# ภาคผนวก

## ผลงานตีพิมพ์

Knowledge-Based Systems 35 (2012) 361–368

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Short communication

# Selective Subsequence Time Series clustering

Sura Rodpongpun, Vit Niennattrakul, Chotirat Ann Ratanamahatana *

*Department of Computer Engineering, Chulalongkorn University, 254 Phayathai Road, Pathumwan, Bangkok 10330, Thailand*

ARTICLE INFO

*Article history:*
Received 11 December 2011
Received in revised form 20 April 2012
Accepted 22 April 2012
Available online 27 April 2012

*Keywords:*
Time series
Subsequence clustering
STS clustering
Meaningful time series clustering
Time series mining

ABSTRACT

Subsequence Time Series (STS) Clustering is a time series mining task used to discover clusters of interesting subsequences in time series data. Many research works had used this algorithm as a subroutine in rule discovery, indexing, classification and anomaly detection. Unfortunately, recent work has demonstrated that almost all of the STS clustering algorithms give meaningless results, as their outputs are always produced in sine wave form, and do not associate with actual patterns of the input data. Consequently, algorithms that use the results from the STS clustering as their input will fail to produce its meaningful output. In this work, we propose a new STS clustering framework for time series data called Selective Subsequence Time Series (SSTS) clustering which provides meaningful results by using an idea of data encoding to cluster only essential subsequences. Furthermore, our algorithm also automatically determines an appropriate number of clusters without user's intervention.



## 1. Introduction

Time series clustering [2,6,15,13,24] is one of the most popular tasks in time series data mining community [5,16,18,25,26,20]. Most algorithms generally perform whole time series clustering [24,15]. More specifically, those algorithms try to group individual time series instances to a set of clusters. On the other hand, Subsequence Time Series (STS) clustering [2,13,6], which will be considered in this work, has been gaining more popularity. STS clustering algorithm discovers clusters of interesting subsequences within a single time series data stream. This algorithm can be used as a subroutine of other data mining tasks, such as rule discovery [28,4,12], indexing [17], classification [3], and anomaly detection [28].

Unfortunately, it has been demonstrated that these STS clustering algorithms produce meaningless results [13]. Because most algorithms use a sliding window to extract subsequences and try to cluster them all, the resulting cluster centers turn out to be some forms of sine waves regardless of the original shape of the patterns in the input data. Therefore, every algorithm that uses this meaningless STS clustering as a subroutine will in turn fail to produce meaningful results as well.

The cause of producing sine waves as outputs has been analyzed by many authors [8,11,21]. They have shown that clustering of every single subsequence leads to meaningless outputs. In fact, some subsequences such as noises or outliers should not be clustered. For instance, consider a speech recognition problem, non-speech segments in a source data has to be determined and removed. Similarly, in sign language recognition, transitions between consecutive signs, called movement epenthesis [29,22], has to be discarded. We show in Fig. 1 that meaningful STS clustering can be achieved by ignoring some subsequences. We use an ECG data from [9] to demonstrate that it is not necessary to include some trivial subsequences in a cluster.

In this work, we propose a new STS clustering framework called Selective Subsequence Time Series (SSTS) clustering, which performs subsequence clustering to produce meaningful cluster centers. We will show that the cluster centers from our algorithm do represent the actual patterns within the input data, instead of producing sine waves. In essence, we adopt an idea of data encoding to determine proper clusters by clustering only important subsequences. Some subsequences that are not significant will be discarded. On the other hand, because it is hard to exactly specify window size of the subsequences, our approach allows window size to be varied. The appropriate sliding window size, $w$, depends on types of data and application requirement. In practice, a user only need to roughly estimate a value of $w$, and then our algorithm will determine an appropriate value. However, due to the flexible window length $w$, the members of clusters could be of different lengths. Moreover, different types of data need different predefined number of clusters $k$, so our algorithm automatically determines an appropriate number of clusters depending on characteristics of input data.

The rest of this paper is organized as follows. In Section 2, we provide review and discussion of some related works. Section 3 offers background knowledge used in this work. Detail of our approach are described in Section 4. Section 5 shows essential experiments in various domains including real and synthetic data. Finally, conclusion and discussion about future research direction are discussed in Section 6.

* Corresponding author. Tel.: +66 8 9499 9400; fax: +66 2 218 6955.
*E-mail addresses:* g53srd@cp.eng.chula.ac.th (S. Rodpongpun), g49vnn@cp.eng.chula.ac.th (V. Niennattrakul), ann@cp.eng.chula.ac.th (C.A. Ratanamahatana).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.04.022

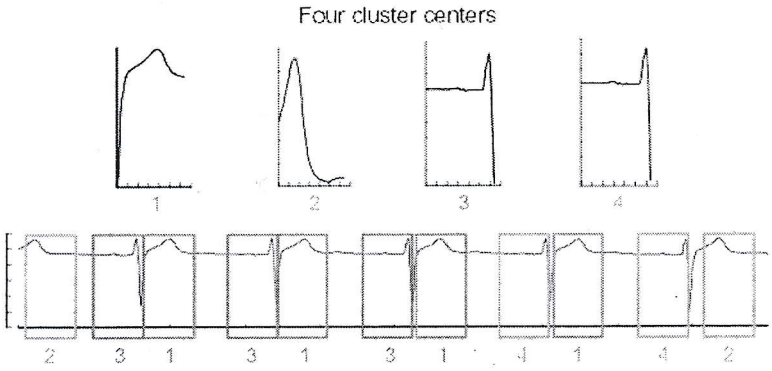



**Fig. 1.** Meaningful STS clustering achieved by ignoring some subsequences.

## 2. Related work

In time series mining communities, clustering task has always been receiving much attention. However, most works focus on clustering individual time series whereas clustering subsequences of a single long time series, the problem considered in this work, is not well resolved. The most referenced work is the one that uses STS clustering as a subroutine for rule discovery [4]. Until recently, it has been discovered that the STS clustering used in that work is meaningless [13]. Because they try to cluster every single extracted subsequences, their output turns out to always be in sine waves regardless of what the input sequence looks like. This problem has then been analyzed by many authors [8,11,21]. It now comes to a conclusion that clustering every subsequence extracted by moving a fix sliding window leads the cluster centers to converge to sine waves. Many authors have proposed algorithms to solve this problem [2,6], but those algorithms require a fixed value of number of clusters $k$ and length of subsequences $w$, which are not suitable in real world problems.

## 3. Background

In this section, we provide definitions and background knowledge used in this work.

### 3.1. Definition and notation

**Definition 1.** A *time series* $T$ of size $m$ is an ordered sequence of real value data, where $T = (t_1, t_2, \ldots, t_m)$.

Our approach takes a sequence of time series $T$ as an input and extracts it to a set of *subsequences*.

**Definition 2.** A *subsequence* of length $n$ of time series $T$ is $T_{i,n} = (t_i, t_{i+1}, \ldots, t_{i+n-1})$, where $1 \leqslant i \leqslant m - n + 1$, $n < m$.

As mentioned in Section 1, we adopt an idea of simple data encryption to determine proper clusters by emulating the clusters as a *codebook*.

**Definition 3.** A *codebook* is a data structure used to store codewords, representing repeating parts in an input data. The input data can be compressed by substituting the repeating parts with smaller codeword symbols. In this work, we emulate cluster centers as the codewords used to represent their member subsequences. Performance of the encoding can be measured by using *Compression ratio* and *Error* defined below.

**Definition 4.** *Compression ratio* is a ratio of the data size between after and before compression, including an overhead of construction of a codebook and codeword symbols. For example, given a 16-charecter string $S$ = "*ABCDEFGHIJKLMNOP*". Suppose that substrings "*ABC*" and "*HIJ*" are similar, we can substitute them with a symbol $x$, therefore the encoded string $S'$ = "*xDEFGxKLMNOP*". In this case, we can eliminate 6 characters ("*ABC*" and "*HIJ*"), but a codeword of size 3, and two $x$'s must be created; thus, the compression is $6 - (3+2) = 1$ character, and the *compression ratio* $\bar{R} = \frac{16-1}{16} = 0.94$.

**Definition 5.** *Error* can be obtained by calculating summation of distances from cluster centers to their cluster members. For example, the two substrings "*ABC*" and "*HIJ*", which are mentioned in the previous definition, are grouped into a cluster $C$, then a codeword (a cluster center) $\bar{C}$ is created. The *Error* of creating a cluster from those substrings is $\bar{E}(C) = Dist(\bar{C}, ABC) + Dist(\bar{C}, HIJ)$.

Next, we summarize background knowledge used in this paper.

### 3.2. Euclidean distance measure

To measure distance between two subsequences, we use Euclidean distance that has been widely used in time series domain. The distance is as shown

$$Dist(X_{i,n}, X_{j,n}) = \sqrt{\sum_{k=0}^{n-1} (x_{i+k} - x_{j+k})^2} \quad (1)$$

Before the distance calculation, all subsequences must be normalized. We use $Z$-normalization [10] that makes the value of mean and standard deviation of a time series to be zero and one, respectively. Given a subsequence $T_{i,n} = (t_i, t_{i+1}, \ldots, t_{i+n-1})$ whose mean is $\mu$ and standard deviation is $\sigma$. The normalized time series is $T'_{i,n} = (t'_i, t'_{i+1}, \ldots, t'_{i+n-1})$, where $t'_k = \frac{t_k - \mu}{\sigma}$.

To create a cluster center, we use amplitude averaging approach to average two sequences. Given subsequences $P = (p_1, \ldots, p_i, \ldots, p_n)$ and $Q = (q_1, \ldots, q_i, \ldots, q_n)$, a new subsequence $R = (r_1, \ldots, r_i, \ldots, r_n)$ is produced by $r_i = \frac{\omega_p p_i + \omega_q q_i}{\omega_p + \omega_q}$, where $\omega_p$ and $\omega_q$ are weight of $P$ and $Q$.

### 3.3. Uniform scaling

Many research works show that uniform scaling technique can improve performance in terms of accuracy [7,30]. Specifically, a subsequence $T = (t_1, \ldots, t_i, \ldots, t_m)$ can be shrunk/stretched, by specifying a scaling factor $f \geqslant 1$, to a new time series $T' = (t'_1, \ldots, t'_j, \ldots, t'_n)$, where $t'_j = t_{\lfloor j \cdot m/n \rfloor}$, $\lceil m/f \rceil \leqslant n \leqslant \lfloor m \cdot f \rfloor$. We

extract input time series to subsequences of different lengths. In detail, clustering algorithm takes two parameters, $w$ and $f$, window length and scaling factor, respectively. Subsequences of length from $\lfloor w/f \rfloor$ to $\lfloor w \cdot f \rfloor$ are extracted, then we use uniform scaling to make them the same length of $w$ before clustering them.

### 3.4. Subsequence matching

Subsequence matching algorithm [27] is usually used as a subroutine in many data mining tasks. By giving a query sequence, we can retrieve a subsequence, which is the most similar to the query, from a longer time series. In this work, we use the Euclidean distance as a distance measure to compare the query sequence with all the extracted subsequences.

### 3.5. Subsequence motif discovery

A subsequence motif [23] is the most similar pair of subsequences in a time series data. Many research works have proposed motif discovery algorithms trying to improve performance in terms of speed and accuracy. In this paper, we use the MK algorithm in [19], which is considered the fastest algorithm to find a pair of motif by using the Euclidean distance.

## 4. Selective Subsequence Time Series clustering framework

This section provides details of our approach called Selective Subsequence Time Series (SSTS) clustering framework. Firstly, we begin by stating the problem definition.

### 4.1. Problem definition

Input of our algorithm is a single time series data. The problem is to first determine a number of clusters $n$, and then to group subsequences into proper clusters; some subsequences can be discarded without being assigned to any cluster. The subsequences are extracted using a sliding window approach. The sliding window can be varied in a range specified by a user. For example, we demonstrate by using a 16-character string $S$ = "*ABCDEFGHIJKLMNOP*" as an input. We use a sliding window $w$ of size 3, and a scaling factor $f$ = 1.5; therefore, the length of the subsequences is varied from 2 to 4. The subsequences are extracted into a set $S'$ = {"*AB*","*BC*",..., "*OP*","*ABC*","*BCD*",...,"*NOP*","*ABCD*","*BCDE*",...,"*MNOP*"}. The algorithm should produce a set of clusters $\bar{C} = \{C_1, \ldots, C_i, \ldots, C_n\}$. Each cluster consists of its members and a cluster center: $C_i = \{t_{i_1}, t_{i_2}, \ldots, t_{i_m}, \bar{C}_i\}$, where $t_{i_j}$ is the $j$th member of the $i$th cluster, and the $\bar{C}_i$ is the cluster center of the $i$th cluster.

### 4.2. Clustering method

To form clusters from a set of subsequences, we must iteratively pick one subsequence and assign it to a cluster. However, in the first place, we do not have any predefined cluster yet, and we must make a decision as follows. Intuitively, we can choose two subsequences which are the most similar, to create the first cluster, then the first cluster center is produced. As a result, we can choose other subsequences, which are the most similar to the already created cluster, to be added to the existing cluster; therefore, the cluster center is then updated. Nevertheless, it is better to create a new cluster if there exist two subsequences that are similar to each other more than to the existing cluster center. Moreover, if there are two clusters that can be grouped together, we can decide to merge them to create a new cluster. Thus, we define three operations for producing clusters from a set of subsequences; those are *Create*, *Add* and *Merge* to iteratively select two subsequences to create a new cluster, to assign a subsequence to an existing cluster, and to merge two clusters into a new cluster, respectively.

Our approach iteratively selects an operation, which are *Create*, *Add* and *Merge* to produce a set of clusters. Accordingly, we adopt an idea of data encoding as a heuristic function to choose an optimal operation in each step of clusters construction. We emulate a set of cluster centers as a codebook, where each cluster center is a codeword used to encode the input time series. Some subsequences from the input time series, which are members of a cluster, will be substituted by a small codeword symbol. *Error* of a cluster is determined by a summation of Euclidean distance from the codeword, which is the cluster center, to their member subsequences.

$$\tilde{E}(C_i) = \sum_{j=1}^{m} Dist(t_{i_j}, \bar{C}_i) \tag{2}$$

where $m$ is a number of members in the $i$th cluster.

*Increased error* $\Delta\tilde{E}$ is obtained after a cluster update.

$$\Delta\tilde{E} = \tilde{E}_{after} - \tilde{E}_{before} \tag{3}$$

*Compression ratio* $\bar{R}$ is determined by calculating data reduction of the original subsequence including overhead from codeword construction and codeword symbol substitution.

In detail, *Compression ratio* and *Increased Error* for each operation are described below.

1. *Create*: Create a new cluster $C$ from two subsequences $P$ of length $u$, and $Q$ of length $v$. A new codeword $\bar{C}$ of length $w$ is obtained by merging $P$ and $Q$. The length of input time series $l$ is reduced by $u + v$. The overhead is added by the codeword construction and the substitution of $P$ and $Q$ by two of a codeword symbol "$x$" of size 1.

$$\Delta\tilde{E} = \tilde{E}(C) \tag{4}$$

$$\bar{R} = [(u + v) - (w + 2)]/l \tag{5}$$

2. *Add*: Update an existing cluster $C$ to a new cluster $C'$ by adding a subsequence $P$ of length $u$, and update the codeword $\bar{C}$ to $\bar{C}'$. This operation reduces the length of input time series $l$ by $u$. The overhead is added by substituting $P$ by "$x$" of size 1.

$$\Delta\tilde{E} = \tilde{E}(C') - \tilde{E}(C) \tag{6}$$

$$\bar{R} = (u - 1)/l \tag{7}$$

3. *Merge*: two clusters $C_i$ and $C_j$ are merged into a new cluster $C'$. A codeword of length $w$ is reduced.

$$\Delta\tilde{E} = \tilde{E}(C') - [\tilde{E}(C_i) + \tilde{E}(C_j)] \tag{8}$$

$$\bar{R} = w/l \tag{9}$$

The problem can be considered as a search space consisting of nodes of the three operations, which is illustrated in Fig. 2. Our approach uses greedy method to iteratively select a node that has minimal *Increased Error*. To do this, as shown in Fig. 3, we apply the MK motif discovery algorithm [19] to discover a pair of subsequences, which has minimal Euclidean distance, to be the best node for the *Create* operation. To search for the optimal *Add* node, all codewords are used as queries for the subsequence matching algorithm to locate the best subsequence to be added to an existing cluster. The optimal *Merge* node can be determined by searching all nodes, due to its small number of nodes. Note that the subsequences can be of different lengths, so we use a uniform scaling technique to make them the same length $w$ before applying the motif discovery and the subsequence matching algorithms.

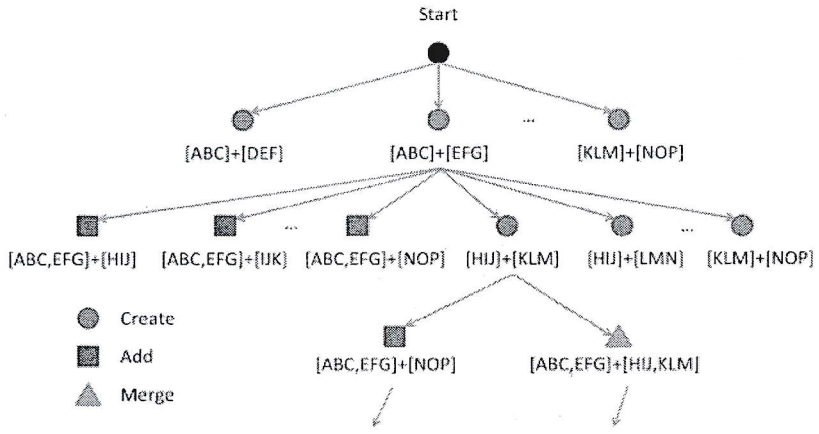



**Fig. 2.** The search space consists of *Create*, *Add* and *Merge* operations.

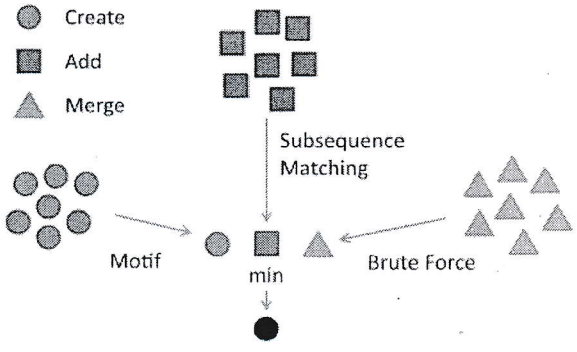



**Fig. 3.** The optimal node can be determined by using motif discovery and subsequence matching algorithms.

As a running example, we use the character string $S$ mentioned in Section 4.1 to demonstrate our algorithm step by step, as shown in Fig. 4. For brevity, we set a window length $w$ to 3 and a scaling factor to 1; so the subsequences can only be of length 3. First, none of the cluster exists. The *Create* operation must be chosen to create the first cluster $C_1$. The motif discovery is applied then we get "*ABC*" and "*EFG*" as its results. A codeword $\bar{C}_1$ is produced by merging "*ABC*" and "*EFG*" together. Second, after creating the first cluster, there are two choices that are *Create* and *Add* operations. We use $\bar{C}_1$ as a query to the subsequence matching algorithm, then we obtain "*NOP*" as the result; after removing "*ABC*" and "*EFG*" from the subsequence set, the motif algorithm returns "*HIJ*" and "*KLM*". In this case, suppose the latter case gives smaller $\Delta\bar{E}$, we choose the *Create* operation to create a new cluster $C_2$ from "*HIJ*" and "*KLM*", then $\bar{C}_2$ is produced. Next, the choices are *create*, *add*, and *merge*. Suppose the smallest $\Delta\bar{E}$ is obtained from the *Merge* operation of $C_1$ and $C_2$, so "*ABC*", "*EFG*", "*HIJ*" and "*KLM*" are merged into the same cluster, and $\bar{C}_1$ and $\bar{C}_2$ are combined. Finally, only $C_1$ remains.

To determine a proper number of clusters, we must choose a state of creating clusters that provides large *compression ratio* while producing less *error*. From the compression–error graph shown in Fig. 5a, it is obvious that there is a *knee point* in the graph where *errors* are dramatically increased. It means applying an operation after that point will lose the clustering accuracies. Thus, we return clusters in that state as a result of the algorithm. To find that point, we determine linear fitting function to the compression–error graph and choose a point that gives minimum residual value to the fitting function as shown in Fig. 5b. Consider a special case that a user want to specify the number of clusters $k$, our algorithm can effortlessly handle it by choosing a latest state that has the number of clusters equal to the one specified by the user.

