Int J Data Sci Anal (2017) 4:113–126
DOI 10.1007/s41060-017-0060-3

REGULAR PAPER

A novel clustering-based method for time series motif discovery under time warping measure

Cao Duy Truong[1] · **Duong Tuan Anh**[1]

Received: 13 December 2016 / Accepted: 8 June 2017 / Published online: 30 June 2017

Abstract The problem of time series motif discovery has attracted a lot of attention and is useful in many real-world applications. However, most of the proposed methods so far use Euclidean distance to deal with this problem. There has been one proposed method, called MDTW_WedgeTree, for time series motif discovery under DTW distance. But this method aims to deal with the case in which motif is the time series in a time series database which has the highest count of its similar time series within a range r. To adapt the above-mentioned method to the case in which motifs are frequently occurring subsequences of a longer time series, we modify MDTW_WedgeTree to a new algorithm for discovering "subsequence" motifs in time series under DTW. The proposed method consists of a segmentation method to divide the time series into motif candidates and a BIRCH-based clustering which can efficiently cluster motif candidate subsequences under DTW distance. Experimental results showed that our proposed method for discovering "subsequence" motifs performs very efficiently on large time series datasets while brings out high accuracy.

Keywords Time series · Subsequence motif · Motif discovery · Dynamic time warping

✉ Duong Tuan Anh
dtanh@cse.hcmut.edu.vn

Cao Duy Truong
caoduytruong@hcmunre.edu.vn

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam

1 Introduction

A time series is a sequence of real numbers measured at equal time intervals. Time series data arise in so many applications of various areas ranging from business, finance, medicine, meteorology, environment to government. One of the important representative patterns in time series is time series motif. One commonly used definition of time series motif is that motifs are frequently occurring but previously unknown subsequences of a longer time series. Motif discovery has been used in various areas, for instance, telecommunications, medicine, web, motion capture and sensor networks. Some well-known algorithms for motif discovery are Random Projection by Chiu et al. [1]; EMD by Tanaka et al. [22] which applies Minimum Description Length principle; MK by Mueen et al. [17]; an algorithm by Gruber et al. [5] which is based on segmentation and clustering; an algorithm by Castro and Azevedo [2] which exploits iSAX representation multiresolution capability to obtain motifs at different resolutions; GrammarViz by Li et al. [15] which is based on grammar inference and a method by Son and Anh ref2n which is based on a multidimensional index. However, most of the proposed methods so far use Euclidean distance to deal with this problem.

Euclidean distance is not a suitable measure for working on time series in various fields, for example, multimedia data. Ding et al. [3] pointed that dynamic time warping (DTW) is the best measure for various kinds of time series data. Nevertheless, it has been suggested many times in the literature that DTW incurs high computational complexity and that is the main obstacle of using DTW in time series data mining tasks. There have been several research works in speeding up the computation of DTW distance. Most recently, Rakthanmanon et al. [19] introduced a suite of techniques for speeding up DTW similarity search on very large time series

datasets (consisting of trillions of time series subsequences). However, the complexity of similarity search on N time series subsequences is just about $O(N)$, while the complexity of motif discovery on N time series subsequences will cost about $O(N^2)$. Due to this difficulty, so far there is little research work on time series motif discovery under DTW distance.

The first attempt to deal with time series motif discovery under DTW distance is the method proposed by Truong and Anh [23]. This method, called MDTW_WedgeTree, utilizes UCR, a suite of techniques to speed up DTW similarity search [19], and a clustering-based technique to discover motifs in a time series database. The clustering technique used in this motif discovery method employs a data structure called Wedge-Tree to facilitate the time series clustering under DTW distance. But MDTW_WedgeTree aims to deal with the case in which motif is the time series in a time series database which has the highest count of its similar time series within a range r. With this definition of time series motif, motif discovery is based on *whole sequence matching*.

To adapt the above-mentioned method to the case in which motifs are frequently occurring but previously unknown subsequences of a longer time series, in this paper, we modify MDTW_WedgeTree method to a new algorithm for discovering "subsequence" motifs in time series under DTW distance. The proposed method consists of the following enhancements:

- We identify significant points in the time series and extract subsequences based on these significant points.
- We develop a BIRCH-based clustering that can efficiently cluster subsequences under DTW distance.
- In clustering, we propose a technique of finding cluster representatives to improve the quality of clustering.

Experimental results revealed that our proposed method for discovering "subsequence" motifs under DTW distance can perform very fast on large time series datasets while brings out high accuracy.

The remainder of the paper is organized as follows. Section 2 provides the problem definition. Section 3 introduces some backgrounds on DTW distance, MDTW_ WedgeTree method, BIRCH algorithm, and significant points in time series. Section 3 explains our proposed method for discovering motif in time series under DTW distance. Section 5 reports the experiments. Some discussion is given in Sect. s6. Finally, Sect. 7 gives some conclusions and future works.

2 Problem definition

In order to describe the problem of motif discovery, we give the related definitions which are from [1,16] as follows.

Definition 1 (*Time Series*) A time series $T = t_1, \ldots, t_n$ is an ordered set of n real values measured at equal intervals.

Definition 2 (*Subsequence*) Given a time series T of length n, a subsequence S of T is a sampling of length $m < n$ of contiguous positions from T, that is, $S = t_p, \ldots, t_{p+m-1}$ for $1 < p < n - m + 1$.

Definition 3 (*Distance*) *Dist* is a function as: $Dist(C, M) \rightarrow \Re$ wherein the inputs are two subsequences C and M, the function returns a nonnegative number, called as the distance from C to M. *Dist* function must hold symmetry.

Definition 4 (*Match*) Given a positive real number r (called range) and a time series T containing a subsequence C beginning at position p and a subsequence M beginning at q, if the distance $Dist(C, M) \leq r$, then M is called a matching subsequence of C.

Any definition of motif should exclude the possibility of overcounting the trivial matches, which are defined below.

Definition 5 (*Trivial match*) Given a time series T, containing a subsequence C beginning at position p and a matching subsequence M beginning at q, we say that M is a trivial match to C if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $Dist(C, M') > r$ and either $q < q' < p$ or $p < q' < q$.

Definition 6 (*1-motif*) Given a time series T, a subsequence length m and a range r, the most significant motif in T (called the *1-motif*) is the subsequence C_1 that has the highest count of nontrivial matches.

The Kth most significant motif in T (called *K-motif*) is the subsequence C_k that has the highest count of nontrivial matches, and satisfies $Dist(C_K, C_i) > 2r$, for all $1 \leq i \leq K$.

All subsequences that are similar to the *K-motif* are called instances of that *K-motif*. Note that Definition 6 forces the sets of instances of the motifs to be mutually exclusive.

The motif definition given in this section indicates that in this work, we deal with discovering "subsequence" motif in time series rather than "whole sequence" motif as in [23].

