[i] > C_count[max] then
4:       max ← i  // obtain the center of 1-motif
5:    end if
6: end for
7: 1-motif_center ← C_max
8: Add 1-motif_center to 1-motif
   // find in Euc_dist the subsequences whose distances
   // to T_max are smaller than R
9: for i = 1 to the length of Euc_dist do
10:   if Euc_dist[i].subr == max then
11:      k ← Euc_dist[i].col
12:      Add C_k to 1-motif
13:   end if
14:   if Euc_dist[i].col == max then
15:      k ← Euc_dist[i].subr
16:      Add C_k to 1-motif
17:   end if
18: end for
19: return 1-motif
```

3.4 Executing the K-motif discovery algorithm

After discovering 1-motif, we ignore the subsequences contained in 1-motif and do further mining based on the rest of the subsequences. In this subsection, we present an algorithm to discover K-motif based on our improved definition. Algorithm 4 describes the process of discovering K-motif.

In Algorithm 4, K-1-motifs represents the identified motifs, and K-motif_center represents the center of K-motif. Algorithm 4 compares the probable candidate motif center and other centers of the identified motifs (lines 20–30). If the distances are greater than $2R$, the candidate motif is regarded as the K-motif. If not, the actual distances are calculated from the center of the candidate motif to all subsequences in identified motifs. If all distances are greater than R, the candidate motif is regarded as the K-motif. If one of the distances is smaller than R, the candidate motif will be excluded.

When looking for the candidate K-motif, we need to adjust the corresponding array information. The subsequences that belong to the identified motifs will not be considered in the following judgment (lines 2–12). Thus, we make a further analysis on Euc_dist and C_count. Comparing the values of

Algorithm 4 Discovering K-motif

```
1: Initialize max ← 1
2: for k = 1 to the length of Euc_dist do
3:    i ← Euc_dist[k].subr
4:    if C_i is in K-1-motifs then
5:       Change the corresponding information of C_i
         in C_count and set the value of distance in
         Euc_dist[i] as 0
         // avoid interference for the following judgment
6:    else
7:       j ← Euc_dist[k].col
8:       if C_j is in K-1-motifs then
9:          Change the corresponding information of C_j
            in C_count and set the value of distance in
            Euc_dist[j] as 0
10:      end if
11:   end if
12: end for // find the K-motif
13: for i = 2 to m do
14:   if C_count[i] > C_count[max] then
15:      max ← i  // obtain the center of K-motif
16:   end if
17: end for
18: K-motif_center ← C_max
19: Add K-motif_center to K-motif
20: if the center C_max of the candidate motif does not
    meet Definition 5 then
21:   C_count[max] ← 0  // change the C_count
22:   Go to line 13  // continue the search
23: else
24:   for k = 1 to the length of Euc_dist do
25:      if Euc_dist[k].subr == max && Euc_dist[k].
         dist > 0 then
26:         j ← Euc_dist[k].col
27:         Add C_j to K-motif  // find in Euc_dist
            // the subsequences whose distances to
            // C_max are smaller than R
28:      end if
29:   end for
30: end if
31: return K-motif
```

C_count, we can determine the center of candidate K-motif (lines 13–18). Then judge whether the candidate K-motif shares the same subsequences with the identified motifs (line 20). If they share the same subsequences, this candidate K-motif is skipped. If not, find the subsequences whose distances to the center subsequence in the K-motif are smaller than R. All the subsequences found are seen as K-motif (lines 24–30).

4 Results and discussion

To verify the accuracy and efficiency of our algorithm, we studied two groups of publicly available datasets: Video Surveillance and EEG. Details about Video Surveillance and EEG datasets are available from the UCR (University of California, Riverside) time series data mining archive and http://www.cs.ucr.edu/∼mueen/MK/, respectively. Meanwhile, to verify the feasibility of our algorithm, we analyzed a Randomwalk dataset. In this section, we consider case studies on these time series datasets.

4.1 Motifs experiments and analysis

Based on the traditional definition and our improved definition of K-motifs, we reveal the results of three comparative experiments on three different time series datasets.

Experiment 1 (Randomwalk dataset) There are 40 000 subsequences of length 512. Fig. 3 gives a partial visualization of this dataset.