Table 1 illustrates our main algorithm. The algorithm starts by extracting subsequences from the input sequence by running SubsequenceExtractor function. After that, it enters a loop to iteratively

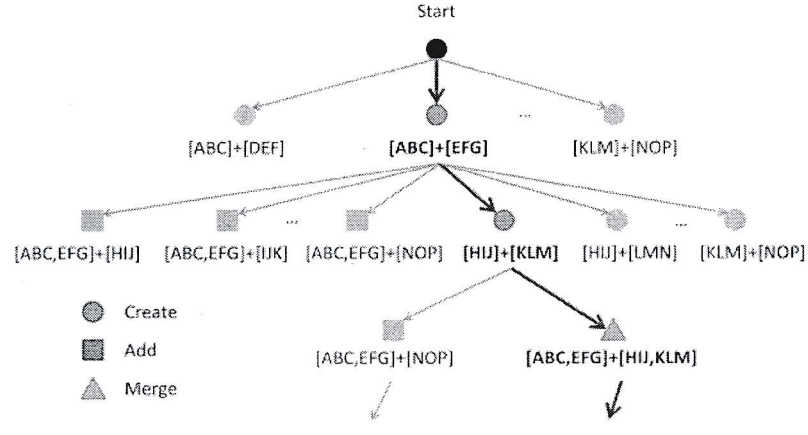



**Fig. 4.** Greedy search example: an optimal node will be chosen in each step.

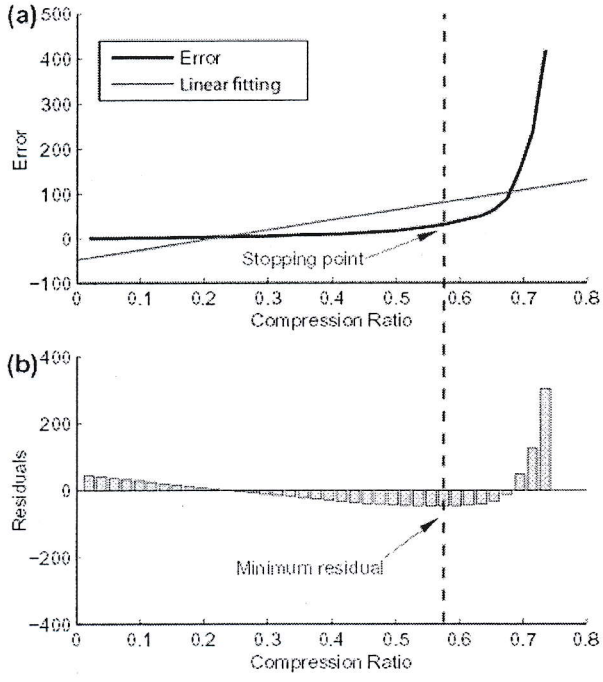



**Fig. 5.** The stopping point can be found at the state that has minimum value of residual. (*a*) A linear estimation line of the *error* line helps find the *knee point* in the *error* line. (*b*) The stopping point is determined at the minimum residuals value to the linear fitting function.

**Table 1**
SSTS clustering.

| Function $[\bar{C}]$ = SSTS($T, w, f$) | |
|---|---|
| 1 | $S$ = SubsequenceExtractor($T, w, f$) |
| 2 | while there is an operation left |
| 3 | $[C[1], S'[1]]$ = CreateCluster($\bar{C}, S$) |
| 4 | $[C[2], S'[2]]$ = UpdateCluster($\bar{C}, S$) |
| 5 | $[C[3], S'[3]]$ = MergeClusters($\bar{C}, S$) |
| 6 | $m$ = ArgMinError($C'$) |
| 7 | $\bar{C} = C'[m]$ |
| 8 | $S = S'[m]$ |
| 9 | $P$.add($\bar{C}$) |
| 10 | return $P$.at(StoppingState($P$)) |

select an operator to create clusters until there is no subsequence left. The *Create*, *Add* and *Merge* operations are applied, then the best one, which gives minimum *error*, is selected in each iteration. Every cluster construction state is kept in a list $P$ for determining the best state later. After breaking the loop, a proper cluster state will be chosen by using StoppingState function.

Details of SubsequenceExtractor function are shown in Table 2. The function extracts subsequences of length varied from $w_{min}$ to $w_{max}$, and makes it the same length by using UniformScaling function. Consequently, the extracted subsequences are normalized by z Normalize function, and they are stored in a list of subsequences $S$.

Table 3 shows CreateCluster function in details. It starts by executing MotifDiscovery to find a motif pair. After that, a cluster is created from the motif pair, and the motif pair and the subsequences that overlap with them are removed from the list of subsequences $S$.

Table 4 explains details of UpdateCluster function. Every cluster center of all created clusters is used as a query sequence for SubsequenceMatching function. The function returns a subsequence

**Table 2**
Subsequence extractor.

| Function $[S]$ = SubsequenceExtractor($T, w, f$) | |
|---|---|
| 1 | $w_{min} = \lfloor w/f \rfloor$ |
| 2 | $w_{max} = \lfloor w \cdot f \rfloor$ |
| 3 | $l$ = Length($T$) |
| 4 | for $i = w_{min}:w_{max}$ |
| 5 | for $j = 1:l$ |
| 6 | $t$ = UniformScaling($s[j:j+i-1]$) |
| 7 | z-Normalize($t$) |
| 8 | $t.start = j$ |
| 9 | $t.end = j + i - 1$ |
| 10 | $S$.add($t$) |
| 11 | return $S$ |

**Table 3**
Create operation.

| Function $[\bar{C}', S']$ CreateCluster($\bar{C}, S$) | |
|---|---|
| 1 | $[l_1, l_2]$ = MotifDiscovery($S$) |
| 2 | $C.\bar{C}$ = Average($S[l_1], S[l_2]$) |
| 3 | $C$.addMember($S[l_1]$) |
| 4 | $C$.addMember($S[l_2]$) |
| 5 | $\bar{C}$.add($C$) |
| 6 | remove $S[l_1]$ and $S[l_2]$ and subsequences that overlap $S[l_1]$ and $S[l_2]$ from $S$ |
| 7 | return $\bar{C}, S$ |

**Table 4**
Add operation.

| Function $[\bar{C}', S']$ = UpdateCluster($\bar{C}, S$) | |
|---|---|
| 1 | for $i = 1:\bar{C}$.numberOfCluster() |
| 2 | $C = \bar{C}[i]$ |
| 3 | $t$ = SubsequenceMatching($S, C.\bar{C}$) |
| 4 | $C.\bar{C}$ = Average($C.\bar{C}, S[t]$) |
| 5 | $C$.addMember($S[t]$) |
| 6 | if $error_{BSF} > C.error()$ |
| 7 | $C' = C$ |
| 8 | $i_{BSF} = i$ |
| 9 | $error_{BSF} = C.error()$ |
| 10 | $t' = t$ |
| 11 | $C[i_{BSF}] = C'$ |
| 12 | remove $S[t']$ and subsequences that overlap $S[t']$ from $S$ |
| 13 | return $\bar{C}, S$ |

**Table 5**
Merge operation.

| Function $[\bar{C}', S']$ = MergeClusters($\bar{C}, S$) | |
|---|---|
| 1 | $n$ = number of clusters in $\bar{C}$ |
| 2 | for $i = 1:n-1$ |
| 3 | for $j = i+1:n$ |
| 4 | $C_1 = \bar{C}[i]$ |
| 5 | $C_1 = \bar{C}[j]$ |
| 6 | $C_1.\bar{C}$ = Average($C_1.\bar{C}, C_2.\bar{C}$) |
| 7 | add all members of $C_2$ to $C_1$ |
| 8 | if $error_{BSF} > C_1.error()$ |
| 9 | $C' = C_1$ |
| 10 | $l_1 = i$ |
| 11 | $l_2 = j$ |
| 12 | $\bar{C}[l_1] = C'$ |
| 13 | $\bar{C}$.remove($l_2$) |
| 14 | return $\bar{C}, S$ |

from $S$ that is the most similar to the query. The subsequence that produces the least *error* is chosen to be added to the cluster that holds cluster center that was used as the query. The cluster center

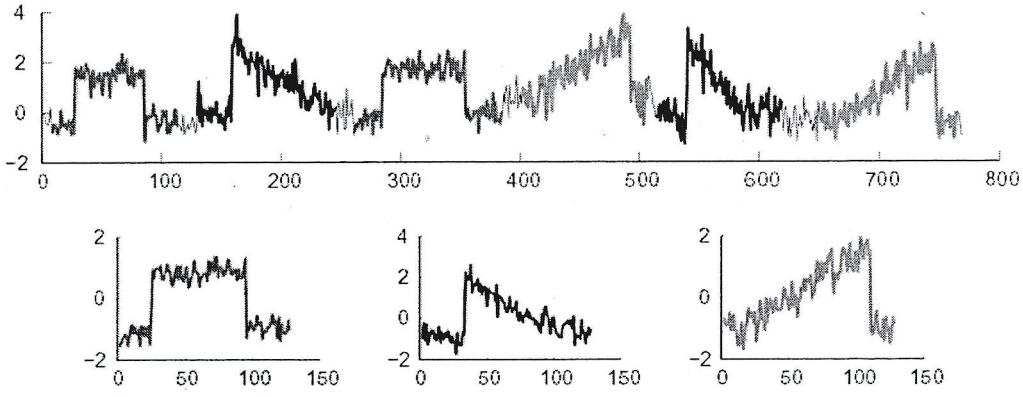


Fig. 6. (*top*) A sequence of CBF dataset. (*bottom*) Cluster centers of each class.

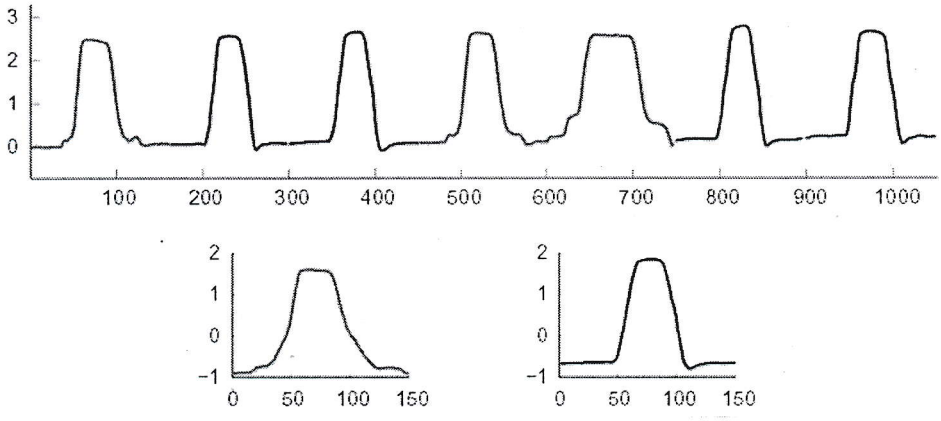


Fig. 7. (*top*) *Gun-Point* data extracted from a video surveillance camera. (*bottom*) Cluster centers of each class.

of that cluster are updated by averaging the old cluster center and the subsequence resulted from the subsequence matching function. After that, the resulted subsequence and its overlapping subsequences are removed from $S$.

The last operation, the *Merge*operation are described as the MergeClusters function shown in Table 5. All combination pairs of the existing clusters are examined, then a pair that gives minimum *error* will be merged.

## 5. Experimental results

This section provides experimental results of our proposed method on various data domains. We separate the experiments into two parts. First, we shows usefulness of our algorithm on real and synthetic datasets from various domains. Second, we demonstrate our search performance by comparing our algorithm with brute-force method on the defined search space.

### 5.1. Usefulness of our method

In this part, we demonstrate that our algorithm can be applied in many types of data domains, i.e., synthetic dataset, data extracted from video surveillance system and images, and real ECG data sequence.

#### 5.1.1. Synthetic data

We experiment on the *Cylinder–Bell–Funnel (CBF)* dataset from the UCR time series archive [14]. It has been shown that most STS clustering algorithms fail to produce meaningful result from this very simple dataset.

In our experiment, we randomly select data from each class, then concatenate them to a single time series, as shown in Fig. 6. The cluster results are illustrated as colored[1] subsequences in Fig. 6. The result shows that the key characteristics of each class are clustered correctly, and the cluster centers can represent the shape of their member subsequences.

#### 5.1.2. Video surveillance problem

In this experiment, we apply our algorithm on the video surveillance domain, which is the gun problem [14]. The time series data is captured from the centroid of each actor's right hand performing two actions: *Gun-Draw* and *Point*. The motion of the two classes of action are very similar and hard to distinguish.

Result of our proposed method, as illustrated in Fig. 7, shows that all subsequences of motions are clustered correctly to their classes. Furthermore, the cluster centers from our method can preserve the important features and shapes in the data.

#### 5.1.3. Time series data extracted from images

This experiment shows the result from clustering data extracted from images, which are created by tracing the local angles from the centroid of an image to its perimeter. We make the input time series by choosing the dataset that has different *complexities* [1]. The

[1] For interpretation of color in Figs. 1–10, the reader is referred to the web version of this article.

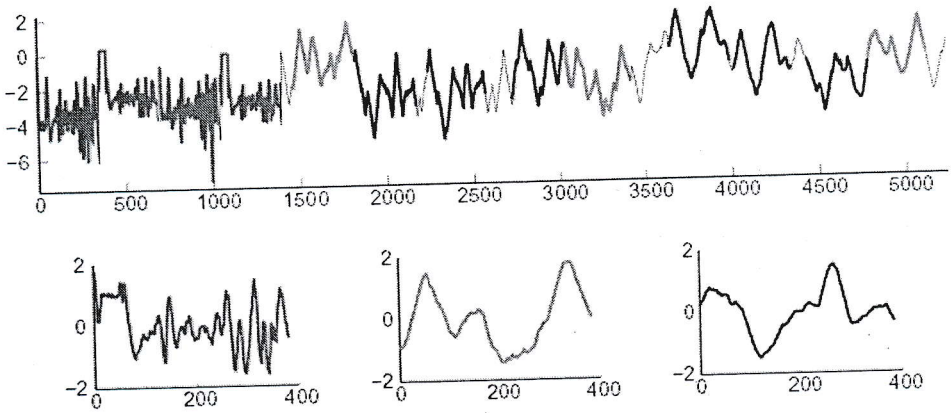


**Fig. 8.** (*top*) A sequence of data extracted from image of faces and leaves. (*bottom*) cluster centers of each class.

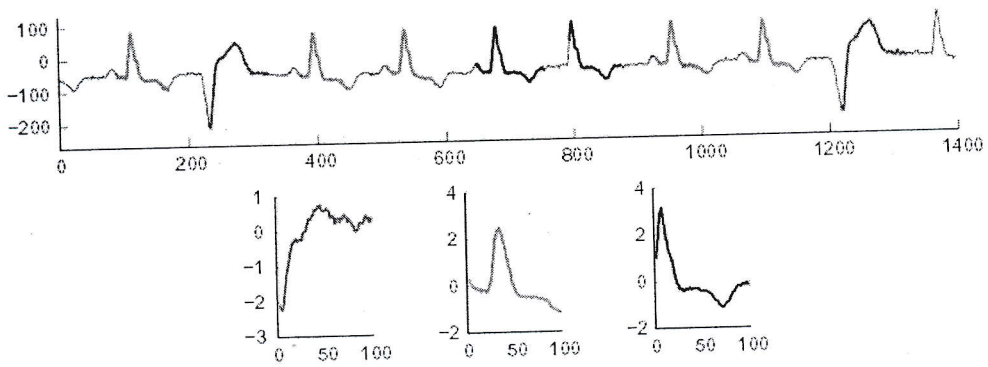


**Fig. 9.** (*top*) ECG sequences with abnormal heartbeats. (*bottom*) Cluster centers from our algorithm.

datasets used here are *Face-all* and *OSU-Leaf* [14], which are extracted from human faces with various expressions on the face, and from different species of leaf images.

Fig. 8 shows that our algorithm can cluster subsequences of the data even when the data has different *complexity* values. The subsequences of face data are grouped in a cluster, which is shown in red, and the leaf subsequences are separated into two subclasses that have the same shape.

### 5.1.4. ECG data

In this experiment, we run our algorithm on a medical dataset, which is an ECG data [9]. Fig. 9 shows that the beats are of different shapes. If we can separate the beats into clusters, the heart diseases will be diagnosed easier. From the result in Fig. 9, three groups of heartbeats are clustered. The normal beats are clustered within the same group as shown in green, the abnormal beats, as shown in red, are clustered into the same group, and the blue cluster contains the beats that have minor anomalies, and are clustered separately.

## 5.2. Comparison with the brute-force method

We will demonstrate our performance on searching through the search space. Fig. 10 illustrates the result of our algorithm comparing with the brute-force method tested on the ECG data used in Section 5.1.4. It roughly contains 6000 possible paths of the input time series of length 1400 data points and a window size of 100. The result of our method is shown in a thick blue line. Our main goal is to maintain *error* while maximizing the *compression ratio*. As shown in Fig. 10, our algorithm can search through the search space closely following the optimal path by examining just 1 out of 6000 possible paths.

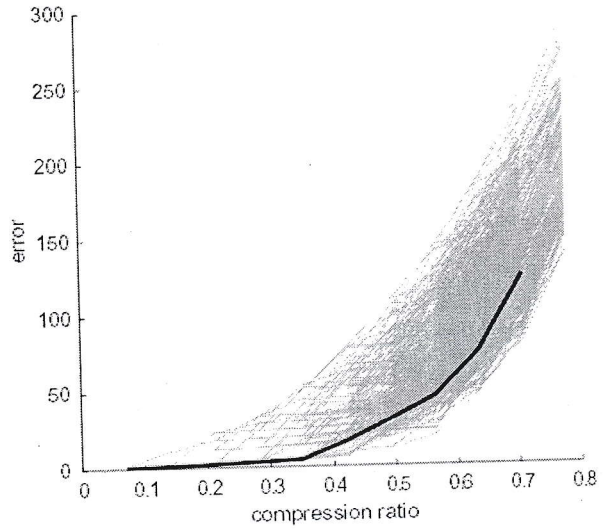



**Fig. 10.** Our algorithm (shown in blue) comparing with the brute-force method.

## 6. Conclusions

In this work, we propose a novel Subsequence Time Series (STS) clustering named Selective Subsequence Time Series (SSTS) clustering. We show that clustering on time series subsequences can be meaningful if some noise or unimportant subsequences are discarded, and different lengths of member subsequences are allowed. We assure the efficiency and usefulness of our algorithm by experimenting in various data domains. Furthermore, our method can perform clustering by requiring only a few parameters where users can easily and flexibly adjust.

## Acknowledgment

This research is partially supported by the Thailand Research Fund (Grant No. MRG5380130), Postdoctoral Fellowship (Ratchadaphiseksomphot Endowment Fund), and the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0319/2551 to S. Rodpongpun).

## References

[1] G.E.A.P.A. Batista, X. Wang, E.J. Keogh, A complexity-invariant distance measure for time series, in: Proceedings of the 2011 SIAM International Conference on Data Mining (SDM'11), Arizona, USA, 2011, pp. 699–710.
[2] J.R. Chen, Making subsequence time series clustering meaningful, in: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05), Texas, USA, 2005, pp. 114–121.
[3] P. Cotofrei, K. Stoffel, Classification rules + time = temporal rules, in: Proceedings of 2002 International Conference on Computational Science, Amsterdam, Netherlands, 2002, pp. 572–581.
[4] G. Das, K. Lin, H. Mannila, G. Renganathan, P. Smyth, Rule discovery from time series, in: 4th International Conference on Knowledge Discovery and Data Mining (KDD'98), New York, USA, 1998, pp. 16–22.
[5] de A.R. Araújo, A class of hybrid morphological perceptrons with application in time series forecasting, Knowledge-Based Systems 24 (4) (2011) 513–529.
[6] A.M. Denton, C.A. Besemann, D.H. Dorr, Pattern-based time-series subsequence clustering using radial distribution functions, Knowledge and Information Systems 18 (2009) 1–27.
[7] A.W.-C. Fu, E. Keogh, L.Y. Lau, C.A. Ratanamahatana, R.C.-W. Wong, Scaling and time warping in time series querying, The VLDB Journal 17 (2008) 899–921.
[8] R. Fujimaki, S. Hirose, T. Nakata, Theoretical analysis of subsequence time-series clustering from a frequency-analysis viewpoint, in: Proceedings of the 2008 SIAM International Conference on Data Mining (SDM'08), Georgia, USA, 2008, pp. 506–517.
[9] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220.
[10] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, second ed., The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, 2006.
[11] T. Idé, Why does subsequence time-series clustering produce sine waves? in: 10th Pacific–Asia Conference on Knowledge Discovery and Data Mining (PAKDD'06), Singapore, 2006, pp. 211–222.
[12] X. Jin, Y. Lu, C. Shi, Distribution discovery: local analysis of temporal rules, in: Proceedings of the 6th Pacific–Asia Conference on Advances in Knowledge Discovery and Data Mining (ICDM'02), London, UK, 2002, pp. 469–480.
[13] E. Keogh, J. Lin, Clustering of time-series subsequences is meaningless: implications for previous and future research, Knowledge and Information Systems 8 (2) (2005) 154–177.
[14] E. Keogh, X. Xi, L. Wei, C.A. Ratanamahatana, The UCR Time Series Classification/Clustering Homepage, 2008. <www.cs.ucr.edu/~eamonn/time_series_data>.
[15] C.-P. Lai, P.-C. Chung, V.S. Tseng, A novel two-level clustering method for time series data analysis, Expert Systems with Applications 37 (2010) 6319–6326.
[16] Y.-S. Lee, L.-I. Tong, Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming, Knowledge-Based Systems 24 (1) (2011) 66–72.
[17] C.-S. Li, P.S. Yu, V. Castelli, Malm: a framework for mining sequence database at multiple abstraction levels, in: Proceedings of the 7th international conference on Information and knowledge management (CIKM'98), New York, USA, 1998, pp. 267–272.
[18] H. Li, C. Guo, Piecewise cloud approximation for time series mining, Knowledge-Based Systems 24 (4) (2011) 492–500.
[19] A. Mueen, E.J. Keogh, Q. Zhu, S. Cash, B. Westover, Exact Discovery of Time Series Motifs, in: SIAM International Conference on Data Mining (SDM'09), Nevada, USA, 2009, pp. 473–484.
[20] V. Niennattrakul, D. Srisai, C.A. Ratanamahatana, Shape-based template matching for time series data, Knowledge-Based Systems 26 (2012) 1–8.
[21] M. Ohsaki, M. Nakase, S. Katagiri, Analysis of subsequence time-series clustering based on moving average, in: Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09), Washington, DC, USA, 2009, pp. 902–907.
[22] S.C.W. Ong, S. Ranganath, Automatic sign language analysis: a survey and the future beyond lexical meaning, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2005) 873–891.
[23] H. Tang, S.S. Liao, Discovering original motifs with different lengths from time series, Knowledge-Based Systems 21 (7) (2008) 666–671.
[24] X. Wang, K. Smith, R. Hyndman, Characteristic-based clustering for time series data, Data Mining and Knowledge Discovery 13 (2006) 335–364.
[25] X. Weng, J. Shen, Classification of multivariate time series using locality preserving projections, Knowledge-Based Systems 21 (7) (2008) 581–587.
[26] X. Weng, J. Shen, Classification of multivariate time series using two-dimensional singular value decomposition, Knowledge-Based Systems 21 (7) (2008) 535–539.
[27] H. Wu, B. Salzberg, G.C. Sharp, S.B. Jiang, H. Shirato, D. Kaeli, Subsequence matching on structured time series data, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD'05), New York, USA, 2005, pp. 682–693.
[28] T. Yairi, Y. Kato, K. Hori, Fault detection by mining association rules from house-keeping data, in: Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Montreal, Canada, 2001, pp. 18–21.
[29] R. Yang, S. Sarkar, B. Loeding, Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 462–477.
[30] D. Yankov, E. Keogh, J. Medina, B. Chiu, V. Zordan, Detecting time series motifs under uniform scaling, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07), New York, USA, 2007, pp. 844–853.