3 Background

3.1 Dynamic time warping

Given two time series Q and C, both of length n, to compare these two time series using DTW, an $n \times n$ matrix is constructed, where the (ith, jth) element of the matrix is the distance $dist(q_i, c_j) = (q_i - c_i)^2$. A *warping path* P is a contiguous set of matrix elements that defines a mapping between Q and C. The tth element of P is defined as $p_t = (i, j)_t$, so we have:

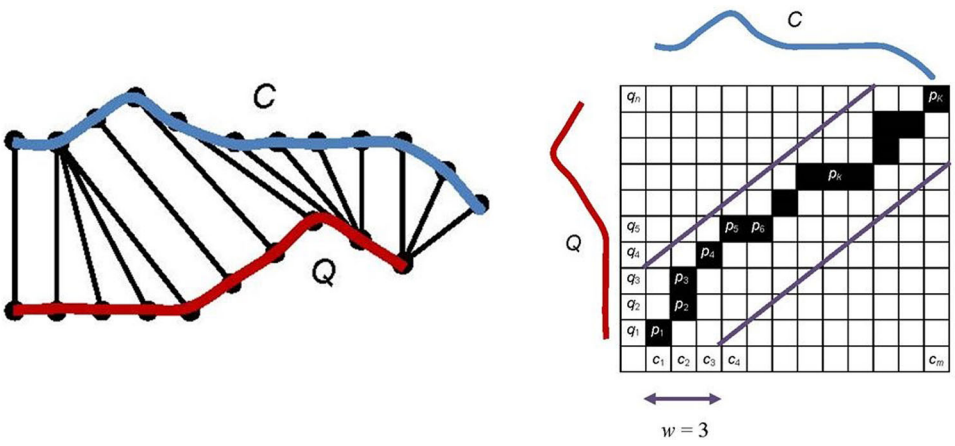

Fig. 1 (*Left*) Two time series Q and C are aligned under DTW distance. Note that Sakoe–Chiba band with width w is used to constrain the warping path (*right*)

$$P = p_1, p_2, \ldots, p_t, \ldots, p_T \quad n \leq T < 2n - 1$$

Given two time series, the *DTW Distance* between them is the minimal cost of all warping paths:

$$ED(Q, C) = min\left(\sqrt{\sum_{i=1}^{T} p_t}\right)$$

The Euclidean distance between two time series can be seen as a special case of DTW distance where the kth element of P is constrained such that $p_k = (i, j)_k, i = j = k$.

To speed up the DTW distance calculation, all practitioners using DTW constrain the warping path in a global manner by limiting how far it may stray from the diagonal. The subset of matrix that the warping path is allowed to visit is called *warping window* or *a band*. Two of the most frequently used global constraints in the literature are the Sakoe–Chiba band proposed by Sakoe and Chiba [20] and Itakura Parallelogram proposed by Itakura [8]. Sakoe–Chiba band is the area defined by two straight lines in parallel with the diagonal (see Fig. 1), and Itakura Parallelogram is the area defined by the parallelogram which is symmetric over the diagonal.

In this work, we use the Sakoe–Chiba band with width w. As mentioned in [19] that similarity search on the time series with different lengths using DTW is very hard and meaningless, in our work, if the lengths of time series are different, we apply some interpolation technique to convert them to the same length before doing motif discovery.

3.2 Speeding up DTW computation

In this section, we describe a suite of techniques, called UCR suite, to speed up the computation of DTW distance as well as to reduce the number of times invoking the DTW distance computation. These techniques were introduced by Rakthanmanon et al. [19]. In our work, we make use of some major ideas from UCR suite as follows.

Using the squared distance Both DTW and ED have a square root calculation. However, as suggested in [19] if we omit this step, it does not change the relative rankings of nearest neighbors. In this work, we will still use DTW and ED; however, the reader may assume that we mean the squared versions of them.

Lower bounding To speed up the computation of DTW distance between two time series Q and C, besides using warping window, researchers proposed some cheap-to-compute lower-bound teTherefore, during the computationchniques [9,13,14,25]. In a DTW lower bounding technique, the lower bound of the DTW distance between two time series Q and C, denoted by $LB(Q, C)$, must satisfy the following inequality: $LB(Q, C) \leq DTW(Q, C)$.

The computation cost of $LB(Q, C)$ is much less than that of $DTW(Q, C)$. If $LB(Q, C) > r$, then $DTW(Q, C) > r$ and therefore we do not have to compute $DTW\ (Q, C)$ and conclude that the two time series Q and C cannot be similar to each other.

Kim et al. [13] proposed a lower bounding technique for DTW called *LB_Kim*. *LB_Kim* extracts a four-tuple feature vector from each time series. The features are the first and last data points of the time series, together with the maximum and minimum values. The maximum squared differences of corresponding features are used as the lower bound. The complexity of *LB_Kim* is $O(n)$. Rakthanmanon et al. [19] improved *LB_Kim* to *LB_KimFL*, with the complexity $O(1)$. *LB_KimFL* uses only the distances between the first (last) pair of points from both time series. In this work, we also use the *LB_KimFL* as proposed in [19].

One of the most popular lower bounding techniques for DTW is *LB_Keogh* [9]. *LB_Keogh* has the complexity $O(n)$. Rakthanmanon et al. [19] use *LB_Keogh* in similarity search on very large time series. Lemire [14] proposed the lower bounding technique called *LB_Improved* which is an improved variant of *LB_-Keogh*. Lemire has proved that *LB_Improved* is a tighter lower bound than *LB_Keogh*.

Instead of using *LB_Keogh* to speed up DTW distance computation, in this work we use *LB_Improved*.

We use the two lower bounding techniques (*LB_Kim-FL* and *LB_Improved*) in a cascade. We first use the *LB_KimFL*. If a candidate is not pruned at this stage, we turn to the *LB_Improved*. This technique is called *cascading lower bounds*.

LB reversing Since $LB(Q, C) \neq LB(C, Q)$, after checking whether $LB(Q, C) < r$ or not, if we cannot prune off a candidate, we can continue to compute $LB(C, Q)$ in the same way given in [19].

Look ahead of DTW For $1 < k < n$, the following inequality will be satisfied:

$$DTW(Q_{1:k}, C_{1:k}) + LB(Q_{k+1:n}, C_{k+1:n}) \leq DTW(Q_{1:n}, C_{1:n})[18]$$

Therefore, during the computation of the distance $DTW(Q, C)$ at any time point k, if $DTW(Q_{1:k}, C_{1:k}) + LB(Q_{k+1:n}, C_{k+1:n}) > r$, then we can conclude that the two time series Q and C cannot be similar to each other and stop the computation of $DTW(Q, C)$ at that time point.

Early abandoning During the computation of the lower bound *LB* or $Dist(Q, C)$, if we note that the current sum of the squared differences between each pair of corresponding data points exceeds the range r, we can stop the calculation at that data point and conclude that $Dist(Q, C) > r$ and Q and C are not similar to each other. We apply this techniques in all steps which check whether a distance satisfies a given distance threshold r.

Besides the above-mentioned techniques, Truong and Anh [23] devised an upper-bound technique to speed up DTW computation which is described as follows.

Upper bound by Euclidean distance Note that $ED(Q, C)$ is a special case of $DTW(Q, C)$, when the warping band w is set to 1. We have the following inequality: $DTW(Q, C) \leq ED(Q, C)$.

If $ED(Q, C)$ is within the range r, then we have $DTW(Q, C) \leq r$. $ED(Q, C)$ can be used as an upper bound of $DTW(Q, C)$. So the computation of $DTW(Q, C)$ includes the computation of $ED(Q, C)$ at the point $p_k = (i, j)_k, i = j = k$. To speed up the computation of DTW, we first compute $ED(Q, C)$ and during the process of computing $ED(Q, C)$ we store the values $p_k = (i, j)_k, i = j = k$ which may be used in the later computation of $DTW(Q, C)$ if necessary. When the computation of $ED(Q, C)$ finishes, if $ED(Q, C) \leq r$, then we can stop the computation and conclude that Q, C are similar.

Look ahead of Euclidean distance Similar to the look ahead of DTW, we have the following inequality:

$$DTW(Q_{1:n}, C_{1:n}) \leq DTW(Q_{1:k}, C_{1:k}) + ED(Q_{k+1:n}, C_{k+1:n}) \quad 1 < k < n$$

Therefore, during the computation of the distance $DTW(Q, C)$ at any time point k, if $DTW(Q_{1:k}, C_{1:k}) + ED(Q_{k+1:n}, C_{k+1:n}) \leq r$, then we can conclude that the two time series Q and C can be similar to each other and stop the computation of $DTW(Q, C)$ at that time point.

3.3 MDTW_WedgeTree method

In this section, we review briefly the MDTW_WedgeTree method [23] for discovering "whole sequence" motifs in a time series database. This method is based on clustering. To mitigate the difficulty of time series clustering under DTW distance, the method uses the support of Wedge-Tree.