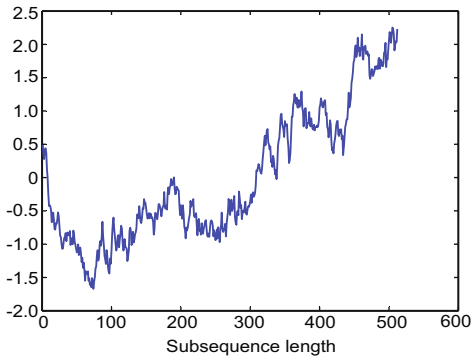

Fig. 3 Partial visualization of the Randomwalk dataset

First we used the motifs discovery algorithm based on the traditional definition of K-motifs. The 1- and 2-motif of this dataset are shown in Figs. 4 and 5a, respectively (R=5.40).

Then we used the motifs discovery algorithm based on our improved definition of K-motifs. The 1-motif is the same as that in Fig. 4, and the 2-motif is shown in Fig. 5b (R=5.40).

The 2-motif in Fig. 5b includes many more subsequences than that in Fig. 5a. Having discovered this interesting result, we continued to check if it is really significant. The first thing is to check whether the 2-motif shares the same subsequences with other motifs. According to our algorithm design, two

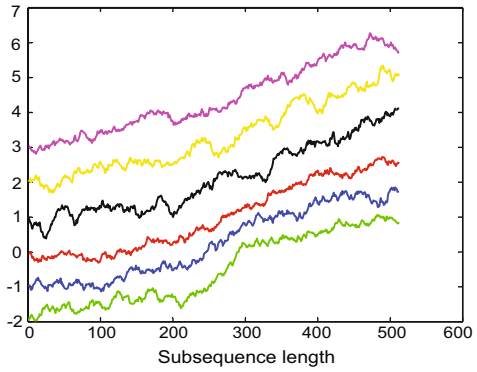

Fig. 4 The 1-motif of the Randomwalk dataset (R = 5.40)

motifs will not share the same subsequences. Moreover, each subsequence in Fig. 4 is not the same as the ones in Fig. 5b. The second thing is to examine the distance between any two subsequences. Our algorithm guarantees that the distances between any two subsequences in 2-motif are smaller than R; thus, the 2-motif in Fig. 5b is a meaningful result. By comparison, Fig. 5b includes many more frequently occurring patterns. The purpose of our algorithm is to discover many more frequently occurring patterns. Thus, the 2-motif in Fig. 5b is much more meaningful. We verify that the improvement on the definition of K-motifs can help find many more frequently occurring patterns in time series data, and the algorithm based on the traditional definition of K-motifs will miss some probable candidate motifs.

Experiment 2 (Video Surveillance dataset) First we used the algorithm based on the traditional definition of K-motifs to discover motifs. Figs. 6 and 7a show the 1- and 2-motif, respectively (R=14.00). Then we used the motifs discovery algorithm based on our improved definition of K-motifs to discover motifs. The 1-motif is the same as that in Fig. 6, and the 2-motif is shown in Fig. 7b (R = 14.00).

Comparison of Figs. 7a and 7b shows that the improvement on the definition of K-motif is meaningful.

Experiment 3 (EEG dataset) First we used the algorithm based on the traditional definition of K-motifs to discover motifs. Figs. 8 and 9a show the 1- and 2-motif of this dataset, respectively (R=16.52). Then we used the motifs discovery algorithm based on our improved definition of K-motifs to discover motifs. The 1-motif is the same as that in Fig. 8, and the 2-motif is shown in Fig. 9b (R = 16.52).

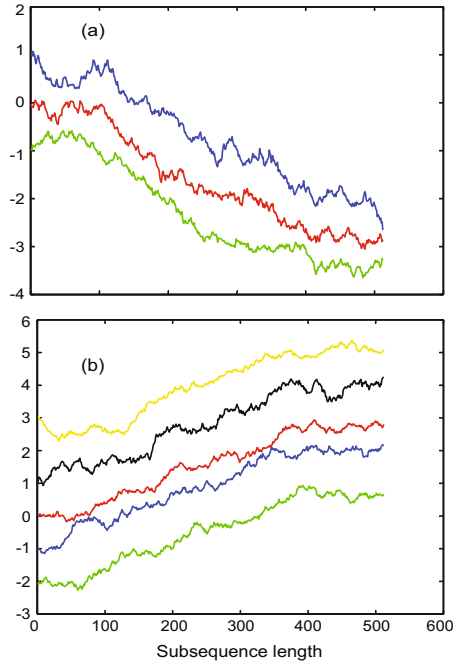

Fig. 5 The 2-motif of the Randomwalk dataset based on traditional (a) and our improved (b) definition of K-motifs (R = 5.40)

According to Figs. 9a and 9b, our improvement on the definition of K-motif is more convincing.