Knowledge-Based Systems 26 (2012) 1–8

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# Shape-based template matching for time series data

Vit Niennattrakul, Dararat Srisai, Chotirat Ann Ratanamahatana *

*Department of Computer Engineering, Chulalongkorn University, 254 Phayathai Road, Pathumwan, Bangkok 10330, Thailand*

ARTICLE INFO

*Article history:*
Received 7 December 2010
Received in revised form 21 April 2011
Accepted 23 April 2011
Available online 29 April 2011

*Keywords:*
Time series
Template matching
Shape averaging
Dynamic time warping
Classification

ABSTRACT

Dynamic time warping (DTW) distance has been proven to be one of the most accurate distance measures for time series classification. However, its calculation complexity is its own major drawback, especially when a massive training database has to be searched. Although many techniques have been proposed to speed up the search including indexing structures and lower bounding functions, for large databases, it is still untenable to embed the algorithm and search through the entire database of a system with limited resources, e.g., tiny sensors, within a given time. Therefore, a template matching is a solution to efficiently reduce storage and computation requirements; in other words, only a few time series sequences have to be retrieved and compared with an incoming query data. In this work, we propose a novel template matching framework with the use of DTW distance, where a shape-based averaging algorithm is utilized to construct meaningful templates. Our proposed framework demonstrates its utilities, where classification time speedup is in orders of magnitude, while maintaining good accuracy to rival methods.



## 1. Introduction

Time series classification [28,27,18,21] is one of the major problems in time series data mining community [3,12,13,26], where its applications contribute in several domains, e.g., speech recognition [6,22], biometrics [14,11,17], multimedia [29,9], etc. Nearest neighbor classifier with dynamic time warping (DTW) distance measure [18,20] has shown in many recent works to outperform dozen other distance measures by using only a few parameters [4]. This leads to active research in DTW distance measure. However, this DTW distance measure has a well-known drawback, i.e., its computational complexity is quadratic. Since the nearest neighbor classifier requires to search through every data sequence in a database, it is definitely impractical to implement on a system with limited resources in either memory storage or computational power, e.g., a tiny sensor or an embedded system.

Recently, many techniques to speed up nearest neighbor search for DTW distance have been widely proposed including lower bounding distance functions [10,30,7,31,23,16] and index structures [7,23,31,16]. A lower bounding distance function, a much faster calculation, is used to estimate DTW distance between two time series sequences with one simple condition: the lower bounding distance must be smaller than or equal to its actual DTW distance. Unlike the lower bounding distance function, an index structure has been proposed to guide the search by accessing only portions of the database instead of searching through the entire database. Although these well-known techniques can reduce some computational time, a large number of sequences are still required to be retrieved for the nearest neighbor calculation. In addition, more storage space is required for storing an index structure, while its main objective is aiming to reduce storage space for a system with limited resources.

Template matching is a solution. Instead of searching for a nearest neighbor from an entire database, only a few templates have to be retrieved, and the class label of the best-matched template is returned as an answer for the issued query. Typically, one template for each class is constructed, so the number of data needed to be stored is merely equal to the number of classes. With a template matching framework, a system with limited resources is now practical; in other words, the system can significantly reduce both storage and computation requirements for classification problems.

Generally, to construct a template, all data sequences of the same class are averaged. Unlike other typical data types, time series data need a shape-based averaging algorithm instead of a typical amplitude averaging approach since correlation among adjacent dimensions of time series exists [15,25]. Additionally, amplitude averaging produces an undesired mean, which leads to an inaccurate classification. Fig. 1(b) shows an undesired averaged result containing two events, whereas both original sequences, *A* and *B*, consist of only one event. In fact, a good template should preserve characteristics of these two data sequences, i.e., only one event should appear, as shown in Fig. 1(c). This characteristic-preserving template can be achieved by a shape-based averaging method.

* Corresponding author. Tel.: +66 8 9499 9400; fax: +66 2 218 6955.
*E-mail addresses:* g49vnn@cp.eng.chula.ac.th (V. Niennattrakul), g51dsr@cp.eng.chula.ac.th (D. Srisai), ann@cp.eng.chula.ac.th (C.A. Ratanamahatana).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.04.015

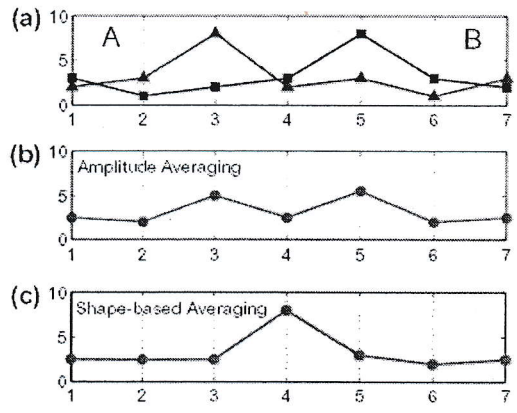



**Fig. 1.** (a) Two time series sequences $A$ and $B$ are averaged using (b) an amplitude averaging and (c) a shape-based averaging.

However, finding a shape-based mean is still controversial because data sequences are averaged in a DTW distance space, not in a Euclidean space. Unfortunately, to the best of our knowledge, no optimal solution has yet been proposed. Over a decade ago, Gupta et al. proposed a heuristic solution called NLAAF [5], but only a handful of work has adapted it to a time series data mining domain [19,24]. Particularly, NLAAF does not produce good averaged results, and consequently this leads to poor classification accuracy (this will be demonstrated and compared with our proposed method in Section 5).

In this work, we propose a novel shape-based template matching framework (STMF) for time series data with deterministic heuristic averaging algorithms. STMF consists of two phases, i.e., a training phase, where templates are constructed, and a test phase, where a query sequence is classified with the constructed templates. To construct a template, a new averaging scheme with two averaging functions, cubic-spline dynamic time warping (CDTW) averaging and iterative cubic-spline dynamic time warping (ICDTW) averaging, is introduced in this paper. With these algorithms, very well-formed templates are stored in the database. In the test phase, templates are retrieved and compared with the query sequence, and a class label of the nearest template will be the answer to the query. It is worth to note that classification with templates typically achieves lower accuracy than classification with an entire database. In experimental evaluation section, we will show that our STMF achieves comparable accuracies, while being able to speed up the classification in orders of magnitude.

The rest of the paper is organized as follows. Sections 2 and 3 provide essential background and related work, respectively. Our framework, STMF (shape-based template matching framework), will be introduced along with two averaging algorithms in Section 4. In Section 5, extensive experimental evaluation will be demonstrated. Finally, we offer conclusions and directions for future work in Section 6.

## 2. Background

This section provides essential background knowledge to understand our proposed methods in this paper.

### 2.1. Dynamic time warping (DTW) distance

DTW distance [1,20,18] is a well-known shape-based similarity measure. It uses a dynamic programming technique to find an optimal warping path between two time series sequences. To calculate the distance, it first creates a distance matrix, where each element in the matrix is a cumulative distance of a minimum of three surrounding neighbors. Suppose we have two time series, a sequence $A = \langle a_1,\ldots,a_i,\ldots,a_n\rangle$ and a sequence $B = \langle b_1,\ldots,b_j,\ldots,b_m\rangle$. First, we create an $n$-by-$m$ matrix, and then each $(i,j)$ element, $\gamma_{i,j}$, of the matrix is defined as:

$$\gamma_{i,j} = |a_i - b_j|^p + \min\{\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i,j-1}\} \quad (1)$$

where $\gamma_{i,j}$ is the summation of $|a_i - b_j|^p$ and a minimum cumulative distance of three elements surrounding the $(i,j)$ element, and $p$ is the dimension of $L_p$-norms. When all elements in the matrix are filled, the DTW distance is determined from the last element $\gamma_{n,m}$ of the matrix. For time series domain, $p = 2$, equipping to a Euclidean distance, is typically used. Since the DTW distance is important background knowledge for this paper, we provide more concrete pseudo code in Table 1 and an illustrative example in Fig. 2.

### 2.2. Dynamic time warping (DTW) averaging

DTW averaging was first introduced by Gupta et al. [5] to find an averaged signal between two time series sequences. Unlike the DTW distance, DTW averaging uses another matrix to store an index of the adjacent element that has a minimum cumulative distance. After elements in the path matrix are filled up, the path is traced back from the last element to the first element. An averaged result is then calculated along the path. Suppose the path $W = \langle w_1,\ldots,w_k,\ldots,w_N\rangle$, where $w_k$ is the $k$th coordinate $(i_k,j_k)$ in the optimal path of sequences $A$ and $B$, where $i_k$ and $j_k$ are indices of data points in sequences $A$ and $B$, respectively. Therefore, a new sequence $C$ is derived from elements $c_k = \frac{a_{i_k}\cdot\omega_A - b_{j_k}\cdot\omega_B}{\omega_A-\omega_B}$, where $\omega_A$ and $\omega_B$ are the weights of the sequences $A$ and $B$, respectively. We also

**Table 1**
Dynamic time warping distance measure.

```
FUNCTION [dist] = DTW-DISTANCE(A,B)
1.  Let n be the length of time series A
2.  Let m be the length of time series B
3.  Initialize D = ARRAY[n][m]
4.  For (i = 1–n)
5.    For (j = 1–m)
6.      If (i = 1 and j ≠ 1)
7.        min = D_{i,j-1}
8.      Else if (i ≠ 1 and j = 1)
9.        min = D_{i-1,j}
10.     Else
11.       min = MIN(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1})
12.     End if
13.     D_{i,j} = min + |a_i − b_j|^p
14.   End for
15. End for
16. Return dist = sqrt(D_{n,m})
```

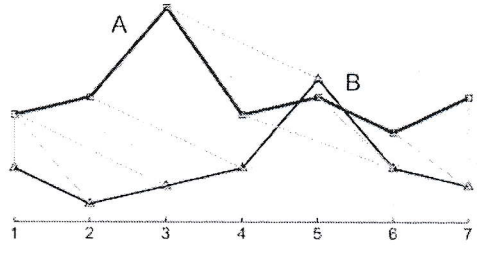



**Fig. 2.** Mapping between two time series sequences $A = \langle 2,3,8,2,3,1,3\rangle$ and $B = \langle 3,1,2,3,8,3,2\rangle$ in DTW distance calculation.

provide a concrete pseudo code and an example of DTW averaging in Table 2 and Fig. 3, respectively.

It is important to note that DTW averaging function is an operation which has only commutative property with no associative property [15]. In the other words, if there are three sequences $A$, $B$, and $C$, a result of averaging $A$ and $B$, then $C$ is not necessarily equal to a result of averaging $B$ and $C$, then $A$. A sequence ordering can largely affect the averaged result. In addition, an averaging sequence will always be longer or equal to the original sequences. If a large dataset is to be averaged, averaging sequences will be very long which will definitely decrease a system's performance. Therefore, in this work, we propose two new DTW averaging functions to resolve this problem and a new averaging scheme to efficiently order averaging sequences.

## 3. Related work

Over a decade ago, Gupta et al. [5] proposed a heuristic shape-averaging scheme called NLAAF, which was first introduced in signal processing community, and later has been utilized in data mining tasks [19,24]. Specifically, NLAAF uses a DTW averaging to produce a mean between a pair of time series sequences. NLAAF consists of two averaging schemes, i.e., $NLAAF_1$ and $NLAAF_2$. $NLAAF_1$ averages sequences in hierarchical manner. Suppose there are eight sequences, i.e., $A_1$ to $A_8$. $A_1$ and $A_2$ are averaged to produce $A_{1,2}$, and $A_3$ and $A_4$ are averaged to produce $A_{3,4}$, and so on. Then, in the next level, $A_{1,2}$ and $A_{3,4}$ are averaged to produce $A_{(1,2),(3,4)}$, and so on. Limitation of $NLAAF_1$ is that it requires the number of sequences to be a power of two. Unlike $NLAAF_1$, $NLAAF_2$ averages sequences in sequential manner. $A_1$ and $A_2$ are first averaged to produce $A_{1,2}$, and then $A_{1,2}$ and $A_3$ are averaged to produce $A_{(1,2),3}$, and so on.

Since $NLAAF_1$ has a limitation that it requires the number of sequences to be a power of two, Gupta et al. recommend to use a combination of both $NLAAF_1$ and $NLAAF_2$. For example, to average 100 sequences, 4 sequences will be discarded, and the rest of the sequences will be separated into three groups of 32 sequences, each of which will be averaged using $NLAAF_1$. Therefore, three averaged sequences produced from $NLAAF_1$ will then be averaged using $NLAAF_2$. Obviously, NLAAF is nondeterministic. Since a DTW averaging function does not have associative property, different orderings of sequences in both $NLAAF_1$ and $NLAAF_2$ will lead to different averaged results. Additionally, an averaged sequence produced by NLAAF will be very long since DTW averaging function will always produce a longer sequence than its original sequences. In this work, we propose two new DTW averaging functions and an averaging scheme which will produce a more accurate averaged result, and when this result is used as a template, it will produce a more accurate classification accuracy.

**Table 2**
Function modified from DTW distance to find an optimal warping path.

| Function $[W]$ = DTW-AVERAGING$(A, B, \omega_A, \omega_B)$ |
|---|
| 1. $W$ = WarpingPath$(A,B)$ |
| 2. Let $N$ be a length of the path $W$ |
| 3. Let $C$ be a time series sequence of length $N$ |
| 4. For $(k = 1$ to $N)$ |
| 5. $[i,j] = w_k$ |
| 6. $c_k \leftarrow \frac{a_i \cdot \omega_A - b_j \cdot \omega_B}{\omega_A - \omega_B}$ |
| 7. Add $c_k$ to $C$ |
| 8. End for |
| 9. Return $C$ |

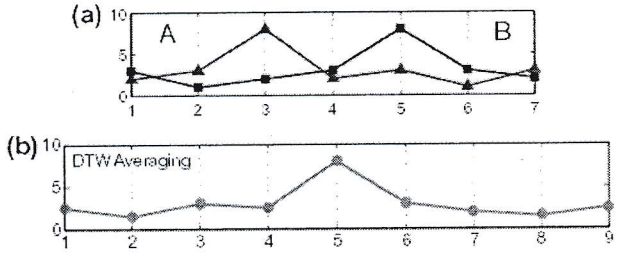



**Fig. 3.** (a) Two sequences $A = (2,3,8,2,3,1,3)$ and $B = (3,1,2,3,8,3,2)$ are averaged by the DTW averaging algorithm to generate (b) an averaged sequence $C = (2.5,1.5,2,3,8,2.5,3,1.5,2.5)$. Note that the length of the result will increase in each averaging.

## 4. Shape-based template matching framework

Shape-based template matching framework (STMF) utilizes shape-based averaging in creating characteristic-preserving templates using the Dynamic time warping (DTW) distance as a similarity measurement. STMF consists of two phases, i.e., a training phase and a test phase. In a training phase, one template for each class is constructed from an entire raw database, and then templates are stored with their class labels. As the best case, only one template for each class is required; however, the number of templates can be more than one. Note that the overall system's performance including a storage requirement and a computational time will improve as the number of templates increases. Table 3 shows a simple idea of a training phase of STMF, where an entire database is an input, and an output is a set of templates.

In a test phase, only a set of templates is retrieved and compared with a query for a closest match, where a set of templates is very small comparing to the original database. Therefore, classification time of template matching will be much faster than the typical one-nearest-neighbor classifier in many orders of magnitude. However, this classification with template matching has a trade-off that its classification accuracy may decrease since some characteristics of data objects in the database could be lost in the averaging process where some details are dominated by a majority of the data. To be more illustrative, Table 4 shows how to classify an incoming query with the stored templates.

To average a set of sequences, we propose a scheme to compute an averaged result since the shape-based averaging does not have a commutative property [17]. Instead of averaging sequences in a random order as done in NLAAF, we propose a heuristic solution to return a good averaged result by averaging a most similar pair of sequences first. After the averaged result is generated, a pair of sequences from the remaining data including the previous averaged result is determined for the next iteration. We keep on going until only one sequence is left. We provide a pseudo code in Table 5.

In this work, we propose two novel averaging functions, i.e., cubic-spline dynamic time warping (CDTW) in Section 4.1 and Iterative cubic-spline dynamic time warping (ICDTW) in Section 4.2.

**Table 3**
Training/template construction phase.

| Function $[\tau]$ = STMF-TrainingPhase$(\sigma)$ |
|---|
| 1. Let $\lambda$ be a set of labels in $\sigma$ |
| 2. For each label $L$ in $\lambda$ |
| 3. $\sigma_L$ = data objects with the same label $L$ |
| 4. $T$ = AveragingScheme$(\sigma_L)$ |
| 5. Add $[T,L]$ in $\tau$ |
| 6. End for |
| 7. Return $\tau$ |

**Table 4**
Test/classification phase.

```
FUNCTION [label_Best] = STMF-TESTPHASE(τ, Q)
1.  dist_Best = INFINITE
2.  label_Best = NULL
3.  For each template T in τ
4.    dist = DTW-AVERAGING (Q,T)
5.    If (dist < dist_Best)
6.      dist_Best = dist
7.      label_Best = T.label
8.    End if
9.  End for
10. Return label_Best
```

Either one of these two averaging functions can be used as the AVERAGINGFUNCTION in Line 4 of Table 5.

### 4.1. Cubic-spline dynamic time warping (CDTW) averaging function

CDTW averaging function produces a more accurate averaged result by considering both position and amplitude of each data point in a new averaged sequence, while the DTW averaging function (Table 2) considers only the amplitude. In other words, the DTW averaging function equally treats every new data point in a new sequence, while the CDTW averaging function additionally determines where a new data point should be placed. Specifically, a position and an amplitude of a data point in the sequence can be observed as an $x$- and $y$-coordinate in time series. To be more illustrative, Fig. 4 shows a comparison between two new sequences generated by CDTW and DTW. From the figure, the sequence generated from the CDTW algorithm is more useful since it preserves both position and amplitude from the warping path.

Suppose the path $W = (w_1,\ldots,w_k,\ldots,w_N)$, where $w_k$ is the $k$th coordinate $(i_k,j_k)$ in the optimal path of sequences $A$ and $B$, where $i_k$ and $j_k$ are indices of data points in sequences $A$ and $B$, respectively. Therefore, a position $c_{k_x}$ of a data point in a new sequence $C$ is determined by $c_{k_x} = \frac{i_k \cdot \omega_A - j_k \cdot \omega_B}{\omega_A - \omega_B}$, and an amplitude $c_{k_y}$ of a data point in a new sequence $C$ is determined by $c_{k_y} = \frac{a_{i_k} \cdot \omega_A - b_{j_k} \cdot \omega_B}{\omega_A - \omega_B}$, where $\omega_A$ and $\omega_B$ are the weights of the sequences $A$ and $B$, respectively.

However, the length of the sequence $C$ is always equal to or longer than the two original sequences; therefore, re-sampling is required. In this work, CDTW uses a cubic-spline interpolation [2] since it requires no parameter and outperforms other interpolation techniques in re-sampling of natural sequences. Additionally, CDTW re-samples positions of the averaged result to integer values. As illustrated in Fig. 5, the sequence $C$ of 9 data points is re-sampled to the sequence $C'$ of 7 data points. We provide a concrete pseudo code of CDTW in Table 6.