Wedge and Wedge-Tree Given a set of time series $C_{set} = C_1, \ldots, C_k$, we can form two new time series U and L using the following formula:

$$U_i = max(C_{1i}, \ldots, C_{ki})$$
$$L_i = min(C_{1i}, \ldots, C_{ki})$$

U and L stand for upper and lower bounds of C_{set}, respectively. They form the smallest possible bounding envelope that encloses all members of the set C_{set} from above and below. As in the work [12], we will call the combination U, L a $wedge$ and denote a wedge as W: $W = \{U, L\}$. We have the following inequality:

$$DTW(C_i, C_j) \leq ED(C_i, C_j) \leq ED(U, L), \quad \forall C_i, C_j \subset C_{set}$$

For convenience, we denote $W_{UL} = ED(U, L)$.

The expansion of $W = \{U, L\}$ of C_{set} when inserting a new time series C to C_{set} can be done simply as follows: $U_i = max(U_i, C_i)$ and $L_i = min(L_i, C_i)$. Similarly, when we have two sets of time series: C_{set1}, with $W_1 = \{U_1, L_1\}$ and C_{set2}, with $W_2 = \{U_2, L_2\}$, then the wedge $W = \{U, L\}$ of $C_{set1} \cup C_{set2}$ can be computed as follows: $U_i = max(U_{1i}, U_{2i})$ and $L_i = min(L_{1i}, L_{2i})$.

Now we construct a tree structure based on the wedges $W = \{U, L\}$, which is called Wedge-Tree. The structure of Wedge-Tree is almost similar to that of R^*_Tree. At each entry in a node, Wedge-Tree stores a wedge W which is the bounding envelope of all child nodes of this entry. There are two parameters in a Wedge-Tree: B, the maximum number of entries in a nonleaf node and r, the distance threshold W_{UL}

of all entries in a leaf node, i.e., all entries in a leaf node must satisfy $W_{UL} \leq r$.

Insertion of a time series C to Wedge-Tree is as follows. At each level of the Wedge-Tree, we insert C to the entry whose wedge W, $ED(U, L)$, needs the least enlargement to include that time series. Insertion of a time series C to an entry at a leaf node can be done only when the enlargement satisfies the condition $W_{UL} \leq r$; otherwise, we have to create a new entry at that leaf node.

The node splitting due to a violation of the threshold B can be done by selecting the two entries E_1 and E_2 in a node such that the wedge W of $E_1 \cup E_2$ is the largest and then based on E_1 and E_2, we split the node in question into two different nodes.

The MDTW_WedgeTree algorithm Based on Wedge-Tree, an algorithm for motif discovery in a time series database $D = \{T_1, T_2, \ldots, T_m\}$ was proposed [23]. This algorithm is called MDTW_WedgeTree (Motif discovery under DTW using Wedge Tree). The algorithm consists of the four following steps:

1. Step 1: [*Constructing Wedge-Tree*] The algorithm builds the Wedge-Tree with two parameters B and r by repeatedly inserting time series T_i to the Wedge-Tree. After constructing the Wedge-Tree, we obtain the subclusters corresponding to the entries at the leaf nodes whose wedges satisfy the condition $W_{UL} \leq r$. That means the time series in the same subcluster are similar to each other.
2. Step 2: [*Merging the subclusters*] In this step, we consider every pair of subclusters $S_i \cup S_j$ and compute W of the set $S_i \cup S_j$. If W satisfies the threshold r, $W_{UL} \leq r$, then we merge the two subclusters to a bigger subcluster, $S_i \cup S_j$.
3. Step 3: [*Enriching the subclusters*] During the process of building the Wedge-Tree, since the subclusters must be mutually exclusive and due to the order of insertion of T_i to the tree, some inaccuracies might exist: two similar time series might be inserted to different subclusters. Therefore, in this phase, we consider each pair of subclusters obtained after Step 2, S_i and S_j, to check whether there exist some instances in S_i that can be inserted to S_j, and vice versa. If that is the case, some instances can be migrated from one subcluster to another subcluster.
4. Step 4: [*Discovering motifs*] After Step 3, we sort the subclusters in the descending order of the number of instances and remove the overlapping subclusters. There may exist overlapped instances in two subclusters. If the overlap percentage is above a certain threshold (e.g., we use 90% overlap), the overlapped subclusters are merged. The subcluster with the highest number of instances will be the *1-motif*, and the subcluster with the kth highest number of instances will be the *k-motif*.

In Step 3, we utilize the lower-bound measure between an arbitrary time series Q to the subcluster S proposed by Keogh et al. [12]. This lower-bound measure can be described as follows.

For each subcluster S, based on the wedge $W = \{U, L\}$ of S and the warping range w, we form two new subsequences DTW_U and DTW_L:

$$DTW\_U_i = max(U_{i-w} : U_{i+w})$$
$$DTW\_L_i = min(L_{i-w} : L_{i+w})$$

Keogh et al. gave the lower bounding measure between an arbitrary subsequence Q to the subcluster S as follows:

$$LB\_Keogh_{DTW}(Q, W) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - DTW\_U_i)^2 & \text{if } q_i > DTW\_U_i \\ (q_i - DTW\_L_i)^2 & \text{if } q_i > DTW\_L_i \\ 0 & \text{otherwise} \end{cases}}$$

and proved that:

$$LB\_Keogh_{DTW}(Q, W) \leq DTW(Q, C_k), \forall C_k \subset S$$

Therefore, if we have $LB\_Keogh_{DTW}(Q, W) > r$, then we can conclude that there are no subsequences in the subcluster S that is similar to Q. In the MDTW_ WedgeTree algorithm, the early abandon technique is applied in all the steps. One weakness of MDTW_Wedge-Tree is that it lacks a principled method to determine subcluster *representatives*. The algorithm selects randomly one instance in a subcluster as its representative. This stochastic factor might reduce the effectiveness of Step 2 (Merging the subclusters) and Step 3 (Enriching the subclusters) in the method.

3.4 Extracting significant points

Detecting change points in a given time series is an important time series data mining task. Change points are the time positions in the original time series where the local trend in the data values has changed. The main goal of time series segmentation is the extraction of segments with similar observations or different from the rest of the time series. Here, in time series segmentation we want that the start point and the end point of a segment should be change points. We extract the change points based on the combination of important extreme points [6] and turning points [4].

The algorithm for finding important extreme points was first introduced by Pratt and Fink [18]. Fink and Gandhi [6] proposed the improved variant of the algorithm for finding important extreme points. Important extreme points are divided into two kinds: important minima and important

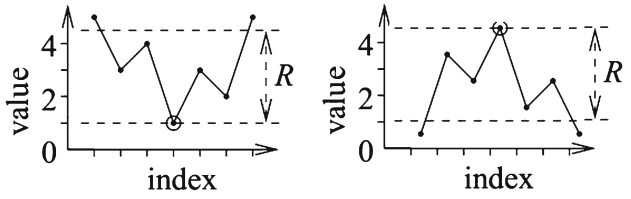

Fig. 2 Illustration of important extreme points: **a** minimum, **b** maximum

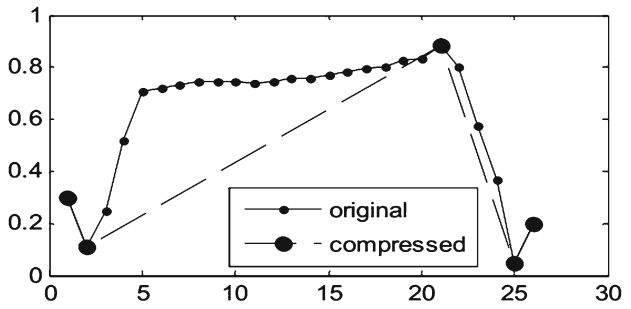

Fig. 3 Representing the original series only by important extrema [4]

maxima. Figure 2 illustrates the definition of important minima (a) and important maxima (b).

We control the compression rate with a positive parameter R; an increase of R leads to the selection of fewer points. Fink and Gandhi [6] proposed some functions to compute the distance between two data values a, b, in a time series. In this work we use the distance $dist(a, b) = |a - b|$.