In conclusion, our algorithm based on the improved definition of K-motif is feasible and effective on different datasets and can help find many more frequently occurring patterns.

4.2 Efficiency analysis on improved triangular inequality

In our algorithm, we leverage triangular inequality to improve the efficiency. Based on the Randomwalk, Network, and Burst datasets, we

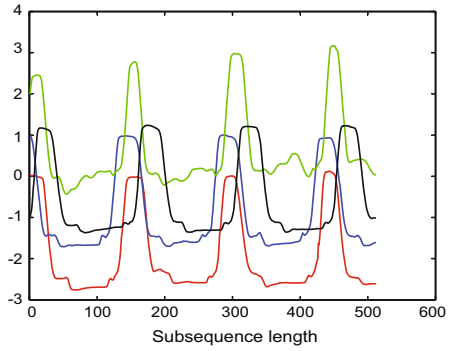

Fig. 6 The 1-motif of the Video Surveillance dataset (R = 14.00)

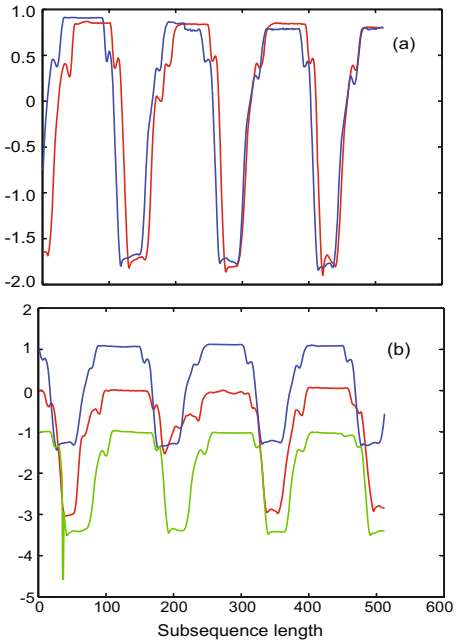

Fig. 7 The 2-motif of the Video Surveillance dataset based on traditional (a) and our improved (b) definition of K-motifs ($R = 14.00$)

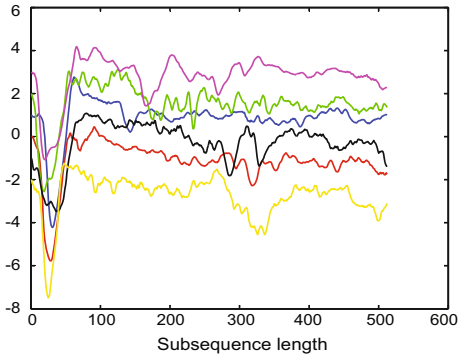

Fig. 8 The 1-motif of the EEG dataset ($R = 16.52$)

compared the time consumption of the triangular inequality pruning algorithm and our improved pruning algorithm. Details of the Network dataset and Burst dataset are available from the UCR time series data mining archive. We set R as 1.52 and obtained the corresponding time consumption. Table 2 shows the results.

Compared with the triangular inequality pruning method, our improved pruning algorithm reduces time consumption; e.g., in the Network dataset, the time reduction ratio is $(455.276-272.510)/455.276 = 0.401$. We can conclude that our improved pruning algorithm saves a large proportion of time and is much more effective than the triangular inequality pruning algorithm.

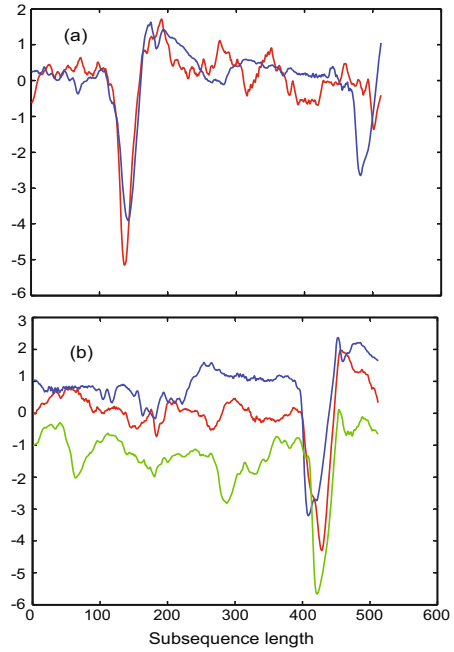

Fig. 9 The 2-motif of the EEG dataset based on traditional (a) and our improved (b) definition of K-motifs ($R = 16.52$)

Table 2 Comparison of time consumption on three datasets between our improved pruning method and the triangular inequality pruning method ($R = 1.52$)

Method	Time (s)		
	Network	Burst	Randomwalk
Our improved pruning method	272.510	258.617	195.953
Triangular inequality pruning method	455.276	385.117	244.250

5 Conclusions

We propose a new motifs discovery algorithm that is significantly more comprehensive and faster than the traditional motifs discovery algorithms. One contribution of this paper is to introduce the relatively comprehensive definition of K-motifs based on the traditional definition of K-motifs (Lin *et al.*, 2002), and this improvement can help avoid the omission of many more frequently occurring patterns. Another contribution is to improve the triangular inequality pruning method from a new perspective. We use this fast pruning method and the optimized matrix structure to realize fast motifs discovery. Results of experiments on three different time series datasets show that our algorithm is feasible and effective.