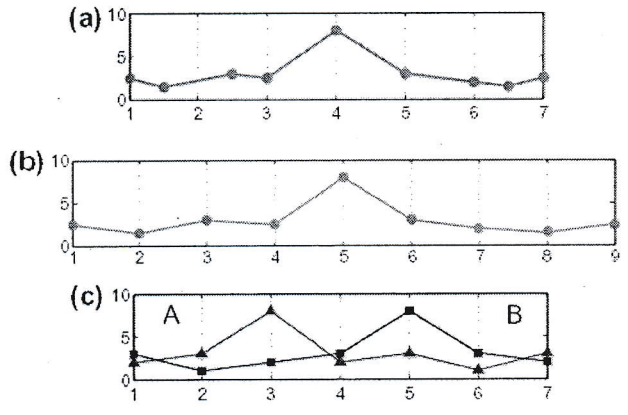



**Fig. 4.** Comparision between averaged results generated from (a) CDTW and (b) DTW averaging functions, where (c) two inputs are $A = (2,3,8,2,3,1,3)$ and $B = (3,1,2,3,8,3,2)$.

### 4.2. Iterative cubic-spline dynamic time warping (ICDTW) averaging function

Although CDTW produces a good averaged result since it considers both a position and an amplitude, another essential but not necessary condition for averaging is that the averaged result should be in the middle of two original sequences. In other words, DTW distances between the sequences and the result should be equal. Therefore, we propose an iterative approach for the CDTW averaging function called iterative cubic-spline dynamic time warping (ICDTW) averaging function that can truly represent characteristics of a set of subsequences. A good real averaged result can be determined from the result that gives minimum summation distances between the result itself and every data sequence. Specifically, if two data sequences are considered, the averaged result is the sequence which not only has minimum summation distance but also gives an equal distance between itself to these two data sequences.

We would like to emphasize that the distances between the generated result from the CDTW function and the two original time series are *not* always equal; therefore, the averaged result needs to be slightly adjusted. Obviously, since all elements in the sequence are real numbers, it is very difficult to obtain the sequence that satisfies this condition. We therefore propose a heuristic and deterministic solution, i.e., ICDTW averaging function mentioned above. To average two time series sequences $A$ and $B$, the ICDTW function will find new weights $\beta_A$ and $\beta_B$ that make the averaged result $C$ be the center between the sequences $A$ and $B$. Obviously, finding both weights $\beta_A$ and $\beta_B$ is not very practical since the weights $\beta_A$ and $\beta_B$ are real numbers. We instead heuristically use a binary search to find only the weight $\beta_A$, when the weight $\beta_B$ is

**Table 5**
STMF averaging scheme.

```
FUNCTION [C] = STMF-AVERAGINGSCHEME(s)
1. Initialize a weight ω = 1 for each sequence S in s
2. While (SIZE(s) > 1)
3.   [A,B] = Most similar sequences in s
4.   C = AVERAGINGFUNCTION(A,B,ω_A,ω_B)
5.   Remove A and B from s
6.   ω_C = ω_A + ω_B
7.   Add C to s
8. End while
9. Return C
```

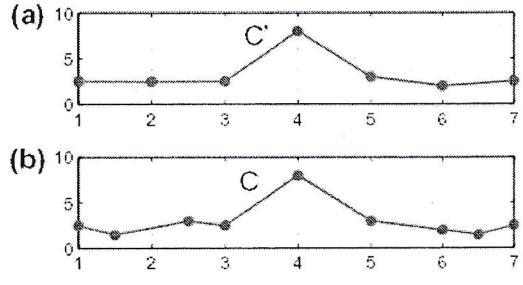



**Fig. 5.** (a) A new sequence $C'$ which is re-sampled from (b) 9 data points in the sequence $C$ in Fig. 4(a).

**Table 6**
Cubic-spline dynamic time warping averaging function.

```
Function [C] = CDTW-Averaging(A, B, ω_A, ω_B)
1.  W = WarpingPath(A,B)
2.  Let N be the length of the path W
3.  Let N' be the length of time series A and B
4.  Let C be a time series sequence of size N
5.  Let C' be a time series sequence of size N'
6.  For (k = 1 to N)
7.    [i,j]=w_k
8.    x = (i·ω_A − j·ω_B)/(ω_A + ω_B)
9.    y = (a_i·ω_A + b_j·ω_B)/(ω_A + ω_B)
10.   Add [x,y] to C
11. End for
12. C' = CubicSpline(C)
13. Return C'
```

**Table 7**
Iterative cubic-spline dynamic time warping averaging function.

```
Function [C] = ICDTW-Averaging(A, B, ω_A, ω_B)
1.  Initialize weights β_A1 = 10^-5, β_A2 = 10^5, and β_B = 1
2.  Initialize weight β_A3 = (β_A1 − β_A2)/2
3.  C = CDTW-Averaging(A, B, β_A3, β_B)
4.  d_CA = DTWDistance(C,A) · ω_A
5.  d_CB = DTWDistance(C,B) · ω_B
6.  β_A3 = d_CA < d_CB ? β_A1 : β_A2
7.  While (|d_CA − d_CB| > 0)
8.    β_A3 = (β_A1 − β_A2)/2
9.    C = CDTW-Averaging(A, B, β_A3, β_B)
10.   d_CA = DTWDistance(C,A) · ω_A
11.   d_CB = DTWDistance(C,B) · ω_B
12.   If (d_CA < d_CB)
13.     β_A1 = β_A3
14.   Else
15.     β_A2 = β_A3
16.   End if
17. End while
18. Return C
```

**Table 8**
Details of datasets.

| Dataset | Number of classes | Length | Size of training set | Size of test set |
|---|---|---|---|---|
| Synthetic Control | 6 | 60 | 300 | 300 |
| CBF | 3 | 128 | 30 | 900 |
| Face All | 14 | 131 | 560 | 1690 |
| OSU Leaf | 6 | 427 | 200 | 242 |
| 50 Words | 50 | 270 | 450 | 455 |
| Trace | 4 | 275 | 100 | 100 |
| Two Patterns | 4 | 128 | 1000 | 4000 |
| Wafer | 2 | 152 | 1000 | 6174 |
| Face Four | 4 | 350 | 24 | 88 |
| Lightning-2 | 2 | 637 | 60 | 61 |
| Lightning-7 | 7 | 319 | 70 | 73 |
| ECG | 2 | 96 | 100 | 100 |
| Adiac | 37 | 176 | 390 | 391 |
| Yoga | 2 | 426 | 300 | 3000 |
| Fish | 7 | 463 | 175 | 175 |

fixed. Specifically, for each iteration, a new weight $\beta_A$ is considered whether or not the generated averaged result $C$ has an equal DTW distance to the sequences $A$ and $B$. If the distance is equal, ICDTW terminates. In other words, we only need to determine the weight $\beta_A$ and hold the weight $\beta_B$ constant because two sets of weights are equivalent. For example, for $\{\beta_A, \beta_B\} = \{4,5\}$, it can be reduced to $\{0.8,1\}$ when the weight $\beta_B$ is fixed to 1; therefore, searching for $\beta_B$ is enough to find any pair of weights $\{\beta_A, \beta_B\}$. Pseudo code of the ICDTW averaging function is provided in Table 7.

Note that both CDTW and ICDTW averaging functions can be used in STMF under users' preference. For CDTW, the averaged result preserves the shapes of both original sequences by considering both the position and the amplitude of the warping alignment, while ICDTW averaging returns more accurate characteristics of the averaged result by calibrating the averaged sequence to have the same distance between itself and the two original sequences. Performances of CDTW and ICDTW will be demonstrated in the next section.

## 5. Experimental evaluation

Three following experiments will demonstrate the superiority of our proposed method over the current existing approaches. The first experiment shows accuracies of our shape-based averaging method, i.e., a new averaging scheme with two proposed CDTW and ICDTW algorithms, comparing with NLAAF. In the second experiment, we show that our STMF with both CDTW and ICDTW outperforms NLAAF and traditional nearest neighbor classification in terms of accuracy, storage requirement, and time usage for classification problems. Our extension of STMF is also evaluated to show that STMF can support multiple templates within each class, and to show that traditional nearest neighbor classification is simply a special case. All codes are implemented in C++ and run on an Intel Core i7 desktop computer. We evaluate our proposed method with 15 datasets from the publically available UCR classification/clustering archive [8]. Table 8 shows the number of classes, length of each time series data, and size of training/test sets.

### 5.1. First experiment

In this experiment, we will demonstrate that our proposed averaging methods, which utilize a new averaging scheme with CDTW and ICDTW, well represent sequences in the datasets. For each dataset, its training data and test data are all combined, and then all sequences are averaged. In real-word applications, sequences should be separated by its own class to achieve maximum utilities. The averaged results are evaluated using SumDist function defined as a summation of the distance between the averaged result and all the original sequences in the dataset. If a value from SumDist is small, it means that this method generates a good averaged result. SumDist function is provided as follows.

$$\mathrm{SumDist}\left(\bar{D}, \square\right) = \sum_{i=1}^{|\square|} \mathrm{DTWDistance}\left(\bar{D}, D_i\right) \tag{2}$$

where $\square$ is a dataset, $\bar{D}$ is the averaged result, and $D_i$ is a data sequence in the dataset $\square$.

Table 9 shows the comparison between SumDist of NLAAF and our proposed methods, CDTW and ICDTW. The lowest SumDist for each dataset is emphasized in bold. Both proposed methods achieve lower SumDist values since all sequences are averaged using a new averaging schemes, while NLAAF averages sequences in random manner. In addition, no re-sampling method is adopted in NLAAF to scale the averaged sequence down to the same length. This means that at each step, NLAAF will produce a longer averaged sequence. It is apparent from the experiment results that CDTW and ICDTW generate more accurate averages.

We can see from the results that ICDTW usually yields smaller SumDist than CDTW in almost all of the datasets, and smaller SumDist than NLAAF in every dataset. However, in some cases such as 50 Words and Yoga datasets, where sequences within the class are very diverse, ICDTW may give slightly larger SumDist than CDTW (but still performs much better than NLAAF); in such cases, CDTW performs better because ICDTW tries its best to equalize the

**Table 9**
SumDist of averaging methods.

| Dataset | NLAAF | STMF | |
|---|---|---|---|
| | | CDTW | ICDTW |
| Synthetic Control | 8962 | 3545 | **3538** |
| CBF | 17,827 | 5119 | **4856** |
| Face All | 43,314 | 12,820 | **12,597** |
| OSU Leaf | 12,179 | 3313 | **3276** |
| 50 Words | 28,810 | **5034** | 5215 |
| Trace | 9417 | 2054 | **2046** |
| Two Patterns | 105,517 | 34,376 | **33,268** |
| Wafer | 693,016 | **54,115** | **54,115** |
| Face Four | 1735 | 757 | **729** |
| Lightning-2 | 2877 | 1223 | **1196** |
| Lightning-7 | 2961 | 1126 | **1113** |
| ECG | 1504 | 545 | **544** |
| Adiac | 1884 | 512 | **495** |
| Yoga | 163,207 | **16,826** | 17,193 |
| Fish | 944 | 371 | **364** |

**Table 10**
Classification accuracy and time usage (seconds) of STMF compared with NLAAF.

| Dataset | NLAAF | STMF | |
|---|---|---|---|
| | | CDTW | ICDTW |
| Synthetic Control | 0.80 (1) | **0.97**(0) | 0.92 (0) |
| CBF | 0.94 (4) | **0.96**(1) | 0.95 (1) |
| Face All | 0.57 (46) | **0.83**(11) | 0.81 (12) |
| OSU Leaf | 0.35 (67) | 0.41(13) | **0.42**(7) |
| 50 Words | 0.42 (75) | **0.60**(44) | 0.58 (43) |
| Trace | 0.92 (6) | 0.98 (1) | **1.00**(2) |
| Two Patterns | 0.92 (122) | **0.97**(7) | 0.95 (13) |
| Wafer | 0.10 (514) | **0.64**(14) | 0.63 (8) |
| Face Four | 0.56 (2) | **0.83**(1) | 0.81 (1) |
| Lightning-2 | 0.56 (16) | **0.56**(1) | 0.54 (2) |
| Lightning-7 | 0.59 (7) | 0.66 (2) | **0.70** (1) |
| ECG | 0.65 (1) | 0.70 (0) | **0.71**(0) |
| Adiac | 0.48 (17) | **0.49** (11) | 0.47 (12) |
| Yoga | **0.48** (684) | **0.48** (27) | **0.48**(27) |
| Fish | 0.57 (25) | 0.58 (7) | **0.59** (8) |

**Table 11**
Storage requirement in KB for storing templates.

| Dataset | NLAAF | STMF | Original dataset |
|---|---|---|---|
| Synthetic Control | 12.40 | **1.44** | 72.00 |
| CBF | 43.91 | **1.54** | 15.36 |
| Face All | 68.35 | **7.36** | 293.44 |
| OSU Leaf | 172.15 | **10.25** | 341.60 |
| 50 Words | 303.87 | **54.00** | 486.00 |
| Trace | 68.87 | **4.40** | 110.00 |
| Two Patterns | 117.81 | **2.05** | 512.00 |
| Wafer | 476.97 | **1.22** | 608.00 |
| Face Four | 24.21 | **5.60** | 33.60 |
| Lightning-2 | 120.37 | **5.10** | 152.88 |
| Lightning-7 | 64.37 | **8.93** | 89.32 |
| ECG | 13.30 | **0.77** | 38.40 |
| Adiac | 44.14 | **26.05** | 274.56 |
| Yoga | 477.70 | **3.41** | 511.20 |
| Fish | 82.44 | **12.96** | 324.10 |

distances, but this in turn could increase the summarized distance (SumDist). ICDTW guarantees that distances between the averaged result and the two original sequences are identical, though not guarantee to be minimal. In other words, there is a tradeoff between having an exact average with equal distances and having an approximate average with minimal distances.

### 5.2. Second experiment

This experiment demonstrates the utility of our proposed shape-based template matching framework (STMF) over a

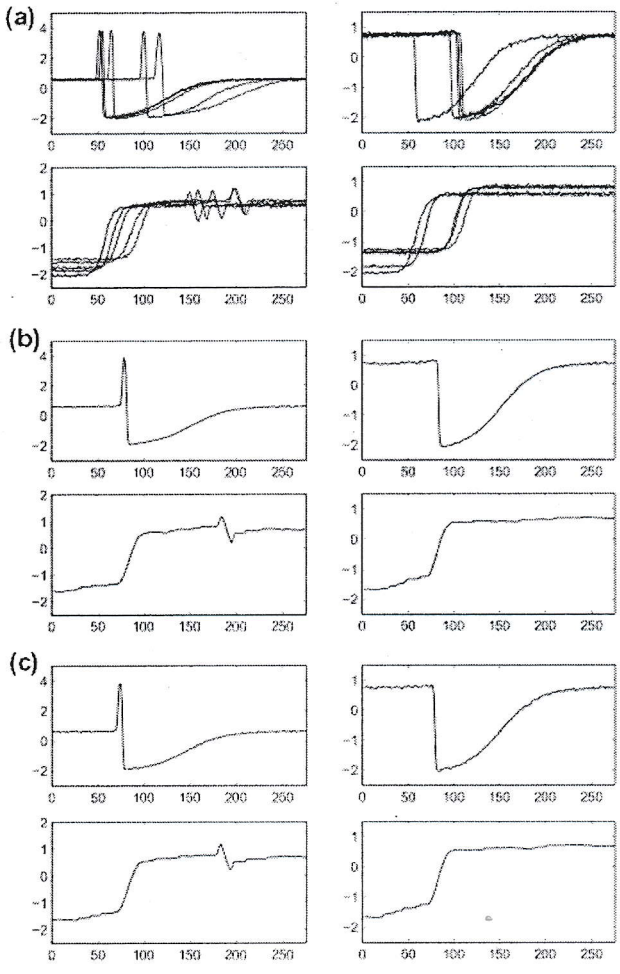



**Fig. 6.** (a) Example sequences of four classes of the Trace dataset and templates of each class generated from STMF with (b) CDTW and (c) ICDTW.

template matching with NLAAF in terms of three metrics, i.e., classification accuracy, storage requirements, and time usage. Classification accuracy is determined by a classification on a set of templates, and storage requirement is measured from the an amount of memory needed to store a set of templates. Time needed to classify the test dataset is also reported. Comparison of accuracy, storage, and time usage are shown in Tables 10 and 11, where our method outperforms NLAAF in every dataset. Templates of Trace dataset generated with CDTW and ICDTW are illustrated in Fig. 6. The best result is emphasized in bold for each dataset.

The main reason is the fact that data sequences in most datasets distribute unequally. Although ICDTW generates a more accurate averaged result, this averaged result may not be the best sequence to represent the whole unequal distribution of the dataset; each class may best be represented by two or more templates since multiple sub-classes may exist. Therefore, by using only one template to represent every sequence of the same class, ICDTW could return averaged characteristics of those multiple sub-classes which tends to increase classification error.

### 5.3. Third experiment

Our proposed averaging scheme allows STMF to produce multiple templates for each class in classification. In this experiment, we show the classification accuracy of STMF when the number of

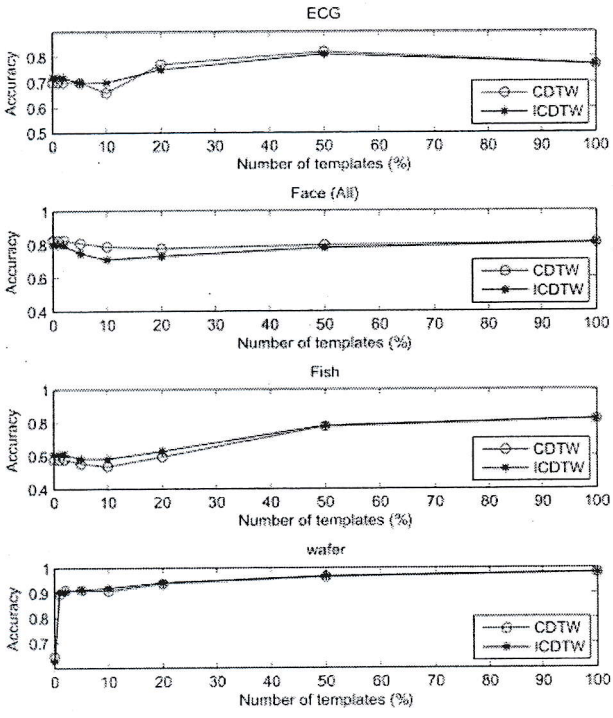



**Fig. 7.** Classification accuracy of ECG, Face All, Fish, and Wafer datasets when the number of templates are varied.

templates are varied. The number of templates used in real-world applications depends on resources of the system, i.e., memory storage and computational power. Specifically, the system with small memory storage and limited computational power is suggested to use smallest number of templates possible. In Fig. 7, we shows classification accuracy when the size of templates are varied in four datasets, i.e., ECG, Face All, Fish, and Wafer. The classic nearest neighbor classification is considered a special case when the number of templates is set to the number of instances in the class. In other words, every training sequence is used in template matching and no sequence is discarded.

## 6. Conclusion and future work

In this work, we propose a novel shape-based template matching framework which utilizes a new averaging scheme and averaging functions to generate an accurate set of templates. This set of templates is used as a dataset for query classification. Compared with the existing method, our proposed method outperforms in every case in terms of classification accuracy, time usage, and storage requirement. In addition, our method can also be extended to generate two or more templates for each class when users have more resources in real-world applications. This research can be applied to diverse domains where time series classification is needed and it will be more useful when the system has limited resource in terms of storage and computational power.

## Acknowledgement

This research was supported in part by the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0141/2549 to V. Niennattrakul and C.A. Ratanamahatana), in part by the Chulalongkorn University Graduate Scholarship to Commemorate the 72nd Anniversary of His Majesty King Bhumibol Adulyadej, and in part by the 90th Anniversary of Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund).