Definition 7 (*Important minimum*) For a given distance function dist and positive value R, a point t_i of a series $t_1, \ldots, t_n$ is an important minimum if there are indices i_l and i_r, where $i_l < i < i_r$, such that t_i is a minimum among $t_{il}, \ldots, t_{ir}$, and $dist(t_i, t_{il}) \geqslant R$ and $dist(t_i, t_{ir}) \geqslant R$.

Definition 8 (*Important maximum*) For a given distance function dist and positive value R, a point t_i of a series $t_1, \ldots, t_n$ is an important maximum if there are indices i_l and i_r, where $i_l < i < i_r$, such that t_i is a maximum among $t_{il}, \ldots, t_{ir}$, and $dist(t_i, t_{il}) \geqslant R$ and $dist(t_i, t_{ir}) \geqslant R$.

The algorithm for extracting important extrema of all types from a time series of length n is given by Fink and Gandhi [6]. The algorithm takes linear time and constant memory.

Fu et al. [4] proposed a method for identifying the change points in a time series which are called turning points. Fu et al. notice that choosing the change points in a time series only from important extrema may not be enough (see Fig. 3). To overcome the imperfection of important extreme point method, Fu et al. proposed the concept of turning points defined as follows.

Definition 9 *Turning point*: For a given positive value R and time series $T = t_1, \ldots, t_n$. If t_j is a previous turning point

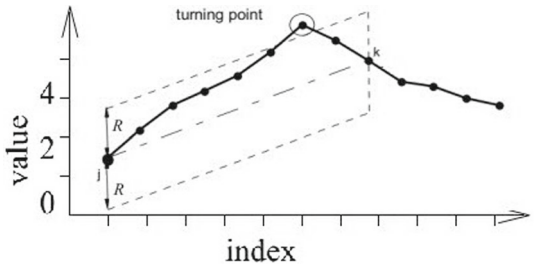

Fig. 4 Turning point [4]

found in time series T at position j, $1 \leq j \leq n$, for each point t_k, $(k > j)$, three slopes are calculated as follows:

$$s_c(k) = \frac{t_k - t_j}{k - j}$$

$$s_u(k) = \frac{t_k - (t_j + R)}{k - j}$$

$$s_l(k) = \frac{t_k - (t_j - R)}{k - j}$$

t_k is considered as a turning point, if $s_c(k) > min(s_l(i))$ or $s_c(k) < max(s_u(i))$ where $i = j + 1, j + 2, \ldots, k - 1$.

In the definition of turning points, R represents a given error bound. The method for identifying turning points is like drawing a thicker and smoother curve that can cover the original curve. For two points (p, t_p) and (q, t_q) where $p < q$, the area between the two lines that joins $(p, t_p + R)$ to $(q, t_p + R)$ and $(p, t_p - R)$ to $(q, t_q - R)$ forms a thick line. We say we can connect two points with a thick line if and only if all the points between the two points are falling in the area of the thick line.

The algorithm for identifying the turning points in a time series is given in [4] as follows:

1. Step 1. The first point of time series T is chosen as a turning point.
2. Step 2. Let (j, t_j) be the newly chosen turning point. Obviously, we can connect (j, t_j) and $(j + 1, t_{j+1})$ with a thick line. Then we grow the length of the line one by one. If we can connect (j, t_j) and $(j + 1, t_{j+1})$ with a thick line and cannot connect (j, t_j) and (k, t_k) where $j + 1 < k < n$, with a thick line, we select the last point (only one point at most time) which is between t_j and t_k and outside the thick line as a new turning point (see Fig. 4) Repeat this step until the last point of T.
3. Step 3. The last point is also a turning point.

The algorithm proposed by Pratt and Fink [18] for identifying the important extrema in some cases ignores some turning points in the time series. This makes the compressed time series not to preserve the shape of the original time

series (see Fig. 3). On the contrary, the algorithm proposed by Fu et al. [4] for identifying the turning points in some cases does not bring out the turning points which match with the important extreme points.

In order to find sufficiently the change points in a time series, in our work, we combine two above-mentioned methods for identifying the change points. We identify not only important extrema but also the turning points in a time series. At each point in a time series, we check whether it is a turning point. If this is not the case, then we check whether it is an extreme point. This algorithm needs one scan through the whole time series.

3.5 BIRCH clustering

BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering at the initial stage and other clustering methods, such as iterative partitioning at the later stage [26]. It introduces two main concepts, *clustering feature* and *clustering feature tree (CF tree)*, which are used to summarize cluster representations. These structures help the clustering method achieve good speed and scalability in large databases. BIRCH is also effective for *incremental* and *dynamic* clustering of incoming objects.

Given N d-dimensional points or objects x_i in a cluster, we can define the centroid x_0 and the radius R of the cluster as follows:

$$x_0 = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$R = \sqrt{\frac{\sum_{i=1}^{N} (x_0 - x_i)^2}{N}}$$

where R is the average distance from member objects to the centroid; R reflects the tightness of the cluster around the centroid. A clustering feature (CF) is a triplet summarizing information about clusters of objects. Given N d-dimensional points or objects in a subcluster, then the CF of the cluster is defined as (N, LS, SS) where N is the number of points in the subcluster, LS is the linear sum on N points, and SS is the square sum of data points.

Given the CF for a cluster, centroid and radius R may be computed. Clustering features are additive. For example, suppose that we have two disjoint clusters, C_1 and C_2, having the clustering features, CF_1 and CF_2, respectively. The clustering feature for the cluster that is formed by merging C_1 and C_2 is simply $CF_1 + CF_2$. Clustering features are sufficient for calculating all of the measurements that are needed for making clustering decisions in BIRCH.

A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering. By definition, a nonterminal node in the tree has descendents or "children." The nonleaf nodes store sums of the CFs of their children and thus summarize clustering information about their children. Each entry in a leaf node is not a single data objects but a subcluster. A CF tree has two parameters: branching factor (B for nonleaf node and L for leaf node) and threshold r. The branching factor specifies the maximum number of children in each nonleaf or leaf node. The threshold parameter specifies the maximum radius of the subcluster stored at the leaf nodes of the tree. The two parameters influence the size of the resulting tree.

BIRCH applies a multiphase clustering technique: a single scan of the dataset yields a basic good clustering, and one or more additional scans can (optionally) be used to further improve the quality.

4 The proposed method

The proposed method is a modification of MDTW_ WedgeTree algorithm for discovering "subsequence" motifs in a long time series under DTW. Besides using a combination of several techniques to speed up DTW computation, our proposed motif discovery method is also based on clustering as in MDTW_WedgeTree algorithm. But there are some major improvements that we include in our proposed method:

(i) We identify significant points in the time series and extract subsequences based on these significant points. This technique is necessary since we have to work with motif candidates that are subsequences extracted from a long time series.
(ii) Instead of using a clustering method with the support of Wedge-Tree, we develop a BIRCH-based clustering which can effectively and efficiently cluster subsequences under DTW distance.
(iii) In clustering, we propose a technique of finding cluster representatives to improve the quality of clustering.

Besides UCR suite of techniques, our proposed method exploits upper-bound technique with Euclidean distance [23] and reference point technique [17] to speed up DTW computation as much as it can.

The proposed method for discovering "subsequence" motif in time series under DTW consists of the following phases:

1. Phase 1: [*Identifying significant points*] We identify all the significant points in a time series T by combining important extreme point method and turning point method as explained in Sect. 3.4.
2. Phase 2: [*Extracting subsequences*] Based on the sequence of the significant points found in Phase 1, we extract the subsequences that satisfy some conditions on the

length and the number of feature points specified by user. Candidate subsequences are subsequences that may have different lengths. To enable the computation of Euclidean or DTW distance between them, we bring them to the same length by using some interpolation method.