There is an important direction, i.e., the expression of motifs, along which we may extend this work. If the time series dataset is very large, the number of discovered motifs may be intimidating. In future work, we will investigate tools for visualizing and navigating the results of a motif search.

References

Abe, H., Ohsaki, M., Yokoi, H., Yamaguchi, T., 2005. Implementing an integrated time-series data mining environment based on temporal pattern extraction methods: a case study of an interferon therapy risk mining for chronic hepatitis. *LNCS*, **4012**:425-435. [doi:10.1007/11780496_45]

André-Jönsson, H., Badal, D.Z., 1997. Using Signature Files for Querying Time-Series Data. Practice of Knowledge Discovery in Databases, **1263**:211-220.

Beaudoin, P., Coros, S., van de Panne, M., Poulin, P., 2008. Motion-Motif Graphs. Symp. on Computer Animation, p.117-126.

Chiu, B.Y., Keogh, E.J., Lonardi, S., 2003. Probabilistic Discovery of Time Series Motifs. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.493-498.

Ding, H., Trajcevski, G., Scheuermann, P., Wang, X.Y., Keogh, E.J., 2008. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. Proc. Int. Conf. on Very Large Data Bases, **1**(2):1542-1552.

Ferreira, P.G., Azevedo, P.J., Silva, C.G., Brito, R.M.M., 2006. Mining approximate motifs in time series. *Discov. Sci.*, **4265**:89-101. [doi:10.1007/11893318_12]

Guyet, T., Garbay, C., Dojat, M., 2007. Knowledge construction from time series data using a collaborative exploration system. *J. Biomed. Inform.*, **40**(6):672-687. [doi:10.1016/j.jbi.2007.09.006]

Hegland, M., Clarke, W., Kahn, M., 2001. Mining the Macho dataset. *Comput. Phys. Commun.*, **142**(1-3):22-28. [doi:10.1016/S0010-4655(01)00307-1]

Lin, J., Keogh, E., Lonardi, S., Patel, P., 2002. Finding Motifs in Time Series. 2nd Workshop on Temporal Data Mining at the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.53-68.

Mueen, A., Keogh, E.J., Zhu, Q., Cash, S., Westover, M.B., 2009. Exact Discovery of Time Series Motifs. Society for Industrial and Applied Mathematics Conf. on Data Mining, p.473-484.

Tanaka, Y., Iwamoto, K., Uehara, K., 2005. Discovery of time-series motif from multi-dimensional data based on MDL principle. *Mach. Learn.*, **58**(2-3):269-300. [doi:10.1007/s10994-005-5829-2]

Ueno, K., Xi, X.P., Keogh, E.J., Lee, D.J., 2006. Anytime Classification Using the Nearest Neighbor Algorithm with Applications to Stream Mining. IEEE Int. Conf. on Data Mining, p.623-632.

Xu, X.K., Zhang, J., Small, M., 2008. Superfamily phenomena and motifs of networks induced from time series. *PNAS*, **105**(50):19601-19605. [doi:10.1073/pnas.0806082105]

Yi, B.K., Faloutsos, C., 2000. Fast Time Sequence Indexing for Arbitrary L_p Norms. Int. Conf. on Very Large Data Bases, p.385-394.

Zhang, J., Small, M., 2006. Complex network from pseudoperiodic time series: topology versus dynamics. *Phys. Rev. Lett.*, **96**:238701. [doi:10.1103/PhysRevLett.96.238701]