## References

[1] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, 1994, pp. 359–370.
[2] R.L. Burden, J.D. Faires, A.C. Reynolds, Numerical Analysis, 1997.
[3] R. de A. Araújo, A class of hybrid morphological perceptrons with application in time series forecasting, Knowledge-based Systems 24 (4) (2011) 513–529.
[4] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E.J. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proceedings of the VLDB Endowment 1 (2) (2008) 1542–1552.
[5] L. Gupta, D. Molfese, R. Tammana, P. Simos, Nonlinear alignment and averaging for estimating the evoked potential, IEEE Transactions on Biomedical Engineering 43 (4) (1996) 348–356.
[6] F. Itakura, Minimum prediction residual principle applied to speech recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 23 (1) (1975) 67–72.
[7] E. Keogh, C.A. Ratanamahatana, Exact indexing of dynamic time warping, Knowledge and Information Systems 7 (3) (2005) 358–386.
[8] E. Keogh, X. Xi, L. Wei, C.A. Ratanamahatana, UCR time series classification/clustering page. <http://www.cs.ucr.edu/~eamonn/time_series_data>, 2010.
[9] E.J. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, P. Protopapas, Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures, The VLDB Journal 18 (3) (2009) 611–630.
[10] S.-W. Kim, S. Park, W.W. Chu, 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'01), Heidelberg, Germany, 2001, pp. 607–614.
[11] Z. Kovacs-Vajna, A fingerprint verification system based on triangular matching and dynamic time warping, IEEE Transactions of Pattern Analysis and Machine Intelligence 22 (11) (2000) 1266–1276.
[12] Y.-S. Lee, L.-I. Tong, Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming, Knowledge-based Systems 24 (1) (2011) 66–72.
[13] H. Li, C. Guo, Piecewise cloud approximation for time series mining, Knowledge-based Systems 24 (4) (2011) 492–500.
[14] M.E. Munich, P. Perona, Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification, in: Proceedings of the 7th International Conference on Computer Vision (ICCV'99), Corfu, Greece, 1999, p. 108.
[15] V. Niennattrakul, C.A. Ratanamahatana, Inaccuracies of shape averaging method using dynamic time warping for time series data, in: Proceedings of the 7th International Conference on Computational Science (ICCS'07), Beijing, China, 2007, pp. 513–520.
[16] V. Niennattrakul, P. Ruengronghirunya, C. Ratanamahatana, Exact indexing for massive time series databases under time warping distance, Data Mining and Knowledge Discovery 21 (3) (2010) 509–541.
[17] V. Niennattrakul, D. Wanichsan, C.A. Ratanamahatana, Hand geometry verification using time series representation, in: Proceedings of 11th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'07), Vietri sul Mare, Italy, 2007, pp. 824–831.
[18] C.A. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints, in: Proceedings of the Fourth SIAM International Conference on Data Mining (SDM'04), Lake Buena Vista, FL, 2004, pp. 11–22.
[19] C.A. Ratanamahatana, E.J., Keogh, Multimedia retrieval using time series representation and relevance feedback, in: Proceedings of 8th International Conference on Asian Digital Libraries (ICADL2005), Bangkok, Thailand, 2005, pp. 400–405.
[20] C.A. Ratanamahatana, E.J. Keogh, Three myths about dynamic time warping data mining, in: Proceedings of the Fifth SIAM International Data Mining Conference (SDM'05), Newport Beach, CL, 2005, pp. 506–510.
[21] J.J. Rodriguez, C.J. Alonso, J.A. Maestro, Support vector machines of interval-based features for time series classification, Knowledge-Based Systems 18 (4–5) (2005) 171–178.
[22] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 43–49.
[23] Y. Sakurai, M. Yoshikawa, C. Faloutsos, FTW: Fast similarity search under the time warping distance, in: Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Baltimore, ML, 2005, pp. 326–337.
[24] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space, Intelligent Data Analysis 11 (5) (2007) 561–580.
[25] D. Srisai, C.A. Ratanamahatana, Time series shape averaging using time-warping alignment with re-sampling, in: Proceedings of the 6th International Joint Conference on Computer Science and Software Engineering (JCSSE'09), Phuket, Thailand, 2009.
[26] H. Tang, S.S. Liao, Discovering original motifs with different lengths from time series, Knowledge-based Systems 21 (7) (2008) 666–671.
[27] X. Weng, J. Shen, Classification of multivariate time series using locality preserving projections, Knowledge-Based Systems 21 (7) (2008) 581–587.

[28] X. Weng, J. Shen, Classification of multivariate time series using two-dimensional singular value decomposition, Knowledge-Based Systems 21 (7) (2008) 535–539.
[29] D. Yankov, E. Keogh, L. Wei, X. Xi, W. Hodges, Fast best-match shape searching in rotation-invariant metric spaces, IEEE Transactions on Multimedia 10 (2) (2008) 230–239.
[30] B.-K. Yi, H.V. Jagadish, C. Faloutsos, Efficient retrieval of similar time sequences under time warping, in: Proceedings of the Fourteenth International Conference on Data Engineering (ICDE'98), Orlando, FL, 1998, pp. 201–208.
[31] Y. Zhu, D. Shasha, Warping indexes with envelope transforms for query by humming, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03), San Diego, CA, 2003, pp. 181–192.

# Efficient Subsequence Search on Streaming Data Based on Time Warping Distance

Sura Rodpongpun[1],
Vit Niennattrakul [2], and Chotirat Ann Ratanamahatana[3], Non-members

## ABSTRACT

Many algorithms have been proposed to deal with subsequence similarity search problem in time series data stream. Dynamic Time Warping (DTW), which has been accepted as the best distance measure in time series similarity search, has been used in many research works. SPRING and its variance were proposed to solve such problem by mitigating the complexity of DTW. Unfortunately, these algorithms produce meaningless result since no normalization is taken into account before the distance calculation. Recently, GPUs and FPGAs were used in similarity search supporting subsequence normalization to reduce the computation complexity, but it is still far from practical use. In this work, we propose a novel Meaningful Subsequence Matching (MSM) algorithm which produces meaningful result in subsequence matching by considering global constraint, uniform scaling, and normalization. Our method significantly outperforms the existing algorithms in terms of both computational cost and accuracy.

Keywords: Subsequence Matching, Dynamic Time Warping Distance, Data Stream, Normalization

## 1. INTRODUCTION

Due to the age of data explosion, analysis of data stream in real time is crucial in many data mining tasks including classification, clustering, anomaly detection, and pattern discovery. Commonly, these tasks require a subsequence matching algorithm as an important subroutine. Recently, SPRING [10], a breakthrough subsequence matching algorithm for data stream under Dynamic Time Warping (DTW) distance [9] has been proposed. SPRING can report an optimal subsequence in linear time. More specifically, it incrementally updates DTW distance, for each new streaming data point, only in time complexity of the query sequence's length. After the proposal of SPRING, many authors [1][7][13] have introduced fast algorithms to improve performance of subsequence matching. In this work, we claim that all of those past research works [1][10][7][13] are meaningless because the query sequence and candidate sequences from the data stream were not normalized. Normalization [3] is essential to achieve accurate and meaningful distance calculation, as it normalizes the data to have similar offset and distribution, regardless of the distance measure used, especially for DTW distance measure. Unfortunately, as we have mentioned above, current subsequence matching algorithms concern mostly about speed enhancement, but neither on accuracy nor meaningfulness. Fig.1 illustrates subsequence searching in ECG data [3]. Many subsequences with similar shape to the query are missed by the search without normalization.

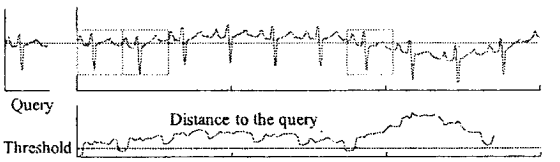



*Fig.1: Subsequence searching without normalization in ECG data. Many subsequences with similar shape to the query are left undetected.*

However, there is an effort to resolve this problem by trying other approaches; the latest one devises some hardware [11] to accelerate the computation time. The authors propose two techniques, i.e. GPUs and FPGAs, to speed up subsequence matching using DTW with normalization. They have shown that GPUs and FPGAs can help speed up the search significantly. However, it is not practical in real world problems; implementation is hardware dependent, and some systems are not flexibly adjusted to the problem.

We introduce a novel subsequence matching algorithm called MSM (Meaningful Subsequence Matching) for data stream under DTW distance. MSM consists of two new ideas. First, we introduce a multi-resolution lower bound, LB_GUN (Lower-Bounding distance function under Global constraint, Uniform scaling, and Normalization) combining with the well-known LB_Keogh [5] lower-bounding function. LB_GUN is a new lower-bounding distance function extended from LB_Keogh. Second, SSM (Scaling

Manuscript received on January 28, 2010 ; revised on January 28, 2011.

[1,2,3] The authors are with Department of Computer Engineering, Chulalongkorn University Phayathai Rd., Pathumwan, Bangkok 10330 Thailand, E-mail: g53srd@cp.eng.chula.ac.th, g49vnn@cp.eng.chula.ac.th and ann@cp.eng.chula.ac.th

Subsequence Matrix) is used for lower-bounding distance estimation of LB_GUN by incrementally estimating value of normalized data point while guaranteeing no false dismissals. The distances for every scaled query sequence are stored in SSM, and then MSM algorithm monitors SSM to report the optimal range query or the optimal top-$k$ query when a new streaming data point is received. From these two ideas, MSM can monitor data stream nearly in linear time, and it also achieves much higher accuracy than existing algorithms as we expected.The remainder of this paper is organized as follows. We provide some essential background in Section 2, and state the problem definitions in Section 3. MSM, our proposed method, is described in Section 4. Experimental results are reported in Section 5, and our work is concluded in Section 6.

## 2. BACKGROUND

In this section, we provide essential background knowledge of Dynamic Time Warping distance measure, global constraint, lower-bounding function for DTW distance, uniform scaling, and normalization.

### 2.1 Dynamic Time Warping Distance Measure

Dynamic Time Warping (DTW) distance measure [9] is a well-known shape-based similarity measure for time series data. It uses a dynamic programming technique to find an optimal warping path between two time series. Suppose we have two time series sequences, a sequence $X$ of length $n$ and a sequence $Y$ of length $m$. The distance is calculated by the following equations.

$$D(X_{1\ldots n}, Y_{1\ldots m}) = d(x_n, y_m) + \min \begin{cases} D(X_{1\ldots n-1}, Y_{1\ldots m-1}) \\ D(X_{1\ldots n}, Y_{1\ldots m-1}) \\ D(X_{1\ldots n-1}, Y_{1\ldots m}) \end{cases} \quad (1)$$

where $D(X_{1\ldots n}, \varnothing) = D(\varnothing, Y_{1\ldots n}) = \infty$, $D(\varnothing, \varnothing) = 0$, and $\varnothing$ is an empty sequence. Any distance metric can be used for $d(x_i, y_j)$, including $L_1$-norm, i.e., $d(x_i, y_j) = |x_i - y_j|$.

### 2.2 Global Constraint

Global constraint efficiently limits the optimal path to give a more suitable alignment. Recently, an R-K band [8], a general model of global constraints, has been proposed. R-K band represents a global constraint by a one-dimensional array R, i.e., $R = \langle r_1, r_2, \ldots, r_i, \ldots, r_n \rangle$, where $n$ is the length of time series, and $r_i$ is the height above the diagonal in $y$-axis and the width to the right of the diagonal in $x$-axis. Each $r_i$ value is arbitrary, making the R-K band an arbitrary-shaped global constraint.

### 2.3 Lower-bounding Function for DTW Distance

Although DTW outperforms many other distance measures, it is known to require huge computational complexity. Therefore, LB_Keogh has been proposed to speed up similarity search. $LB_{keogh}(Q, C)$ between the query sequence $Q = \langle q_1, q_2, \ldots, q_i, \ldots, q_n \rangle$ and a candidate sequence $C = \langle c_1, c_2, \ldots, c_i, \ldots, c_n \rangle$ can be computed as follows

$$LB_{keogh}(Q, C) = \sum_{i=1}^{n} \begin{cases} |c_i - u_i| & ;\text{if } c_i > u_i \\ |l_i - c_i| & ;\text{if } c_i > l_i \\ 0 & ;\text{otherwise} \end{cases} \quad (2)$$

where $u_i = \max\{q_{i-r_i}, \ldots, q_{i+r_i}\}$ and $l_i = \min\{q_{i-r_i}, \ldots, q_{i+r_i}\}$ are envelope elements calculated from a global constraint $R = \langle r_1, r_2, \ldots, r_i, \ldots, r_n \rangle$.

### 2.4 Uniform Scaling

Many research works [2][12] have been shown that when the uniform scaling technique is applied, performance, especially the accuracy, significantly increases. More specifically, uniform scaling technique shrinks/stretches a time series sequence $X = \langle x_1, x_2, \ldots, x_i, \ldots, x_n \rangle$ to a new time series sequence $Y = \langle y_1, y_2, \ldots, y_i, \ldots, y_m \rangle$, where $y_j = x_{\lceil j \cdot n/m \rceil}$. We also define a scaling factor $f$ as a ratio between length $m$ of new time series $Y$ and length $n$ of original time series $X$ or $f = m/n$, and define a scaling range $[f_{min}, f_{max}]$, where $f_{min}$ and $f_{max}$ are minimum and maximum scaling factors which give lengths $n_{min}$ and $n_{max}$, respectively.

### 2.5 Normalization

The two time series sequences are compared using any similarity measure; all the data should first be normalized. Z-normalization [3] has been proposed and widely used in time series data mining community, making mean and standard deviation values of the new time series sequence to be zero and one, respectively. Suppose we normalize time series sequence $X = \langle x_1, \ldots, x_i, \ldots, x_n \rangle$ to sequence $Y = \langle y_1, \ldots, y_i, \ldots, y_n \rangle$, we can simply formulate transformation function as $y_i = (x_i - \mu_x)/\sigma_x$, where $\mu_x$ and $\sigma_x$ are the mean and standard deviation of time series sequence $X$, respectively.

## 3. PROBLEM DEFINITION

In this paper, we focus on two main query problems on streaming time series data, i.e., optimal range query and optimal top-$k$ query. The objective of the optimal range query is to find non-overlapping normalized subsequences from a data stream, whose distance between a candidate sequence and a query sequence must be less than a threshold $\varepsilon$, where the query sequence is scaled and normalized under uniform scaling between scaling range $[f_{min}, f_{max}]$. On

the other hand, optimal top-$k$ query reports top-$k$ non-overlapping normalized subsequences. Nevertheless, the scaled query sequences and all candidate subsequences in the data stream must be normalized in order to return meaningful results. A naïve method to monitor incoming data stream first initializes a set of normalized scaled query sequences, and then candidate sequences are extracted from the data stream using sliding-window model. After normalization, distance calculation is performed on the extracted subsequences and non-overlapping optimal results are reported (if any). However, this naïve method requires as high as $O(n^3)$ time complexity for each new incoming streaming data point.

## 4. PROPOSED METHOD

Since the naïve method consumes too heigh time complexity, we propose a novel approach for subsequence matching which gives meaningful result. We call our proposed method as an MSM algorithm (Meaningful Subsequence Matching), which contains two new ideas, i.e., a multi-resolution lower-bounding function LB_GUN (Lower-Bounding function under Global constraint, Uniform scaling, and Normalization), and SSM (Scaling Subsequence Matrix) which incrementally estimates value of LB_GUN under global constraint, uniform scaling, and normalization in linear time while guaranteeing no false dismissals. Three following subsections of LB_GUN, SSM, and MSM algorithm are precisely described.

### 4.1 Lower-Bounding Distance under Global Constraint, Uniform Scaling, and Normalization (LB_GUN)

LB_GUN is a lower-bounding function of DTW distance extended from LB_Keogh [5] whose distance calculation can be done in linear time. Before calculation, LB_GUN first creates an envelope $E'$ from scaled and normalized envelopes. More specifically, three sequence sets are generated, i.e., sets of $Q^{\sim}$, $R^{\sim}$, and $E^{\sim}$. The scaled query set $Q^{\sim} = \{Q'_{n_{min}}, \ldots, Q'_k, \ldots, Q'_{n_{max}}\}$is first generated by scaling and normalizing a query sequence $Q$ to every normalized scaled query sequence $R'_k$, and the scaled global constraint $R^{\sim} = \{R'_{n_{min}}, \ldots, R'_k, \ldots, R'_{n_{max}}\}$ set is derived from scaling a specific global constraint set $R^{\sim}$ with all possible scaling lengths from $n_{min}$ to $n_{max}$. An envelope $E_k$ of a normalized scaled query sequence $Q'_k$ and a scaled global constraint $R'_k$ for sequence length $k$ is created as in LB_Keogh, and is stored in the envelope set $E^{\sim} = \{E'_{n_{min}}, \ldots, E'_k, \ldots, E'_{n_{max}}\}$. Then, $E'$ is generated by merging all envelopes in the set $E^{\sim}$ together, where $E^{\sim} = \{\langle u'_1, l'_1\rangle \ldots, \langle u'_i, l'_i\rangle, \ldots, \langle u'_{max}, l'_{max}\rangle\}$. To find lower-bounding distance between a query sequence $Q$ and a candidate sequence $C$ under global constraint, uniform scaling, and normalization, an envelope $E'$ of a query sequence $Q$ is generated as mention above. $LB_{GUN}(Q, C, n)$ is shown in Equation (3).

$$LB_{GUN}(Q,C,n) = \frac{1}{\sigma_{C_{1\ldots n}}}\left(\sum_{i=1}^{n}\alpha_i + \mu_{C_{1\ldots n}}\sum_{i=1}^{n}\beta_i\right) + \sum_{i=1}^{n}\gamma_i \quad (3)$$

$$\alpha_i = \begin{cases} c_i & ; c'_i \geq u'_i \\ -c_i & ; c_i \geq l'_i \\ 0 & ; \text{otherwise} \end{cases} \quad (4)$$

$$\beta_i = \begin{cases} -1 & ; c'_i \geq u'_i \\ 1 & ; c'_i \leq l'_i \\ 0 & ; \text{otherwise} \end{cases} \quad (5)$$

$$\gamma_i = \begin{cases} -u'_i & ; c'_i \geq u'_i \\ l'_i & ; c_i \leq l'_i \\ 0 & ; \text{otherwise} \end{cases} \quad (6)$$

where $\mu_{C_{1\ldots n}}$ and $\sigma_{C_{1\ldots n}}$ are arithmetic mean and standard deviation of data points 1 to $n$ of a candidate sequence $C$, $c'_i = (c_i - \mu_{C_{1\ldots i}})/\sigma_{C_{1\ldots i}}$, $n_{min}$ and $n_{max}$ are desired scaling lengths, and $n_{min} \leq n \leq n_{max}$.

### 4.2 Scaling Subsequence Matrix

SSM (Scaling Subsequence Matrix) is another important component in MSM algorithm. It stores lower-bounding distances determined by LB_GUN for each new incoming streaming data point st at time $t$ from data stream $S$. Suppose we have a query sequence $Q$; each element of the matrix contains five values, i.e., $v_{t,j}$, $w_{t,j}$, $x_{t,j}$, $y_{t,j}$, and $z_{t,j}$, calculated from time $t-j$ to time $t$. Therefore, values in matrix element $\langle t, j\rangle$ can be incrementally updated from the matrix element $\langle t-1, j-1\rangle$ according to the following equations.

$$v_{t,j} = v_{t-1,j-1} + \begin{cases} s_t & ; s'_t \geq u'_j \\ -s_t & ; s'_t \leq l'_j \\ 0 & ; \text{otherwise} \end{cases} \quad (7)$$

$$w_{t,j} = w_{t-1,j-1} + \begin{cases} -1 & ; s'_t \geq u'_j \\ 1 & ; s'_t \leq l'_j \\ 0 & ; \text{otherwise} \end{cases} \quad (8)$$

$$x_{t,j} = x_{t-1,j-1} + \begin{cases} -u'_j & ; s'_t \geq u'_j \\ l'_j & ; s'_t \leq l'_j \\ 0 & ; \text{otherwise} \end{cases} \quad (9)$$

$$y_{t-j} = y_{t-1,j-1} + s_t \quad (10)$$

$$Z_{t,j} = z_{t-1,j-1} + (s_t)^2 \quad (11)$$

$$lb_{t,j} = \frac{1}{\sigma_{t,j}}(v_{t,j} + \mu_{i,j} \cdot w_{t,j}) + x_{t,j} \qquad (12)$$

where $s_t' = \frac{s_t \mu_{t,j}}{\sigma_{t,j}}$, $\mu_{t,j} = \frac{y_{t,j}}{j}$, $\sigma_{t,j} = \sqrt{\frac{z_{t,j}}{j} - (\mu_{t,j})^2}$ , $u_j$ and $l_j$ are from an enveloped generated from a query sequence $Q$,$1 \leq j \leq n_{max}$, $n_{min} \leq j \leq n_{max}$, and $lb_{t,j}$ is a lower-bounding distance LB_GUN for an element $\langle t, j \rangle$.

### 4.3 Meaningful Subsequence Matching

Since SSM is updated at every arrival of new streaming data point st, our MSM algorithm can monitor both optimal range query and optimal top-$k$ query. More specifically, for optimal range query, MSM first calculates and updates values including lower-bounding distances in SSM, which is an estimation of LB_GUN and then checks whether best-so-far distance $d_{best}$ is smaller than threshold $\varepsilon$. If so, MSM reports an optimal subsequence when there is no overlapping subsequence, and MSM resets $d_{best}$ and values in SSM. For all $lb_{tj}$ which are smaller than $d_{best}$ in range from $n_{min}$ to $n_{max}$, LB_GUN and LB_Keogh are calculated and compared to $d_{best}$ respectively to prune off the DTW distance calculation. If they are not pruned by any lower-bounding distances, DTW distance is computed to update $d_{best}$ and the optimal subsequence's position. Additionally, MSM uses only two columns of SSM that are values in time $t$ and values in time $t-1$. All lower-bounding distances and DTW distance are normalized by dividing by $i$. The MSM algorithm for optimal range query is described in Table 1.

*Table 1: MSM Algorithm for optimal range query*

MSMOPTIMALRANGEQUERY ALGORITHM

```
1  Input: a new streaming data point s_t
2  Output: an optimal subsequence (if any)
3  update v_i, w_i, x_i, y_i and z_i for all i, 1 ≤ i ≤ n_max
   and lb_i for all i, n_min ≤ I ≤ n_max
4  if (d_best < ε and ∀_i, t_best^end ≤ t - i)
5    REPORT(d_best, S[t_best^end, t_best^start]
6    d_best = ∞
7    reset v_i, w_i, x_i, y_i and z_i = ∞ for all i, t_best^end > t - i
8  for (i = n_min to n_max)
9    if (lb_i ≤ d_best)
       if (LB_GUN(Q'_i, NORMALIZE(S[t - i : t])) < d_best
         if (LB_KEOGH(Q'_i, NORMALIZE(S[t - i : t])) < d_best
10         distance = DTW(Q'_i, NORMALIZE(S[t - i : t]))
11         if (distance ≤ d_best)
12           d_best = distance; t_best^end = t - i; t_best^start = t
13 substitute v_i', w_i', x_i', y_i' and z_i' for v_i, w_i, x_i, y_i, and z_i
```

MSM algorithm for optimal top-$k$ query is implemented based on the optimal range query. With a priority queue, MSM stores the $k$-best non-overlapping subsequence with DTW distance from the result of MSMOPTIMALRANGEQUERY. First, we initialize a threshold $\varepsilon$ to positive infinity. Then, for every new streaming data point $s_t$, the queue is updated, and the threshold $\varepsilon$ is set to the largest DTW distance in the queue. The MSM algorithm for optimal top-$k$ query is described in Table 2.