3. Phase 3: [*Clustering subsequences*] In this phase, we cluster the subsequences based on Euclidean distance. Note that $ED(Q, C)$ is a special case of $DTW\ (Q, C)$, when the warping band w is set to 1. We have the following inequality:

$$DTW(Q, C) \leq ED(Q, C)$$

If $ED(Q, C)$ is within the range r, then we have $DTW(Q, C) \leq r$. Therefore, if any subsequence in a cluster satisfies the distance threshold r according to Euclidean distance, then it also satisfies the distance threshold r according to DTW distance.
4. Phase 4: [*Finding subcluster representatives*] After Phase 3, we select the representatives for all the subclusters. Besides, depending on the clustering algorithm, there may be some defects in Phase 3: some instances that are not similar to the representative of the subcluster can belong to the subcluster. Therefore in Phase 4, we remove all the instances that are not similar within r threshold to the subcluster representative. We repeat Phase 3 and Phase 4 until we remove all these misclustered instances.
5. Phase 5: [*Enriching subclusters*] Due to the property $DTW(Q, C) \leq ED(Q, C)$, there may be several instances that do not belong to a subcluster but satisfy the distance threshold r to the subcluster representative under DTW distance. Therefore, we have to check all this kind of instances and insert them into the subcluster if necessary.
6. Phase 6: [*Evaluating subclusters*] We evaluate the subclusters based on motif scores in order to discover motifs in the time series. For convenience, we call our method SPs_BIRCH_DTW (motif discovery using Significant Points with BIRCH algorithm and under DTW distance).

4.1 Extracting of subsequences and building a set of subsequences

The significant points can be used as the start points or end points of motif candidates. We identify all the significant points in a time series T by combining important extreme point method and turning point method as explained in Sect. 3.4. Then we extract the subsequences from a time series using the significant points and based on the common length of subsequences which is specified by user in finding motif (from L_{min} to L_{max}). The subsequence extraction algorithm is described as follows.

The subsequence extraction algorithm works by anchoring the start point of a potential subsequence at the first significant point of a time series and then attempting to extend the subsequence to the right with increasing number of significant points. At some $(i+1)$th significant points, the length of the potential subsequence is greater than the user-specified length, so the subsequence from the anchor to the ith significant point is transformed to a subsequence. The anchor is moved to the second significant point, the algorithm tries to extract the second subsequence, and the process repeats until the entire time series has been transformed into a sequence of subsequences.

The extracted subsequences may have different lengths. In order to be able to compare these subsequences under Euclidean distance or DTW distance, we apply homothetic transformation to convert these subsequences to the same length.

Transforming subsequences Homothety is a simple and effective technique which also can transform the subsequences with different lengths to those of the same length. Homothety is a transformation in affine space. Given a point O and a value $k \neq 0$. A homothety with center O and ratio k transforms M to M' such that $\overrightarrow{OM'} = k \times \overrightarrow{OM}$

The algorithm that performs homothety to transform a subsequence T with length N ($T = \{Y_1, \ldots, Y_N\}$) to subsequence of length N' ($T' = \{Y'_1, \ldots, Y'_N\}$) is given as follows.

1. Let $Y\_Max = Max\{Y_1, \ldots, Y_N\}; Y\_Min = Min\{Y_1, \ldots, Y_N\}$
2. Find a center I of the homothety with the coordinate: $X\_Center = N/2, Y\_Center = (Y\_Max + Y\_Min)/2$
3. Perform the homothety with center I and ratio $k = N'/N$, where the value of T' at time point i, Y'_i, is equal to the value of T at the time point:

$$j = \frac{i - X\_Center}{k} + X\_Center$$

The homothety is a simple transformation with low computational complexity. However, when $N' \ll N$, it may cause some information loss and cannot preserve the shapes of the original subsequences. Therefore, in this work we perform homothetic transform with $N' > N$. In Phase 2 of SPs_BIRCH_DTW, we transform all the candidate subsequences to the same length of the largest subsequence.

4.2 Clustering subsequences with BIRCH and refining the clustering results

By using BIRCH algorithm for clustering the subsequences, we obtain the resulting subclusters in which all the subse-

quences that are similar to each other within the threshold r under Euclidean distance form a subcluster. Since the Euclidean distance is always greater than DTW distance, there may be some subsequences that belong to other subclusters but can be similar within r threshold of DTW distance to the subsequences in the subcluster under consideration.

Since we have applied BIRCH algorithm under Euclidean distance for clustering the subsequences, we need to refine the BIRCH algorithm in Phase 3 and Phase 4 in order to make its resulting subclusters consistent with DTW distance. In Phase 3, we merge the two subclusters in which the DTW distance between their centroids is less than or equal to the threshold r. In Phase 4, instead of using the centroids, we will use the representative of each subcluster in checking whether there exist some subsequences that do not belong to the subcluster but may have the DTW distance to the representative of the subcluster within the r threshold. If that is the case, we insert these subsequences to the correct subcluster.

4.3 Finding representatives of subclusters

In SPs_BIRCH_DTW, to improve the quality of clustering, we devise a principled method to determine representatives of subclusters, as described as follows.

Definition 10 (*Representative of a subcluster*) Given a subcluster $S = C_1, \ldots, C_k$, instance C_i, $1 \leq i \leq k$, is *representative* of the subcluster if the distance from C_i to its farthest neighbor in the subcluster is smallest, i.e., $max(DTW(C_i, MC_i)) < max(DTW(C_j, MC_j))$, $1 \leq j \leq k$ where MC_i is a neighbor of C_i.

In order to speed up the process of finding subcluster representatives, we apply *early abandoning* technique (described in Section 3.2) as follows: let $minDis$ denote the smallest DTW distance to the farthest neighbor of the subcluster representative found so far.

At the index j, if $DTW(C_j, C_q) > minDis$ (where C_q is a neighbor of C_j) then C_j cannot be the representative of the subcluster, so we stop the process of checking C_j. In order to apply early abandoning technique earlier, we sort $C_1, \ldots, C_k$ in ascending order of Euclidean distances from each of them to $C_{centroid}$ (the centroid of S), and at the index j, we compute $DTW(C_j, C_q)$, where q varies from k to 1.

In the process of finding the farthest neighbor of C_j, we apply the reference point technique [17] as follows. Let $maxDis_j$ denote the DTW distance from C_j to its farthest neighbor found so far; we have the following inequality:

$$DTW(C_j, C_q) \leq ED(C_j, C_q) \leq ED(C_j, C_{centroid}) + ED(C_q, C_{centroid})$$

Therefore, at the current index, if $ED(C_j, C_{centroid}) + ED(C_q, C_{centroid}) \leq maxDis_j$, we should stop the process of finding the farthest neighbor of C_j at the instance with the index q since C_q and any C_k where $1 \leq k < q$ cannot be the farthest neighbor of C_j. The pseudocode of the algorithm for finding the representative of a subcluster is described in Table 1.

Table 1 Algorithm for finding the representative

FindRepresentive $(S = s_1, s_2, \ldots s_N)$

```
1.  C_centroid= centroid of the cluster S
2.  Sort all subsequences in S in ascending
    order of Euclidean Distance from
    ech subsequence to C_centroid
3.  C_S = s_1
4.  best-so-far = Inf
5.  for i = 1 to N
6.    maxDTWNeighbor_i = 0
7.    for j = N to 1
8.      if (ED(s_i, C_centroid) + ED(s_j, C_centroid))
            <= maxDTWNeighbor_i break
9.      if LB(s_i, s_j) > best-so-far break
10.       if ED(s_i, s_j) <= maxDTWNeighbor_i
              continue
11.     if DTW(s_i, s_j) >best-so-far break
12.     if DTW(s_i, s_j) > maxDTWNeighbor_i
13.       maxDTWNeighbor_i = DTW(s_i, s_j)
14.     if maxDTWNeighbor_i <best-so-far
15.       best - so - far = maxDTWNeighbor_i
16.       C_S = s_i
17. return C_S
```

In the functions $LB(s_i, s_j)$, $ED(s_i, s_j)$, $DTW(s_i, s_j)$ we also apply early abandoning technique. That means it is possible to abandon the lower bound, the Euclidean distance or DTW distance computation as soon as the cumulative sum goes beyond the current *best-so-far*.

During the process of finding the representative for subcluster S, we also identify the instances that do not satisfy the r threshold to the subcluster representative. We remove all of these instances out of the subcluster and form a new subcluster for them. We perform again BIRCH algorithm and finding the representatives for the new subclusters formed from these instances.

4.4 Enriching the subclusters based on DTW distance

For a pair of subclusters S_i and S_j, checking whether any instances of S_i can be included in S_j, and vice versa can be done as follows.

1. In each subcluster S_i, we find the representative of this subcluster. This time series is denoted as C_S_i.
2. Based on LB_Keogh lower bound (described in Section 3.3), we check whether any instance in the subcluster S_j is similar to C_S_i. If that is the case, we apply the speedup techniques for DTW computation by lower bound and

upper bound. If there is any instance in S_j similar to C_S_i, we insert it to S_i.

In this step we also apply the *reference point* technique to speed up the computation. In the process of finding the representative of a subcluster, we have sorted the instances in subcluster S_j in increasing order of the Euclidean distance from each instance to $C_{centroid}\_S_j$ (the centroid of S_j). And we have the following inequality:

$$DTW(C\_S_i, C_q) \leq ED(C\_S_i, C_q)$$
$$\leq ED(C\_S_i, C_{centroid}\_S_j) + ED(C_q, C_{centroid}\_S_j), \forall C_q \text{ in } S_j$$

Therefore, at some index q, if $ED(C\_S_i, C_{centroid}\_S_j) + ED(C_q, C_{centroid}\_S_j) \leq r$, we can stop the process and have $DTW(C\_S_i, C_k) \leq r$, where $1 \leq k \leq q$. That means all such instances C_k can be included in the subcluster S_i.

Remove Trivial Matches In our SPs_BIRCH_DTW method, trivial matches may arise since we extract motif candidates with the different lengths varying from L_{min} to L_{max}. These trivial matches can cause negative effects on the clustering quality. To exclude possible trivial matches, after obtaining the subclusters by using BIRCH algorithm, we examine all the subclusters to exclude any instances which have some overlap in one another more than 90% data points.

4.5 Evaluating subclusters

To discover *1-motif* in a time series, after clustering the subsequences, we evaluate the subclusters using motif score. The definition of motif score is given as follows.

Definition 11 *Motif score* For each subcluster S_i, motif score of S_i, $mscore_i$, is calculated by the formula:

$$mscore_i = m_1 \times a - m_2 \times b + m_3 \times c$$

where a is the number of subsequences in S_i (normalized to the range [0,1]); b is W_{UL} of the subcluster S_i (normalized to the range [0,1]); c is the length of the longest subsequence in S_i; m_1, m_2, m_3 are the weights specified by user.

After computing *motif score* for each subcluster, the subcluster with the highest motif score will contain the instances of the *1-motif*. Similarly, we can determine the subcluster with the kth highest motif score will contain the instances of the *k-motif* .

5 Experimental evaluation

We evaluate the performance of our proposed method for discovering motif under DTW in several publicly available time series datasets downloaded from the web pages [7,10,11]. The datasets are from different areas (medicine, engineering, industry). Among the eight datasets used in the experiments, TEK16 and TEK17 are sensor time series, representing normal Space Shuttle Marotta Valve Time Series annotated by NASA engineer. There are five electrocardiogram datasets: ECG (stdb308), ECG(qtdbsele0606), ECG (chfdbchf15), ECG (mitdbx_108), and Koski-ECG. Power dataset is the data on power demand. The names and lengths of the eight datasets are given in Table 1. All these datasets are normalized by *z-normalization*.

Table 2 Datasets

Dataset	Length	No. Sig. Pts	No. subs
TEK 16	4992	46	58
TEK 17	5000	43	56
ECG (stdb308)	5400	153	281
ECG (qtdbsele0606)	15,000	797	4488
ECG (chfdbchf15)	15,000	466	731
ECG (mitdbx_108)	21,600	683	2299
Power	35,040	1446	5258
Koski-ECG	144,404	2587	3613

We implemented the two algorithms with Microsoft Visual C# and conducted the experiments on an Intel Core i7 6700HQ 2.6Hz, RAM 8GB PC. We evaluate our proposed method based on two performance metrics: effectiveness and efficiency.

Our proposed method requires from user 6 parameters: L_{min} and L_{max} (the minimum and maximum length for each motif candidate), R (compression rate for computing important extreme points or turning points), r (the range threshold for matching subsequences), B (the branching factor of a nonterminal node in CF tree), and the warping range w (Sakoe–Chiba band) for DTW.

For all the datasets, we select $R = 1$ for compression rate, $L_{min} = 200$, $L_{max} = 300$. We conduct the experiments on the datasets with different range thresholds r. We used some different values of r, ranging from $0.01L_{max}$ to $0.1L_{max}$. We select $B = 1000$, $w = 0.05L_{max}$. To evaluate the motif scores, we select $m_1 = 1, m_2 = 0.1$ and $m_3 = 0.01$.

Table 2 shows the test datasets and the results of segmentation for each dataset. The number of identified significant points and the number of extracted subsequences for each dataset are reported in the third column and the fourth column.

5.1 Accuracy

In this experiment, following the tradition established in previous works, such as [1,16], we use the brute-force algorithm given by Lin et al. [16], as the baseline to evaluate the accu-