*Table 2: MSM Algorithm for optimal top-k query*

MSMOPTIMALTOPKQUERY ALGORITHM

```
1  Input: a new streaming data point s_t
2  Output: update set P of top-k subsequence
3  [C, d_C] = MSMOPTIMALRANGEQUERY(s_t, ε)
4  If(C ≠ NULL)
5    P.push(C, d_C)
6    if (size(P) > k)
7      P.pop()
8      ε = P.peek().d_C
```

## 5. EXPERIMENTAL EVALUATION

Since none of the current subsequence matching algorithms under DTW distance can handle the changes of data distribution, offset, and scaling, we compare our proposed method with naïve approach in terms of computational time only since our proposed method and the naïve method will both achieve the same accuracy. On the other hand, we compare our accuracy with SPRING, the best existing subsequence matching under DTW distance. Note that we do not compare our running time with that of SPRING; while SPRING will have smaller running time, its results are inaccurate due to lack of normalization, therefore is not a reasonable comparison.

Streaming datasets are generated by combining training data sequences from the UCR classification/clustering datasets [6] and synthesized random walk sequences. A stream is initialized with a random walk sequence, and then a training data sequence is appended to the stream. To smooth the stream, before concatenation, each sequence is offset by the last value of the stream. The dataset we used in the experiments are Aidac, Beef, CBF, Coffee, ECG200, Gun Point, Lighting7, Olive Oil, Trace and Synthetic Control which are represented by Data 1, Data 2, Data 3, Data 4, Data 5, Data 6, Data 7, Data 8, Data 9 and Data 10, respectively.

In the first experiment, we compare our MSM algorithm with naïve method in terms of computational cost by measuring the number of distance calculations. Fig.2 shows the numbers of all distance calculations by varying global constraint to 2, 4, 6, 8 and 10 respectively, and in Fig.3, scaling range $[f_{min}, f_{max}]$ are varied from [0.8, 1.2], [0.85, 1.15], [0.9, 1.1] and [0.95, 1.05] respectively. The numbers of all distance calculations are normalized to 100% which represent numbers of DTW calculations used in the Naïve method. As expected, MSM is much faster than the naïve method by a large margin. Additionally, in MSM, our multi-resolution lower-bounding function is efficiently used to filter out several candidate sequences in linear time while guaranteeing no false dismissals; therefore, MSM algorithm requires

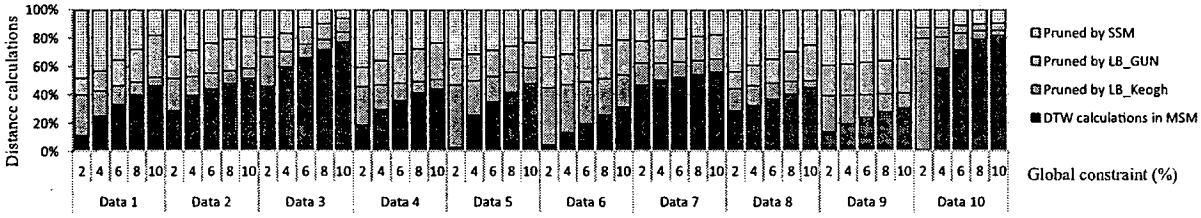



*Fig.2: DTW distance calculations filtered out by MSM with varying global constraints*

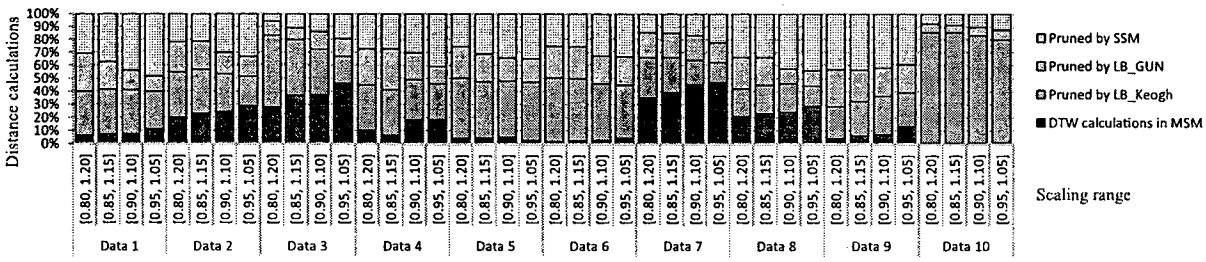



*Fig.3: DTW distance calculations filtered out by MSM with varying scaling ranges*

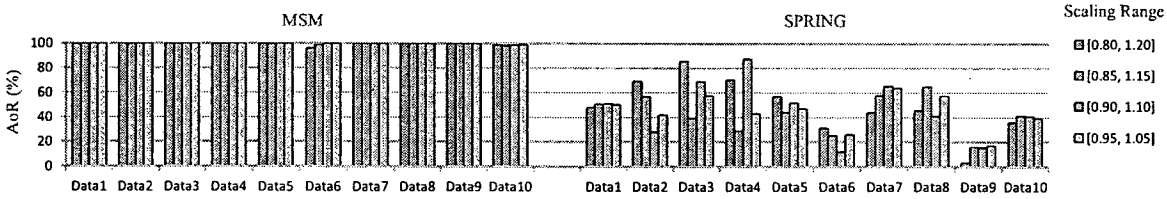



*Fig.4: MSM outperforms SPRING every scaling range in terms of AoR*

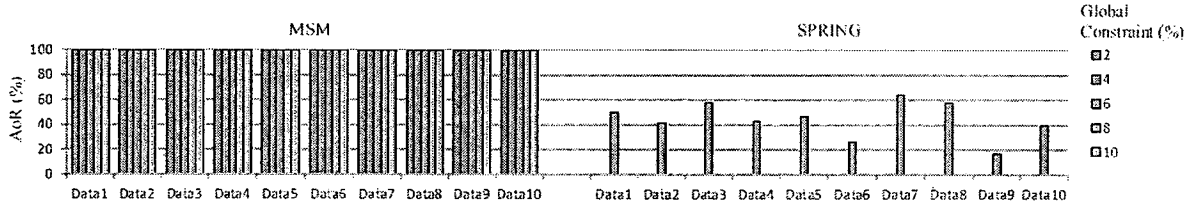



*Fig.5: MSM outperforms SPRING every global constraint value in terms of AoR*

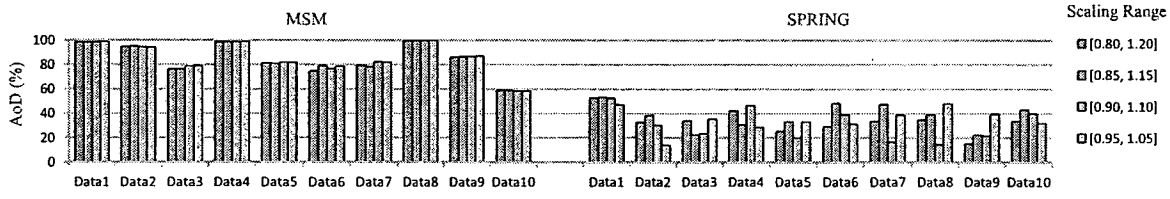



*Fig.6: MSM outperforms SPRING every scaling range in terms of AoD*

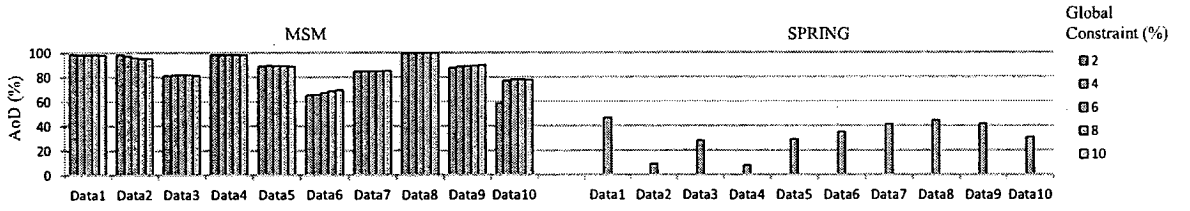



*Fig.7: MSM outperforms SPRING every global constraint value in terms of AoD*

only a small number of DTW distance calculations comparing with the naïve method.

Then, we compare our MSM algorithm with SPRING to measure performance in terms of accuracy, both Accuracy-on-Retrieval (AoR) and Accuracy-on-Detection (AoD). AoR reflects quality of an algorithm that is able to find the patterns in data stream; on the other hand, AoD reflects quality of the returned results. Suppose we have data stream $S$, a set of expected pattern sequences $E$, and a set of retrieved sequences $R$. We first define an overlapping subsequence. Let $S[ts : te]$ be the subsequence starting at $t_s$ and ending at $t_e$. Overlapping subsequence $O_{X,Y}$, where $X = S[a : b]$ and $Y = S[c : d]$, and overlap percentage $P_{X,Y}$ are defined as $O_{X,Y} = S[\min\{a,c\} : \min\{b,d\}]$ and $P_{X,Y} = \frac{|O_{X,Y}|}{\max\{b,d\} - \min\{a,c\} + 1}$, respectively. Both AoR and AoD can be defined over overlapping subsequence $O_{X,Y}$ and overlapping percentage $P_{X,Y}$ as $AoR = \frac{|\{O_{X,Y}|P_{X,Y} > p, X \in R, Y \in E\}|}{|E|}$ and $AoD = \frac{\sum\{P_{X,Y}|P_{X,Y} > p, X \in R, Y \in E\}}{|\{O_{X,Y}|P_{X,Y} > p, X \in R, Y \in E\}|}$, respectively,

where p is a threshold of PX,Y that defines a sequence in R as a discovered sequence. Fig.4 and Fig.5 compare AoRs of MSM and SPRING under various scaling ranges and global constraints, respectively. Fig.6 and Fig.7 illustrate AoDs on every scaling range and global constraint, respectively. The results show that MSM produces more meaningful result since SPRING does not support global constraint (illustrated as one single column of 100% global constraint in Fig.5 and Fig.7), uniform scaling, nor normalization.

## 6. CONCLUSION

This paper proposes a novel and meaningful subsequence matching algorithm, so called MSM (Meaningful Subsequence Matching), under global constraint, uniform scaling, and normalization. Two ideas have been introduced in MSM algorithm, i.e., a multi-resolution lower-bounding function LB_GUN (Lower-Bounding distance function under Global constraint, Uniform scaling, and Normalization, and a Scaling Subsequence Matrix (SSM) which estimates value of LB_GUN for each candidate subsequence. Our algorithm can update lower-bounding distance incrementally under normalization, while guaranteeing no false dismissals in linear time. With these two ideas, MSM algorithm can efficiently monitor data stream and can answer both optimal range query and optimal top-$k$ query problems. Since none of the current algorithm produces meaningful result, we evaluate our proposed method comparing with the naive method in terms of time consumption and SPRING, the best existing subsequence matching under DTW distance, in terms of accuracies. As expected, our MSM algorithm is much faster and more accurate by a very large margin.

## 7. ACKNOWLEDGMENT

This research is partially supported by the Thailand Research Fund (Grant No. MRG5380130), the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0319/2551 to S. Rodpongpun and PHD/0141/2549 to V. Niennattrakul).

## References

[1] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos, "Approximate Embedding-based Subsequence Matching of Time Series," *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pp. 365-378, 2008.

[2] A. W. C. Fu, E. J. Keogh, L. Y. H. Lau, C. A. Ratanamahatana, and R. C. W. Wong, "Scaling and Time Warping in Time Series Querying," *The VLDB Journal*, vol. 17, pp. 899-921, 2008.

[3] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley, *PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals*, Circulation 101(23):e215-e220, 2000.

[4] J. Han and M. Kamber, Data Mining: *Concepts and Techniques*: Morgan Kaufmann, 2001.

[5] E. J. Keogh and C. A. Ratanamahatana, "Exact Indexing of Dynamic Time Warping," *Knowledge and Information Systems*, vol. 7, pp. 358-386, 2005.

[6] E. J. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "UCR Time Series Classification/ Clustering Page," http://www.cs.ucr.edu/ eamonn/ time_series_data.

[7] V. Niennattrakul, D.Wanichsan, and C. A. Ratanamahatana, "Accurate Subsequence Matching on Data Stream under Time Warping," *PAKDD 2009 International Workshops (PAKDD'09)*, pp. 156-167, 2010.

[8] C. A. Ratanamahatana and E. J. Keogh, "Making Time-Series Classification More Accurate Using Learned Constraints," *Proceedings of the $4^{th}$ SIAM International Conference on Data Mining (SDM'04)*, pp. 11-22. 2004.

[9] C. A. Ratanamahatana and E. J. Keogh, "Three Myths about Dynamic Time Warping Data Mining," *Proceedings of the $5^{th}$ SIAM International Conference on Data Mining (SDM'05)*, pp. 506-510, 2005.

[10] Y. Sakurai, C. Faloutsos, and M. Yamamura, "Stream Monitoring under the Time Warping Distance," *Proceedings of the $23^{th}$ International Conference on Data Engineering (ICDE'07)*, pp. 1046-1055, 2007.

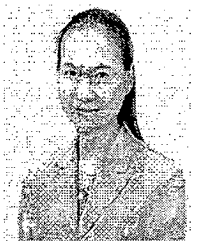
[11] D. Sart, A. Mueen, W. Najjar, E. Keogh, and V. Niennattrakul, "Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs," *IEEE International Conference on Data Mining (ICDM'10)*, pp. 1001-1006, 2010.

[12] D. Yankov, E. J. Keogh, J. Medina, B. Chiu, and V. B. Zordan, "Detecting Time Series Motifs under Uniform Scaling," *Proceedings of the $15^{th}$ ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pp. 844-853, 2007.

[13] P. Zou, L. Su, Y. Jia, W. Han, and S. Yang, "Fast Similarity Matching on Data Stream with Noise," *Proceedings of the $24^{th}$ International Conference on Data Engineering Workshops (ICDEW'08)*, pp. 194-199, 2008.

**Sura Rodpongpun** is a Ph.D. student in Computer Engineering at Chulalongkorn University. He received his B.Eng. in Computer Engineering from Mahidol University. He was granted scholarships from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program. His research interests are time series data mining, machine learning, and artificial intelligence.

**Vit Niennattrakul** received his B.Eng and Ph.D. in Computer Engineering from Chulalongkorn University in 2006 and 2011, respectively. He was granted scholarships from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program from 2007 to 2011, Chulalongkorn University Graduate Scholarship to Commemorate the 72nd Anniversary of His Majesty King Bhumibol Adulyadej, and the 90th Anniversary of Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment Fund). His research interests include time series data mining, machine learning, and natural language processing.



**Chotirat Ann Ratanamahatana** is an assistant professor in Computer Engineering Department, Chulalongkorn University, Bangkok, Thailand. She received her undergraduate and graduate studies from Carnegie Mellon University and Harvard University, respectively, and received her Ph.D. from the University of California, Riverside. Her research interests include data mining, information retrieval, machine learning, and human-computer interaction.

# Shape-Based Clustering for Time Series Data

Warissara Meesrikamolkul, Vit Niennattrakul,
and Chotirat Ann Ratanamahatana

Department of Computer Engineering, Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok, Thailand 10330
{g53wms,g49vnn,ann}@cp.eng.chula.ac.th

**Abstract.** One of the most famous algorithms for time series data clustering is $k$-means clustering with Euclidean distance as a similarity measure. However, many recent works have shown that Dynamic Time Warping (DTW) distance measure is more suitable for most time series data mining tasks due to its much improved alignment based on shape. Unfortunately, $k$-means clustering with DTW distance is still not practical since the current averaging functions fail to preserve characteristics of time series data within the cluster. Recently, Shape-based Template Matching Framework (STMF) has been proposed to discover a cluster representative of time series data. However, STMF is very computationally expensive. In this paper, we propose a Shape-based Clustering for Time Series (SCTS) using a novel averaging method called Ranking Shape-based Template Matching Framework (RSTMF), which can average a group of time series effectively but take as much as 400 times less computational time than that of STMF. In addition, our method outperforms other well-known clustering techniques in terms of accuracy and criterion based on known ground truth.

**Keywords:** Time Series, Clustering, Shape-based Averaging.

## 1 Introduction

Time series data mining is increasingly an active research area since time series data are ubiquitous, appearing in various domains including medicine [15], geology [13], etc. One of its main mining tasks is clustering, which is a method to seperate unlabeled data into their natural groupings. In many applications related to time series data [14], $k$-means clustering [2] is generally used with the Euclidean distance function and amplitude averaging (arithmetic mean) as an averaging method.

Although the Euclidean distance is popular and simple, it is not suitable for time series data because its distance between two sequences is calculated in one-to-one manner. As a result, $k$-means with Euclidean distance does not cluster well because time shifting among data sequences in the same class usually occurs. In time series mining, especially in time series classification, Dynamic Time Warping (DTW) [1] distance has been proved to give more accurate results than Euclidean distance. Unfortunately, $k$-means clustering with the DTW

P.-N. Tan et al. (Eds.): PAKDD 2012, Part I, LNAI 7301, pp. 530–541, 2012.

distance still does not work practically [8][7] because current averaging function does not return a characteristic-preserving averaging result. Traditional $k$-means clustering fails to return a correct clustering result since this cluster centers do not reflect characteristics of the data, as shown in Fig. 1. In this work, we will demonstrate that our proposed method can resolve this problem.

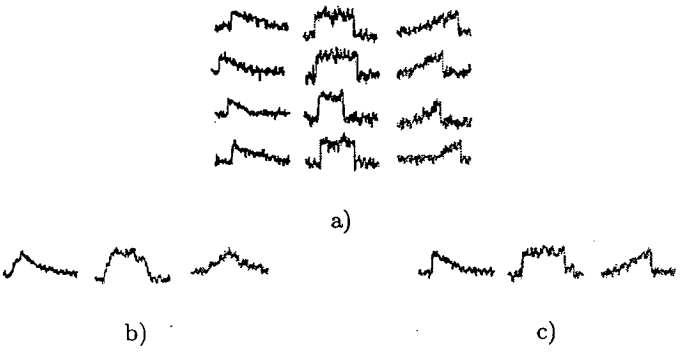



**Fig. 1.** a) Sample 3-class CBF data [3] and its cluster centers from b) traditional $k$-means clustering and from c) our proposed method

We propose a novel method called Shape-based Clustering for Time Series (SCTS) which incorporates $k$-means clustering and DTW distance measure, together with our new averaging method, called Ranking Shape-based Template Matching Framework (RSTMF) extended from Shape-based Template Matching Framework (STMF) [10] for classification. Unlike STMF, our RSTMF uses distances from clustering to approximate an order of sequences to be averaged, giving a few orders of magnitude speedup comparing to STMF. Our evaluation also shows that our proposed method outperforms other well-known clustering techniques in terms of accuracy and criterion based on known ground truth. In addition, the accuracy of our proposed method can future improve when a global constraint [11] is utilized in distance calculation and data averaging.

The rest of the paper is organized as follows. In section 2 and 3, we offer background knowledge and related works. In section 4, we explain our new framework for time series clustering, which is Shape-based Clustering for Time Series (SCTS). The experiments and results are shown in section 5. Finally, conclusions are provided in section 6.

## 2 Background

This section provides background knowledge on $k$-means clustering, Dynamic Time Warping (DTW) distance measure, and global constraint.

### 2.1 $K$-means Clustering

$K$-means clustering [2] is a well-known and very simple partitioning clustering algorithm. Its algorithm tries to group similar data into the same cluster by using an objective function that minimizes a sum of squared errors between a cluster center to its members. The algorithm is done as follows:

1. Initialize $k$ cluster centers.
2. Measure the similarity between each data and all cluster centers and assign data into the most similar cluster.
3. Calculate a new cluster center of every cluster using an averaging function.
4. Repeat steps 2 and 3 until the cluster membership does not change.

$K$-means clustering consists of two major subroutines, which are a distance function to measure the similarity between data sequences and an averaging function to return a new cluster center. Generally, most time series clustering works use Euclidean distance and amplitude averaging method. However, both cluster centers and their cluster members are inaccurate. In this work, we resolve this problem by using the DTW distance measure with our newly proposed averaging method called RSTMF.

### 2.2 Dynamic Time Warping (DTW) Distance Measure

DTW distance [1] is an accurate similarity measurement which is generally used for time series data [9], especially in classification [6]. An optimal alignment and distance between two sequences $P = \langle p_1, \ldots, p_i, \ldots, p_n \rangle$ and $Q = \langle q_1, \ldots, q_j, \ldots, q_m \rangle$ can be determined as follows.

$$DTW(P,Q) = \sqrt{dist(p_n, q_m)} \tag{1}$$

$$dist(p_i, q_j) = (p_i - q_j)^2 + min \begin{cases} dist(p_{i-1}, q_j) \\ dist(p_i, q_{j-1}) \\ dist(p_{i-1}, q_{j-1}) \end{cases} \tag{2}$$

DTW distance is computed through dynamic programming to discover the minimum cumulative distance of each element in $n \times m$ matrix. In addition, the warping path between two sequences can be found by tracing back from the last cell.

In this work, DTW distance is used to measure the similarity between each time series data and cluster centers to give more accurate results.

### 2.3 Global Constraint

The global constraint is used when we need to limit the amount of warping in the DTW alignment. In some applications such as speech recognition [12], two data sequences are considered the same class when only small time shifting occurs; so,

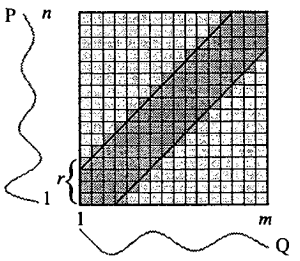



**Fig. 2.** The warping window of $P$ and $Q$ is limited by the global constraint of size $r$

the global constraint is used to align the sequences more precisely. The Sakoe-Chiba band [12], one of the most popular global constraints, has been originally proposed for speech community and also has been used in various tasks in time series mining [11]. The size of the warping window is defined by $r$ (as shown in Fig. 2), the percentage of the time series' length, which is symmetric in both above and on the right of a diagonal. In this work, we will show in experiments that the global constraint plays an important role in improving the accuracy.

## 3 Related Work

In the past few decades, there are many clustering techniques proposed to cluster time series data [5], for example, agglomerative hierarchical clustering [13], which merges most similar objects until all objects are in the cluster. However, this technique is still inaccurate, especially when outliers are present.

Another popular clustering technique is partitional clustering, which tries to minimize an objective function. The well-known algorithms are $k$-medoids and $k$-means clustering, which are different in their approaches to find new cluster centers. For $k$-medoids clustering application [4], DTW distance is used as a similarity measure among data sequences, and a sequence with minimum sum of distance to the rest of the sequences in the cluster is selected as a new cluster center. However, medoid is not always a centroid of a cluster, so the sequences can be assigned to wrong clusters.

In contrast to $k$-medoids clustering, $k$-means clustering mostly uses Euclidean distance as a distance metric, and an arithmetic mean or amplitude averaging is simply used to find a new cluster center [14]. Although the DTW distance is more appropriate for time series data, there currently is no DTW averaging method that provides a satisfied averaging result.

According to this, many research works have tried to improve the quality of the averaging result. Shape-based Template Matching Framework (STMF) [10] was recently introduced to average time series sequences. Table 1 shows the algorithm of this framework; the most similar pair of sequences is averaged by Cubic-spline Dynamic Time Warping (CDTW) algorithm (in line 6).

**Table 1.** Shape-based Template Matching Framework algorithm [10]

| **Algorithm** STMF($D$) | |
|---|---|
| 1. | $D$ is the set of time series data to be averaged |
| 2. | initialize weight $\omega = 1$ for every sequences in $D$ |
| 3. | while(size($D$) $> 1$) |
| 4. | $\{C_1, C_2\}$ = the most similar pair of sequences in $D$ |
| 5. | $Z$ = CDTW($C_1$, $C_2$, $\omega_{C_1}$, $\omega_{C_2}$) |
| 6. | $\omega_Z = \omega_{C_1} + \omega_{C_2}$ |
| 7. | add $Z$ to $D$ |
| 8. | remove $C_1$, $C_2$ from $D$ |
| 9. | end while |
| 10. | return $Z$ |

Given $C_1$ and $C_2$ as the most similar sequences, first, we find the warping path between these two sequences. The variables $c_{1i}$ and $c_{2j}$ are elements of $C_1$ and $C_2$, which are warped. The averaged sequence $Z$, which has coordinates $z_{k_x}$ and $z_{k_y}$ can be computed as follows.

$$z_{k_x} = \frac{\omega_{c_1} c_{1i} + \omega_{c_2} c_{2j}}{\omega_{c_1} + \omega_{c_2}} \tag{3}$$

$$z_{k_y} = \frac{\omega_{c_1} c_{1i_x} + \omega_{c_2} c_{2j_y}}{\omega_{c_1} + \omega_{c_2}} \tag{4}$$

In equations 3 and 4, $\omega_{c_1}$ and $\omega_{c_2}$ are the weight of the sequences $C_1$ and $C_2$, respectively. After we get the result, a number of points in the averaged sequence is re-sampled by using cubic-spline interpolation [10]. As shown in Fig. 3a), the averaging result from DTW averaging gives a sequence with 9 unequally spaced data points, whereas in Fig. 3b), the sequence is resampled with cubic spline interpolation to obtain a sequence of 7 equally spaced data points.

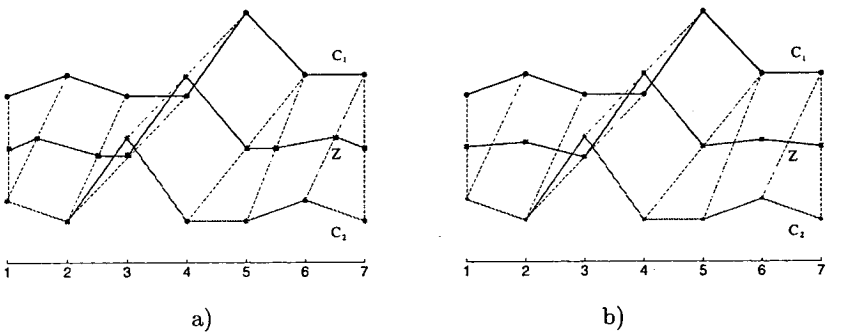



**Fig. 3.** The average sequences between $C_1$ and $C_2$ using DTW alignment a) before applying cubic spline interpolation and b) after applying cubic spline interpolation

However, according to this framework, finding the most similar pair for each time of averaging is enormously computationally expensive because the DTW distance of every pair of the sequences must be computed. Therefore, our RSTMF will mainly focus on improving its time complexity by estimating an order of sequences before averaging while maintaining the accuracy of the averaging results.

## 4 Shape-Based Clustering for Time Series (SCTS)

In this paper, we propose Shape-based Clustering for Time Series (SCTS) by incorporating $k$-means clustering and DTW distance, together with a novel averaging function, Ranking Shape-based Template Matching Framework (RSTMF). Although STMF can still be used to determine a cluster center, it is computationally expensive; therefore, computational time of $k$-means clustering significantly increase.

We provide an overview of the proposed clustering algorithm in Table 2; the DTW distance is used instead of the Euclidean distance in a membership assignment process. After we finished assigning each data sequence into the most similar cluster, RSTMF is utilized to average all of the sequences within each cluster until all cluster centers are updated. Unlike STMF, RSTMF approximates an order of averaged sequences by looking at the $Dist$ value, which is the DTW distance between data sequences in $M$ and all cluster centers in $C$. Accordingly, RSTMF can provide the average sequence by using less computation time than that of STMF, which calculates the distance between every pair of data and the most similar pair of sequences is averaged, making it very computationally expensive.

Table 3 shows our RSTMF averaging algorithm, which determines a cluster center by using Cubic-spline Dynamic Time Warping (CDTW) [10] to average a pair of time series sequences. RSTMF utilizes $Dist$ to approximate a similarity distance between every sequence pair, defined by $dist_{approx}$. After that, CDTW is used to average a pair of sequences with the minimum $dist_{approx}$ value. Then, we update $S$ and continue the averaging until only one sequence remains.

In RSTMF algorithm, the $dist_{approx}$ between each pair of the sequences can be computed by using the $Dist$ value. Suppose $P$ and $Q$ are data sequences in $M$, we have $Dist_{M_P,...} = \langle Dist_{M_P,C_1}, \ldots, Dist_{M_P,C_k}, \ldots, Dist_{M_P,C_K} \rangle$ and $Dist_{M_Q,...} = \langle Dist_{M_Q,C_1}, \ldots, Dist_{M_Q,C_k}, \ldots, Dist_{M_Q,C_K} \rangle$ where $Dist_{M_P,C_k}$ and $Dist_{M_Q,C_k}$ are the distance between $P$ or $Q$ and its $k^{th}$ cluster center, and $K$ is a number of cluster. By applying the triangular inequality theorem, $p_k$ and $q_k$ are assumed to be two sides of a triangle. Then, the $dist_{approx}$ of $P$ and $Q$, which is another side of the triangle, can be approximated by equation 5 and collected into $S$.

$$dist_{approx}(Dist_{M_P,...}, Dist_{M_Q,...}) = \max_{1 \le k \le K} |Dist_{M_P,C_k} - Dist_{M_Q,C_k}| \quad (5)$$

After finishing an averaging of two sequences, we insert the resulting sequence into $M$ and delete these two sequences. Then, we update $S$ by using the algorithm in Table 4.

**Table 2.** Shape-based Clustering for Time Series (SCTS)

| | **Algorithm** SCTS($D$, $K$) |
|---|---|
| 1. | $D$ is the set of time series data |
| 2. | $C$ is the set of cluster centers |
| 3. | $K$ is the number of cluster in $C$ |
| 4. | $M$ is the set of data in each cluster |
| 5. | $Dist$ is the matrix of the distance between data sequences and all cluster centers |
| 6. | initialize $C$ as cluster centers of $K$ clusters |
| 7. | do |
| 8. | for $i$ = 1:size($D$) |
| 9. | for $k$ = 1:$K$ |
| 10. | $Dist_{D_i,C_k}$ = DTW($D_i$, $C_k$) |
| 11. | end for |
| 12. | if($Dist_{D_i,C_k}$ is minimal) |
| 13. | assign $D_i$ into $M_k$ |
| 14. | end if |
| 15. | end for |
| 16. | for $k$ = 1:$K$ |
| 17. | $C_k$ = RSTMF($M_k$, $Dist$) |
| 18. | end for |
| 19. | while(the cluster membership changes) |
| 20. | return the cluster members and the cluster centers |

**Table 3.** The RSTMF algorithm

| | **Algorithm** RSTMF($M$, $Dist$) |
|---|---|
| 1. | $M$ is the set of data in each cluster |
| 2. | $Dist$ is the matrix of the distance between data sequences and all cluster centers |
| 3. | $S$ is the matrix of the distance between data sequences in $M$ |
| 4. | initialize weight $\omega$ = 1 for every sequences in $M$ |
| 5. | for $i$ = 1:size($M$) |
| 6. | for $j$ = $i$+1:size($M$) |
| 7. | $S_{M_i,C_j}$ = $S_{M_j,C_i}$= $dist_{approx}$($Dist_{M_i}$,... ,$Dist_{M_j}$,... ) |
| 8. | end for |
| 9. | end for |
| 10. | while(size($M$) > 1) |
| 11. | $S_{M_i,C_j}$ = minimum value in $S$ |
| 12. | $M_z$ = CDTW($M_i$, $M_j$, $\omega_{M_i}$, $\omega_{M_j}$) |
| 13. | $\omega_{M_z}$ = $\omega_{M_i}$+ $\omega_{M_j}$ |
| 14. | add $M_z$ to $M$ |
| 15. | UPDATE($S$, $i$, $j$, $z$) |
| 16. | remove $M_i$, $M_j$ from $M$ |
| 17. | end while |
| 18. | return $M_z$ |

**Table 4.** The UPDATE algorithm

| **Algorithm** UPDATE($S$, $a$, $b$, $z$) |
|---|
| 1. $S$ is the matrix of the distance between data sequences in $M$ |
| 2. for $i$ = 1:size($S$) |
| 3. $S_{M_z,M_i} = S_{M_i,M_z} = \min(S_{M_a,M_i}, S_{M_b,M_i})$ |
| 4. end for |
| 5. remove $S_{M_a,\ldots}$, $S_{\ldots,M_a}$, $S_{M_b,\ldots}$, $S_{\ldots,M_b}$ from $S$ |

By using the $dist_{approx}$ and the UPDATE method, our RSTMF can achieve large speedup because we can estimate an order of the sequences before averaging. In contrast, the original STMF needs to calculate the DTW distance to select the most similar pair of the sequences every time of averaging.

## 5 Experiments and Results

In this work, we evaluate our method by comparing it with other clustering techniques, which are typical $k$-means clustering with the Euclidean distance and amplitude averaging function, $k$-medoids clustering with the DTW distance [4], and $k$-hierarchical clustering [13] using both the Euclidean and the DTW distance. We compare our SCTS using RSTMF with that using the original STMF. Our experiments are evaluated on ten datasets from the UCR datasets classification/clustering archive [3] in diverse domains, as shown in Table 5.

**Table 5.** The details of datasets

| Datasets | Number of classes | Length of data | Size of training set | Size of test set |
|---|---|---|---|---|
| Synthetic Control | 6 | 60 | 300 | 300 |
| Trace | 4 | 275 | 100 | 100 |
| Gunpoint | 2 | 150 | 50 | 150 |
| Lightning-2 | 2 | 637 | 60 | 61 |
| Lightning-7 | 7 | 319 | 70 | 73 |
| ECG | 2 | 96 | 100 | 100 |
| Olive Oil | 4 | 570 | 30 | 30 |
| Fish | 7 | 463 | 175 | 175 |
| CBF | 3 | 128 | 30 | 900 |
| Face Four | 4 | 350 | 24 | 88 |

We execute each algorithm for 40 times with random initial cluster centers, and the $k$ value is set to the a number of classes in each dataset. With the luxury of labeled datasets used in all experiments, an accuracy, which is the number of correctly assigned data sequences in all clusters, is used evaluation. Fig. 4 shows the accuracy of our proposed method, comparing other well-known clustering methods mentioned above. According to the results, our method outperforms others in almost all datasets.

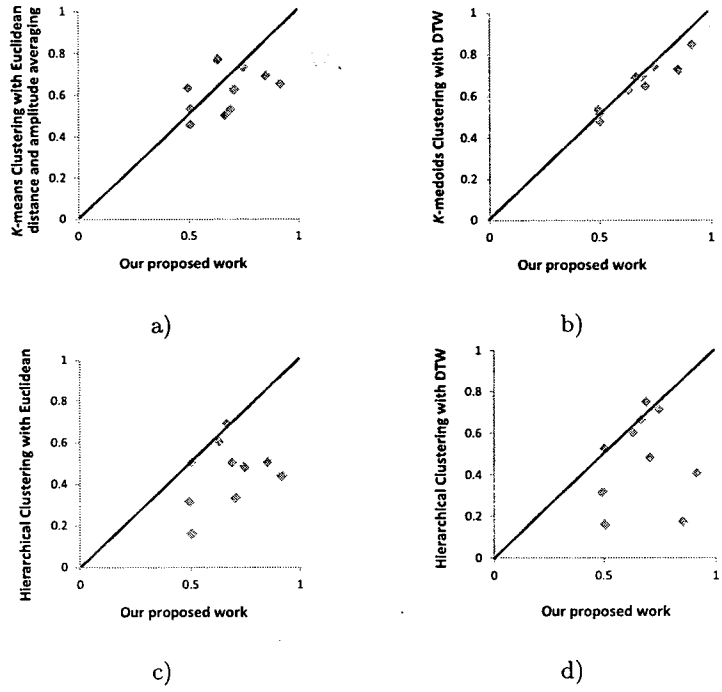



**Fig. 4.** The accuracy of our RSTMF method on 10 datasets, comparing with a) general $k$-means clustering, b) $k$-medoids clustering, and $k$-hierarchical clustering using c) the Euclidean distance and d) the DTW distance, respectively

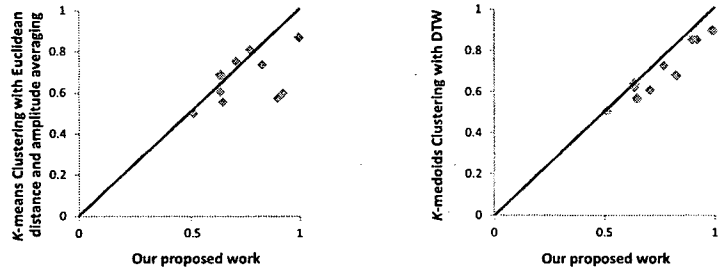
To re-emphasize our finding, we also use another criterion based on known ground truth [5] to measure a similarily between two sets of clusters, i.e., ground-truth clusters and results from clustering algorithms. Suppose $G$ and $C$ are sets of $k$ ground truth clusters and the clusters from our clustering technique. The similarity between $G$ and $C$ is calculated by the following equations.

$$Sim(G, C) = \frac{1}{k}\sum_{i=1}^{k} \max_{1 \le j \le k} Sim(G_i, C_j) \tag{6}$$

$$Sim(G_i, C_j) = \frac{2\,|G_i \cap C_j|}{|G_i| + |C_j|} \tag{7}$$

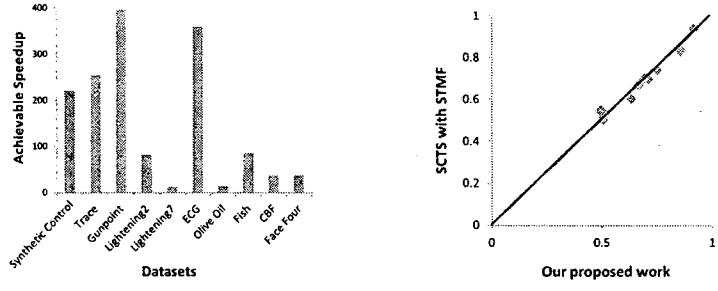
In Fig. 5, we compare our proposed work with the general $k$-means clustering and the $k$-medoids clustering using this criterion. The results show that the clusters obtained from our method are more similar to the ground-truth clusters because the RSTMF averaging method does give the new cluster centers that represent the overall charactheristic of the data within each cluster.

Furthermore, RSTMF can reduce the time complexity by a few orders of magnitude (as shown in Fig. 6a), while still providing comparable accuracy to STMF (as shown in Fig. 6b).




a) b)

**Fig. 5.** The criterion based on known ground truth, comparing our proposed method with a) general $k$-means clustering and b) $k$-medoids clustering




a) b)

**Fig. 6.** a) The speedup achieved by our proposed work. b) The accuracy of our proposed work comparing with that using STMF.

In some cases, it appears that SCTS with DTW distance achieves a lower accuracy than the general $k$-means clustering. In an attempt to alleviate this drawback, we experiment on the global constraint parameter of DTW, Sakoe-Chiba band. We can improve the clustering accuracy, comparing with the original $k$-means clustering (warping window size is 0%). Fig. 7 shows the accuracy of our proposed RSTMF and STMF, which are comparable, as warping window sizes vary. In almost datasets, the larger warping window size does not always provide the better accuracy; so, the appropriate warping window size is around 20%. However, in some dataset such as ECG, the wider warping window can lead to pathological warping and make the accuracy of clustering decreases.

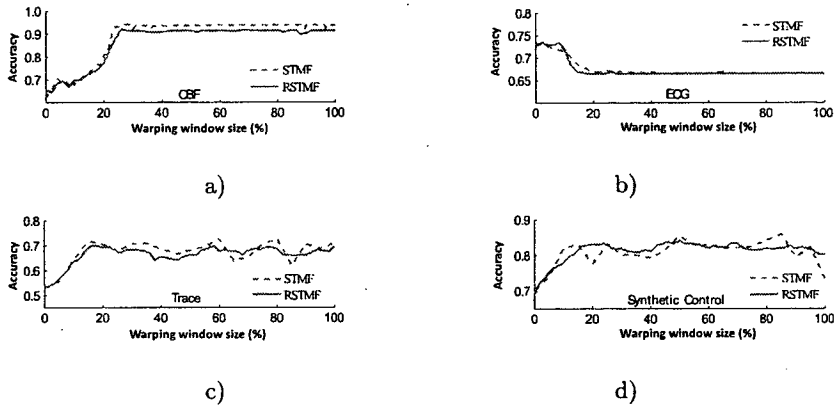



**Fig. 7.** The accuracy of Shape-based clustering using STMF and our proposed RSTMF of a) CBF, b) ECG, c) Trace, and d) Synthetic Control datasets

## 6 Conclusion

In this paper, we propose time series data clustering technique called Shape-based Clustering for Time Series (SCTS), which incorporates $k$-means clustering with a novel averaging method called Ranking Shape-based Template Matching Framework (RSTMF).

Comparing with the other well-known clustering algorithms, our SCTS yields better cluster results in terms of both accuracy and the criterion based on known ground truth because our RSTMF averaging function provides cluster centers that preserve characteristics of data sequences within the cluster (as shown in Fig. 8). Furthermore, RSTMF does gives a comparable sequence averaging result while consuming much less computational time than STMF in a few orders of magnitude; therefore, RSTMF is practically applied in clustering algorithm. We also used global constraint to increase an accuracy of our clusters. The results show that our SCTS can provide more accurate clustering when the width of warping window is about 20% of time series length.