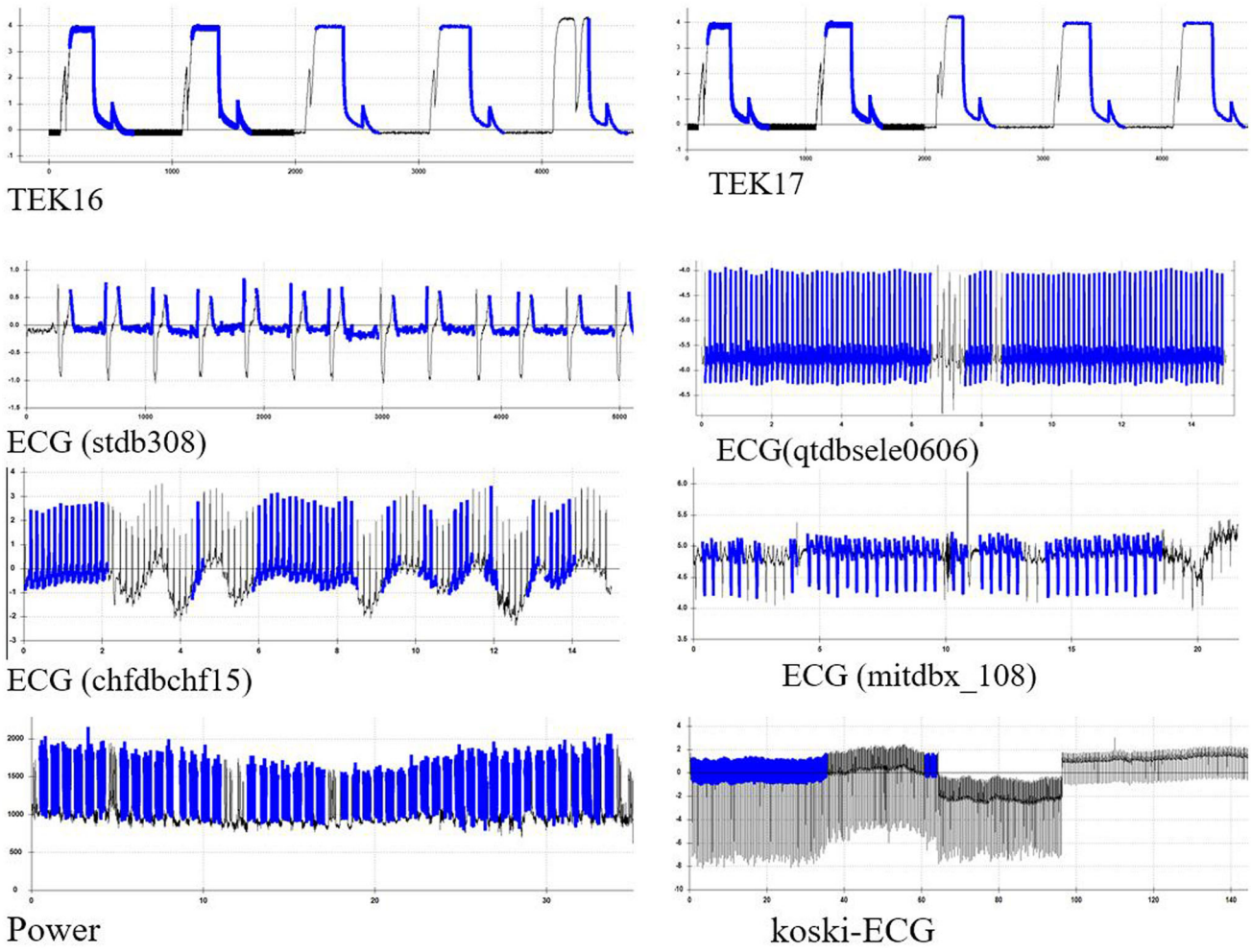

Fig. 5 The *1-motif* (in *blue*) discovered by our method in some datasets

racy of the proposed method. The brute force is an exhaustive search algorithm which guarantees to catch all true *1-motif* patterns. This algorithm can be implemented by a two-layer nested loop. The outer loop considers each possible candidate subsequence, and the inner loop is a linear scan to identify the nontrivial matches of the candidate within a range r. We also apply DTW distance in this brute-force algorithm rather than Euclidean distance.

Experimental results show that for each dataset, the *1-motif* discovered by SPs_BIRCH_DTW is exactly the same as the *1-motif* discovered by the brute-force algorithm. Figure 5 shows the 1-motifs discovered by SPs_BIRCH_DTW in some datasets (TEK 16, TEK 17, ECG (stdb308), ECG (qtdbsele0606), ECG (chfdbchf15), ECG (mitdbx_108), Power, Koski-ECG).

Besides, for each dataset we check whether the instances of the *1-motif* discovered by SPs_BIRCH_DTW are the same as those of the *1-motif* found by the brute-force algorithm or not. The experimental results show that for each dataset, the overlap (intersection) of the two sets of motif instances discovered by the two methods reaches to about 90%.

5.2 Efficiency

Since our proposed method is the first attempt in "subsequence" motif discovery in time series with DTW distance, we use the brute-force algorithm given in [16], as the baseline to evaluate the efficiency of the proposed algorithm. We evaluate the efficiency of the motif discovery algorithms by considering the ratio of how many times of the distance function must be invoked by the proposed algorithm over the number of times it must be called by the brute-force algorithm. However, in this work we compute the efficiency ratio by using the number of times $dist(q_i, c_j)$ is invoked (see Definitions 3, 4) rather than the number of times $Dist(T_i, T_j)$ is called as in some previous works [16,24]. This new way of computation is especially suitable in the case that we have to compute the lower bound *LB* as well as the upper bound *ED* and these computational costs are significant.

Table 3 log(*Eff*) of SPs_BIRCH_DTW

Dataset	0.02	0.04	0.06	0.08	0.1
TEK 16	5.15	5.23	5.21	5.21	5.25
TEK 17	5.18	5.20	5.24	5.19	5.20
ECG (stdb308)	4.03	4.11	4.17	4.18	4.24
ECG (qtdbsele0606)	3.18	3.28	3.28	3.33	3.37
ECG (chfdbchf15)	4.62	4.67	4.61	4.64	4.64
ECG (mitdbx_108)	3.88	3.90	3.92	3.93	3.97
Power	3.79	3.85	3.95	4.01	4.06
Koski-ECG	5.88	5.92	5.77	5.67	5.88

The efficiency ratio of a motif discovery algorithm, *Eff*, is defined as follows:

$$Eff = \frac{\text{number calls of } dist(q_i, q_j) \text{ in } SPs\_BIRCH\_DTW}{\text{number calls of } dist(q_i, q_j) \text{ in } bruteforce}$$

The range of the efficiency ratio is from 0 to 1. The algorithm with lower efficiency ratio is better. Since the number of calls of $dist(q_i, c_j)$ function is very large and the value of *Eff* is small, we represent the efficiency ratio *Eff* by using log(*Eff*).

Table 3 reports log(*Eff*) of SPs_BIRCH_DTW for range r varying from $0.02L_{max}$ to $0.1L_{max}$ against the brute-force algorithm in finding motif with the length of $0.1L_{max}$.

From the experimental results in Table 4, we can see that in comparison with the brute-force algorithm, SPs_BIRCH_DTW has the average value of log(*Eff*) at 4.52 that means SPs_BIRCH_DTW runs faster than the brute-force algorithm about $10^{4.52}$ times.

We also measure the runtimes of SPs_BIRCH_DTW over 8 dataset and for different values of range r. The results of CPU time (in seconds) on 8 datasets are shown in Table 3. From the results in Table 3, we can see that SPs_BIRCH_DTW is very time efficient. The runtime of SPs_BIRCH_DTW for Koski-ECG, the largest dataset with 144,404 data points, is just less than 30 seconds. Due to the support of BIRCH clustering and a combination of several techniques to speed up DTW computation, our proposed method can work with large time series datasets. With the experimental results in Table 4, it is obvious that DTW distance can be competitive with Euclidean distance in time series motif discovery.

Besides, we can see that the efficiency ratio and runtime of SPs_BIRCH_DTW do not change much with different values of the range r. When r is large, SPs_BIRCH_DTW can apply early abandoning by using Euclidean distance as upper bound. On the contrary, when r is small, SPs_BIRCH_DTW can apply early abandoning by using lower-bound values. The length and number of instances of *1-motif* discovered by SPs_BIRCH_DTW over each dataset for different values of the range r are reported in Table 5. We can see that for larger values of r, the number of instances of *1-motif* increases and for different values of r, the instances of *1-motif* can be different in length and locations.

Table 4 Runtime (seconds) of SPs_BIRCH_DTW

Dataset	0.02	0.04	0.06	0.08	0.1
TEK 16	0.50	0.49	0.57	0.48	0.64
TEK 17	0.64	0.56	0.52	0.76	0.70
ECG (stdb308)	1.91	1.50	1.71	1.40	1.36
ECG (qtdbsele0606)	37.52	31.51	31.79	31.16	26.69
ECG (chfdbchf15)	2.83	2.52	2.93	2.53	2.90
ECG (mitdbx_108)	23.62	21.57	20.46	18.8	17.51
Power	90.64	63.97	52.21	45.99	41.30
Koski-ECG	18.80	17.88	23.41	28.04	18.83

Table 5 The length and number of instances of *1-motif* found for different values of range

Dataset	0.02	0.04	0.06	0.08	0.1
TEK 16	258	258	258	258	258
	#05	#05	#05	#05	#05
TEK 17	259	259	259	259	259
	#05	#05	#05	#05	#05
ECG (stdb308)	287	287	293	293	293
	#12	#13	#13	#13	#13
ECG (qtdbsele0606)	282	230	282	271	235
	#92	#92	#93	#93	#93
ECG (chfdbchf15)	291	293	283	291	284
	#19	#28	#32	#37	#43
ECG (mitdbx_108)	269	254	253	269	260
	#23	#30	#33	#37	#39
Power	242	208	214	208	242
	#50	#90	#118	#126	#136
Koski-ECG	272	209	295	266	266
	#91	#99	#100	#157	#195

5.3 Comparing SPs_BIRCH_DTW to MDTW_WedgeTree

For completeness, we attempt to compare SPs_BIRCH_DTW to MDTW_WedgeTree even though MDTW_WedgeTree aims to discover "whole sequence" motif, while SPs_BIRCH_DTW aims to discover subsequence motif. To make the comparison meaningful, given a time series, we apply Phase 1 and Phase 2 in SPs_BIRCH_DTW to extract subsequences of the same length from the time series and we form a database of short time series from these subsequences. We continue to apply the Phases 3 to 6 of SPs_BIRCH_DTWs on the extracted subsequences to obtain

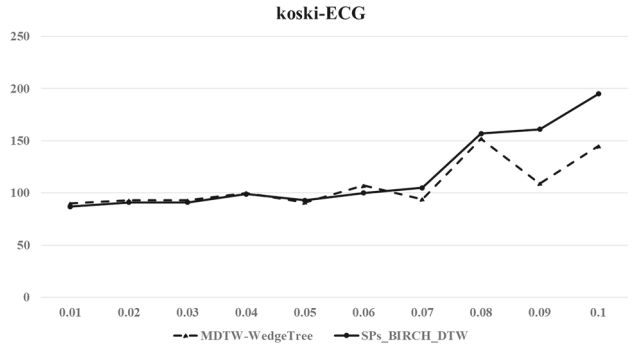

Fig. 6 The number of the instances of the *1-motif* discovered by the two algorithms when the range r varies

the *1-motif*. Then, we apply MDTW_WedgeTree on the database of short time series to obtain the *1-motif* in its own way.

We conducted the comparison of the two algorithms over the above-mentioned eight datasets. The experimental results show that the *1-motif* discovered by MDTW_WedgeTree is approximately the same as the *1-motif* discovered by SPs_BIRCH_DTW. However, when the range r increases, the number of instances of *1-motif* in both algorithms increases, but the change in MDTW_WedgeTree is random and unstable, while the change in SPs_BIRCH_DTW is more stable and robust. Figure 6 shows the number of the instances of the *1-motif* discovered by the two algorithms on the Koski-ECG dataset for range r varying from $0.01L_{max}$ to $0.1L_{max}$. In Fig. 6, the curve of SPs_BIRCH_DTW indicates a smoothly gradual increase, while the curve of MDTW_WedgeTree is a randomly abrupt variation. We attribute this weakness of MDTW_WedgeTree to the fact that MDTW_WedgeTree selects randomly one instance in a subcluster as its representative and this stochastic selection might reduce the robustness of the algorithm.

6 Discussion

Unlike several previous works on time series motif discovery, in SPs_BIRCH_DTW, we do not require the user to supply one important parameter: the length of motif, which is unknown and very difficult to determine. We mitigate the difficulty of setting the motif length by replacing it with the three other easy parameters, R, L_{min}, and L_{max} in the time series segmentation method which is based on significant points.

It is possible to extend SPs_BIRCH_DTW so that the proposed method can discover motifs and anomaly patterns in a time series at the same time. This can be done by applying the following extension in Phase 6 of SPs_BIRCH_DTW. We define the anomaly score for each subcluster found in BIRCH clustering and calculate these scores immediately after calculating the motif scores for all subclusters. The subcluster with the highest anomaly score will contain the most significant anomaly pattern (*1-anomaly*) in the time series.

7 Conclusion

In this paper, we modify MDTW_WedgeTree, a method for discovering "whole sequence" motif, to a new method for discovering "subsequence" motif in time series under DTW distance. This method first extracts subsequences from a time series based on significant points and then uses BIRCH to cluster these subsequences. For each subcluster, the method determines the representative of the subcluster and enriches the subcluster based on DTW distance. Finally, the method evaluates motif score for each subcluster in order to discover *1-motif* in the time series. In order to accelerate the similarity search under DTW distance, the method applies not only UCR suite of techniques but also upper bounding technique using Euclidean distance and reference point technique.

Experimental results reveal that our proposed method can run very fast and bring out motifs with high accuracy on large time series datasets. We attribute the high performance of our proposed method to the efficiency of the chosen segmentation method, the BIRCH-based clustering of extracted subsequences under DTW, and a combination of techniques to speed up DTW computation.

For future work, we plan to modify our two methods, MDTW_WedgeTree or SPs_BIRCH_DTW, so that they can discover motifs in database of shapes.

Acknowledgements We are grateful to Prof. Eamonn J. Keogh for kindly providing necessary datasets and related code for this research work. This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under Grant Number C2016-20-04.

References

1. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proceedings of 9th International Conference on Knowledge Discovery and Data Mining (KDD 2003), pp. 493–498 (2003)
2. Castro, N., Azevedo, P.: Multiresolution motif discovery in time series. In: Proceedings of the SIAM International Conference on Data Mining, April 29–May 1, Columbus, Ohio, USA, pp. 665–676 (2010)
3. Ding, H., Wang, X., Mueen, A., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representations and distance measures for time series data. Data Min. Knowl. Discov. **26**, 275–309 (2013)
4. Fu, Y., Lu, H., Zhou, J., Hu, X., Deng, Y.: An intuitive time series compression method. J. Comput. Inf. Syst. **6**(1), 189–195 (2010)
5. Gruber, C., Coduro, M., Sick, B.: Signature verification with dynamic RBF network and time series motifs. In: Proceedings

of the 10th International Workshop on Frontiers in Hand Writing Recognition (2006)
6. Fink, E., Gandhi, H.S.: Important extrema of time series. In: Proceedings of the IEEE International Conference on System, Man and Cybernetics, Montreal, Canada, pp. 366–372 (2007)
7. http://physionet.org/physiobank/
8. Itakura, F.: Minimum prediction residual principle applied to speech recognition. IEEE Trans. Acoust. Speech Signal Process. **23**, 67–72 (1975)
9. Keogh, E.: Exact indexing of dynamic time warping. In: Proceeding of the 28th International Conference on Very Large Databases (VLDB 2002) (2002)
10. Keogh, E.: Shape matching website. http://www.cs.ucr.edu/~eamonn/shape/shape.htm
11. Keogh, E.: UCR Time Series Classification/Clustering Page website. http://www.cs.ucr.edu/~eamonn/time_series_data
12. Keogh, E., Wei, L., Xi, X., Lee, S.H., Vlachos, M.: LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In: Proceedings of the 32th International Conference on Very Large Data Bases (VLDB 2006) (2006)
13. Kim, S., Park, S., Chu, W.: An index-based approach for similarity search supporting time warping in large sequence databases. In: Proceedings of the 17th International Conference on Data Engineering (ICDE 2001) (2001)
14. Lemire, D.: Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognit. **42**(9), 2169–2180 (2009)
15. Li, Y., Lin, J., Oates, T.: Visualizing variable-length time series motifs. In: Proceedings of the SDM (2012)
16. Lin, J., Keogh, E., Patel, P., Lonardi, S.: Finding motifs in time series. In: Proceedings of the 2nd Workshop on Temporal Data Mining, The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002) (2002)
17. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact discovery of time series motif. In: Proceedings of the 2009 SIAM International Conference on Data Mining (SIAM 2009) (2009)
18. Pratt, K.B., Fink, E.: Search for patterns in compressed time series. Int. J. Image Graph. **2**(1), 89–106 (2002)
19. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012) (2012)
20. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978)
21. Son, N.T., Anh, D.T.: Discovery of time series k-motifs based on multidimensional index. Knowl. Inf. Syst. **46**, 59–86 (2016)
22. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time series motif from multi-dimensional data based on MDL principle. Mach. Learn. **58**(2–3), 269–300 (2005)
23. Truong, C.D., Anh, D.T.: A fast method for motif discovery in large time series database under dynamic time warping. In: Van-Nam-Huynh et al. (Eds.) Proceedings of the 6th International Conference on Knowledge and Systems Engineering, 9-11 October, Hanoi, Advances in Intelligent Systems and Computing, Vol. 326, pp. 155–168. Springer (2014)
24. Xi, X., Keogh, E., Li, W., Mafra-neto, A.: Finding motifs in a database of shapes. In: Proceedings of the 7th SIAM International Conference on Data Mining (SDM) (2007)
25. Yi, B., Jagadish, H., Faloutsos, C.: Efficient retrieval of similar time sequences under time warping. In: Proceedings of the 14th International Conference on Data Engineering (ICDE 1998) (1998)
26. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1996)