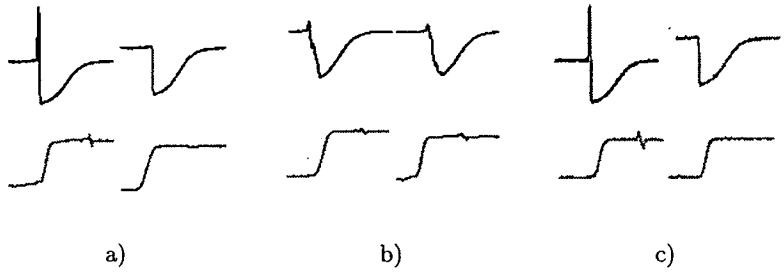



**Fig. 8.** The cluster centers obtained from a) our proposed method and b) the original $k$-means clustering of c) sample 4-class Trace data

**Acknowledgements.** This research is partially supported by the Thailand Research Fund (Grant No. MRG5380130), the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0141/2549 to V. Niennattrakul and C.A. Ratanamahatana), and CU Graduate School Thesis Grant.

## References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proceedings of AAAI Workshop on Knowledge Discovery in Databases, pp. 359–370 (1994)
2. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In: Proceedings of the International Conference on Machine Learning (ICML 1998), pp. 91–99 (1998)
3. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A. (2011), `http://www.cs.ucr.edu/~eamonn/time_series_data`
4. Liao, T.W., Bodt, B., Forester, J., Hansen, C., Heilman, E., Kaste, R.C., O'May, J.: Understanding and projecting battle states. In: Proceedings of 23rd Army Science Conference (2002)
5. Liao, T.W.: Clustering of time series data-a survey. Pattern Recognition, 1857–1874 (2005)
6. Meesrikamolkul, W., Niennattrakul, V., Ratanamahatana, C.A.: Multiple shape-based template matching for time series data. In: Proceedings of the 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2011), pp. 464–467 (2011)
7. Niennattrakul, V., Ratanamahatana, C.: On clustering multimedia time series data using k-means and dynamic time warping. In: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering, pp. 733–738 (2007)
8. Niennattrakul, V., Ratanamahatana, C.A.: Inaccuracies of Shape Averaging Method Using Dynamic Time Warping for Time Series Data. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 513–520. Springer, Heidelberg (2007)
9. Niennattrakul, V., Ruengronghirunya, P., Ratanamahatana, C.: Exact indexing for massive time series databases under time warping distance. Data Mining and Knowledge Discovery 21, 509–541 (2010)
10. Niennattrakul, V., Srisai, D., Ratanamahatana, C.A.: Shape-based template matching for time series data. Knowledge-Based Systems 26, 1–8 (2011)
11. Ratanamahatana, C.A., Keogh, E.: Making time-series classification more accurate using learned constraints. In: Proceedings of SIAM International Conference on Data Mining (SDM 2004), pp. 11–22 (2004)
12. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech and Signal Processing, 43–49 (1978)
13. Shumway, R.H.: Time-frequency clustering and discriminant analysis. Statistics and Probability Letters, 307–314 (2003)
14. Vlachos, M., Lin, J., Keogh, E., Gunopulos, D.: A wavelet-based anytime algorithm for k-means clustering of time series. In: Proceedings of Workshop on Clustering High Dimensionality Data and Its Applications, pp. 23–30 (2003)
15. Wismuller, A., Lange, O., Dersch, D.R., Leinsinger, G.L., Hahn, K., Pütz, B., Auer, D.: Cluster analysis of biomedical image time-series. International Journal of Computer Vision, 103–128 (2002)

# The Proper Length Motif Discovery Algorithm

Sorrachai Yingchareonthawornchai, Haemwaan Sivaraks, Sura Rodpongpun, Chotirat Ann Ratanamahatana
Department of Computer Engineering
Chulalongkorn University
Phayathai Rd., Pathumwan, Bangkok, Thailand, 10330
{sorrachai.y, haemwaan.s}@student.chula.ac.th, {g53srd, ann}@cp.eng.chula.ac.th

***Abstract***— *Motif discovery algorithm or finding of frequently occurring patterns, one of the fundamental data mining tasks, has drawn numerous attentions in the past decade. However, all of the previous attempts require some predefined parameters, e.g., a length of the underlying patterns. Unfortunately, such information is very difficult to determine. The next problem is that once the motifs are discovered in various lengths, it is exceedingly arduous to rank them. In this paper, we propose a novel parameter-free algorithm to discover time series motifs with a proper length. Our ranking scheme to determine the best motifs is based on an ability to compress the time series data by its motif. Extensive experiments in both planted and real datasets confirm the validity and effectiveness of our algorithm.*

***Keywords: motif discovery; time series mining.***

## I. INTRODUCTION

Motif discovery algorithm [1][6][7][8][9][15][16] is basically a search algorithm for patterns within time series sequences. It has the implications in higher level data mining tasks such as clustering and anomaly detection [10][11][12][21]. There are a plethora of motif discovery algorithms with various techniques—the great advancement. For example, MK motif discovery algorithm has utilized the use of early abandoning to prune off many unnecessary search spaces. The time complexity of the algorithm has reduced to be essentially linear [2]. However, users do suffer from selecting a set of parameters[13]; an initial window size is a typical one. Therefore, a parameter-free motif discovery algorithm is highly in demand. A common approach is to find a motif at every possible window size. However, the problem yet remains, as there is no criterion to judge the quality of the solutions among different window sizes, not to mention the cost in terms of time required to perform on every possible window size. [9] proposed a fair method of ranking system that allows all possible motifs compete across window size. The algorithm, nonetheless, costs an over-pruning problem— yielding mediocre quality of answers. To address and resolve these problems, our paper proposes a novel parameter-free algorithm. Our first priority is to propose an algorithm to measure and assure the quality of the solutions. The paramount mechanism is to use compression as a ranking scheme. In this work, the minimum description length [10] (MDL) concept is used for motif discovery. The MDL allows parameter-free nature. Furthermore, it is quite effortless to utilize parallel computing techniques to speed up the algorithm. The extensive experimental results confirm our proposed compression-based ranking scheme. Fig.1 shows the typical total *bitsave* plotted for every window size of the time series dataset.

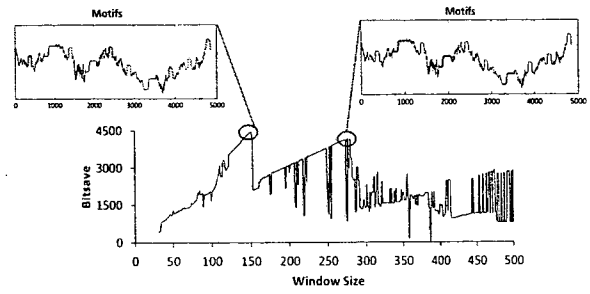



Figure 1. A plot of total bitsave or fitness of each window size. Notice there are two peak values which match the size of various classes of the motif.

## II. DEFINITION AND NOTATION

In order to understand terminology in this context, the definitions are clarified. First of all, we begin by defining the data type in this context, a *time series*:

**Definition 1:** A *time series* $T$ of length $n$ is an ordered set of real numbers of a sequence $t_1, t_2, t_3, ..., t_n$.

We are interested in finding *subsequences* or patterns of the time series:

**Definition 2:** A *subsequence* $C$ of length m of the time series $T$ of length $n$ is a subset of an ordered sequence $t_1, t_2, t_3, ..., t_n$ with the consecutive position $t_i, t_{i+1}, t_{i+2}, ..., t_{i+m-1}$ where $1 \le i \le n-m+1$.

In order to obtain subsequences, we extract the time series into *sliding windows*:

**Definition 3:** The *sliding windows* of length $L$ are all possible subsequences of the time series extracted by a window of length $L$.

When we have sliding windows, an overlapping pair of motifs may occur. This kind of overlapping pair is called a trivial match.

**Definition 4**: A pair of the subsequence of length $L$ is *trivial* when at least a single pair of positions is overlapped.

We give a discrete string T of alphabets for simplicity.

T = tkxyxyxyppllqpplqzas

To illustrate, if we consider sliding windows of length 4, the pair {xyxy, xyxy} is not preferable because it has overlapping part, i.e., it shares common {xy} which is considered *trivial*. In other words, the pair {ppll, ppll} is more interesting subsequence pattern, as it is not considered trivial.

**Definition 5:** A pair of the subsequence of length $L, \{X, Y\}$, is *too good* when the difference of the boundary deviates less than a standard deviation of a set of the difference in each position from position 1 to position L.

Here is the reason why we need to define the word "too good" above. Consider the following example:

TABLE I. AN EXAMPLE OF BITSTRING

| *T = psabcdefgxnmhbfabcdefgoposhijkbbohijkllmmz* | | |
|---|---|---|
| ***Subsequence*** | ***Length*** | ***Position*** |
| *abcdefg* | *7* | *a=3* |
| *abcdefg* | *7* | *a=16* |
| *hijk* | *4* | *h=27* |
| *hijk* | *4* | *h=34* |

For simplicity, we use an alphabet string to represent a time series sequence. Assuming that the algorithm runs from left to right for any length of sliding windows, there are two pairs of pattern here: *abcdefg* and *hijk*. In the case of sliding windows of length 4, there are two possible choices: *abcd* and *hijk*. Unfortunately, typical algorithms will discover only a pair of subsequence *abcd*. To resolve this, we need to make sure that *abcd* is a true motif for this sliding window. In this case, it is easy to see that by moving forward one data point from position (6, 19) to (7, 20) of time series *T* above. It is ('*e*', '*e*') which is the same. Therefore, it is considered too good (to be true) motif. The other sliding window lengths will eventually find this subsequence too. The sliding window of length 4 disregards this subsequence and retries the next pair. The next pair of the subsequences is *hijk* at the position (27, 34). In this case, by moving forward or backward one data point, none of the pairs of data, (26, 33) = ('*s*', '*o*') and (31, 38) = ('*b*', '*l*'), is the same. This means a pair of subsequence *hijk* is considered not *too good* at this window length.

**Definition 6**: The time series motif of length $L$ *(L-length-motif)* of a time series $T$ is an unordered pair of subsequences of length $L$ that has the smallest distance among sliding windows of length $L$, which is not considered *too good.*

This definition tells us that, in the window of length $L$, we disregard the smallest distance which is too good. This action allows the larger length of window to find the abandoned subsequence instead while this length still finds a proper pair of subsequences.

**Definition 7:** A *potential candidate* for a motif of the time series $T$ has the following properties,

*Similarity:* the candidate is the *L-length-motif.*
*Frequency:* the candidate has many neighbors that are similar to the candidate itself.
*Competitiveness:* the candidate can be a motif evenhandedly regardless of the length of the sliding windows.

Many readers may be familiar with a motif that is the most similar pair of subsequences. However, this work prefers a motif that is not only similar, but also frequent. These two properties seems conflicting to each other. Therefore, our proposed work manages this into two steps. First is to find the most similar pair of subsequence and not too good or *L-length-motif.* Here, the similarity has a major role. The "winner" or *L-length-motif* will be a representative of length $L$ to compete with other window sizes. The scoring function will base on both similarity and frequency. One of the most important properties of scoring function is *competitiveness.* This work achieves these three properties by using compression-based algorithm for the ranking scheme. While many compression techniques are available, our work utilizes an MDL framework [10] because it supports parameter free nature, but yet easy and quite intuitive. Nonetheless, the MDL framework requires discrete data to represent a bit for compression.

**Definition 8:** A discrete normalization function is a function to discretize the real value of time series into *a*-bit discrete values of range $[1, 2^a]$ as follows

$$DiscreteNorm(T) := round(\frac{T - \min}{\max - \min}) * (2^a - 1) + 1$$

where min and max are the minimum and the maximum value of subsequence *T*, respectively.

In this work, we simply use Euclidean distance as a distance function for similarity comparison.

**Definition 9:** The distance between two subsequences is defined as the Euclidean distance, given by

$$Dist(T_1, T_2) := \sqrt{\sum_{i=0}^{L} (t_{1,i} - t_{2,i})^2}$$

In information theory, *Shannon entropy* is widely used for defining the lossless compression technique. It quantifies the expectation value of the information. In other words, it represents the lower bound of expectation value (or average) of number of bits required to perform the lossless data compression on each data point [22]. In this context, we simply call it entropy.

**Definition 10:** The entropy of a time series $T$ is defined as:

$$Entropy(T) := \sum_i p_i \log_2(\frac{1}{p_i})$$

where $p_i$ is the probability that a symbol in $T_i$ will occur.

Notice that the time series is already discretized (Definition 8) before calculating the entropy. We have the expectation value of each data point. Now, we are ready to define the description length (DL) of the time series. See [10] for more information.

**Definition 11:** The description length DL of time series $T$ of length $m$ is the total number bits required to represent it, given as

$$DL(T) := m * Entropy(T)$$

**Definition 12**: A *hypothesis* $H$ is a subsequence used to encode one or more subsequences with the same length.

**Definition 13:** A Description Length of a Group (DLG) $C$ is the number of bits required to represent all subsequences in the group.

$$DLG(C) := DL(H) + \sum_{A \in C} DL(A - H) - \max_{A \in C}(DL(A - H))$$

The center of the group is hypothesis $H$ obtained by the average of all members in the group.

**Definition 14:** A *bitsave* is the number of bits saved after applying an operation. It can be calculated from the difference of the description length, before and after, i.e.,

$$bitsave := DL(old) - DL(new)$$

In this context, there are two operations: creating a group and adding a member to a group. They are defined as follows,

**Creating** a new group $C'$ from subsequence $A$ and $B$:

$$bitsave := DL(A) + DL(B) - DLG(C')$$

**Adding** a subsequence $A$ to an existing group $C$:

$$bitsave := DL(A) + DLG(C) - DLG(C')$$

**Definition 15:** The *potential motif windows* are ranked by the ability to compress the time series $T$. In other words, they are ranked by total bitsave of a group created by *potential candidate* in descending order.

## III. THE INTUITION BEHIND ALGORITHM

There are three main techniques here: lower bound, checking for a true motif, and the use of Minimum Description Length (MDL) [5][14]. The lower bounding techniques have been used extensively for Euclidean distance due to the utilization of the triangular inequality properties. The Euclidean distance is a part of the algorithm. The reason why we use Euclidean distance is that the calculation is fast, and yet it is a basis unit for distance measurement. First of all, the algorithm runs on every possible sliding window size. For each windows of size $L$, the algorithm searches for the most similar pair of subsequence of length $L$. This pair cannot yet be *L-length-motif* (Definition 6) because it needs to be checked for a true motif. Then, we check that this pair is a true motif. If it fails, we then look for the next pair and repeat the process. If a pair of subsequence passes the checking conditions, it will be used to create a group of motif by using the MDL technique. If the pair is similar, it has a significant amount of *bitsave*. Then, at the next step, we find the next neighbor by comparing the subsequence in sliding windows with the center of the group. At this stage, the Euclidean distance is used in conjunction with the lower bounding technique.

### A. Lower bound

A brute force algorithm to find a pair of motif has $O(n^2)$ complexity. However, it is possible to use a heuristic approach to prune off the unnecessary computation. The technique is known as *early abandoning* [2]. The key approach is to randomly select a sliding window as a pivot. Then, calculate distance of all sliding windows, and sort the other sliding windows by their distance to the pivot. This linear ordering of the data has useful heuristic information, as shown in Fig.2 Notice that the result of the sorted data has its own "lower bound" distance between adjacent pair. Therefore, the search will begin by using the *best-so-far* to enhance the speed. To illustrate, consider this example,

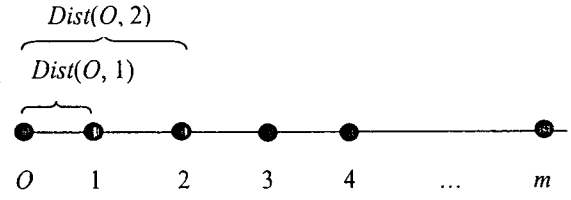



Figure 2. Linear ordering of the data. *O* is a pivot.

In the Fig.2, we assume $O$ is a pivot. The other subsequences in sliding windows are calculated by distance which used pivot as a reference. So, we have *Dist*(*O*, 1), *Dist*(*O*, 2), *Dist*(*O*, 3) ..., *Dist*(*O*, *m*). Without loss of generality, we focus on positions 1 and 2 as illustrated in the example. We can determine the lower bound distance between position 1 and position 2 by using the triangular inequality property as follows.

$$Dist(O,1) - Dist(O,2) \leq Dist(1,2)$$

By searching for the smallest distance, we are now ready to search and prune off the unnecessary parts, i.e., a variable used for recording the *best-so-far* distance. By scanning from left to right and broadening the offset, it is possible to have all $\frac{m(m-1)}{2}$ pairs in this search [2]. In each offset, we scan for the best-so-far distance and update the best-so-far value. If the best-so-far value is not updated in all the pairs at the same offset, we know that the rest of the unsearched pairs will not be able to update the best-so-far value as well. Therefore, we can disregard the unsearched pairs. This would save us enormous amount of computation time.

### B. *Checking for the true motif*

After a candidate motif is found, it might be a "too good" to be true motif. What does the word "too good" exactly mean? A possible situation is given by a time series T1 as in Fig.3. In case of small window length, the first motif found might be at data point 2, regardless of data point 1. In essence, we do not need data point 2 since other window lengths would find this data point 2 as well. Moreover, it makes much more sense if the smaller window lengths find the data point 1 as a true motif. It seems impossible to know whether it is good enough to be a true motif because the only indicator used in this context is a distance measure; sometimes the distance measure gives zero value without telling us much on whether or not it is a true motif. One possible approach is to broaden our perspective. The approach is divided into two phases: learning and detecting the deviation of the difference between particular points of a subsequence pair. For example, it is typically a pair of subsequence that has smallest distance. If we statistically collect the data of the difference value, we will have a set of data. The set of data can be used to make a model of normal distribution curve. The next step is to find difference at the outside of the boundary as in Fig.3. The key insight here is the use of standard deviation as an indicator. If the difference between the outside of the boundary deviates less than a standard deviation, it is most likely *too good* (Definition 5). In this manner, the other sizes of the window will find this pair of subsequences instead.

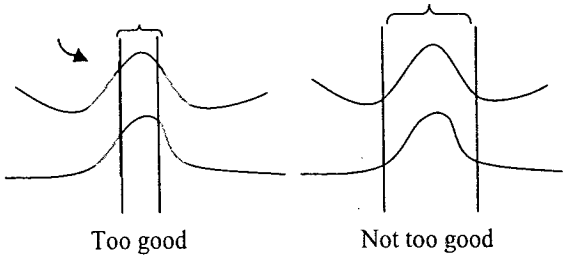



Figure 3. (left) The sliding window find this pair but it is too good because there is some room left on both outer boundaries as indicated with yellow areas. (right) This sliding window is fit to the length of the pattern, and thus considered as not too good (or true motif).

### C. *The use of Minimum Description Length (MDL)*

After a true motif of length $L$ is found, a group of two subsequences is ready to be constructed. We then find a nearest neighborhood in order to add to a group and calculate *bitsave* (Definition 14). The *bitsave* is an indicator of the fitness. The description length of a group specifies how many bits are required in representing a group of subsequences. Therefore, at first, we construct a group of two subsequences along with the calculation of *bitsave* after creating a group. In this manner, *bitsave* is merely the ability to compress subsequences as they are similar to each other. As in definition 13, the entire element of the group is subtracted from the center or hypothesis of the group in order to calculate *bitsave*. The algorithm runs until they can no longer be compressed any further or the neighbors are all used up. In short, Minimum Description Length is the lowest number of bits required to represent the time series sequence. In other words, the maximum number of *bitsave* (in this case, we consider as the fitness function) of that window size is what we are looking for.

## IV. OUR ALGORITHM IN DETAIL

**Algorithm** : MDL based motif discovery
**Input**: Timeseries Ts
**Output**: the *potiential motif windows* (Definition 15)

```
1    for w=2 to Ts.Length/2
2        do
3            bsf := RefreshPivot(ts,w)
4            (A,B) := NextPair(ts,bsf,w)
5        while CheckTrueMotif(A,B,w)= false
         //create new group
6        {G,bitsave}:= createGroup(A,B,w)
7        while HasNextNeighbor
8            C := NextNearestNeighbor( ts,center,w)
9            bs := AddToGroup( G , C,w);
10           if bs > 0 then bitsave := bitsave + bs
11           else break
12       end while
13   window[w] := bitsave
14   end for
15   sort(window)
```

Here, the algorithm is essentially parameter-free because the only input is a time series sequence. The output is *the potential motif windows* (see Definition 15 for more detail). The algorithm runs at all possible window lengths in this context, from 2 to Ts.Length/2. The core concept of this algorithm is the use of description length of a cluster of *L-length-motif* (see Definition 6) and their neighbors. As mentioned above, the smaller value of description length of the cluster means it would take fewer bits to represent the whole cluster. In other words, an element in the cluster is similar to each other. Therefore, the *bitsave* (Definition 14) objectively represents the fitness of the potential candidates (Definition 7) regardless of the length of the sliding windows. There are two main loops in the algorithm: finding the *L-length-motif*, then using it as the creator of the cluster and finding all neighbors of the *L-length-motif* and computing *bitsave* value. The algorithm terminates in two conditions: either the next neighbor cannot compress anymore or the neighbors are all used up.

The algorithm begins by entering a **for** loop (line 1). Notice that each *w* in the **for** loop is independent. Hence, the parallel computing technique can further improve the speed of this algorithm. In the **do while** loop (lines 2-5), *RefreshPivot* and *NextPair* together are the process of finding a motif in a sliding window of length *w*. These two functions could be grouped together. However, in practice, this loop is a hot spot because the RefreshPivot takes $O(nlogn)$ for sorting distances between every sliding window and a pivot. The motif extracted from these two functions must be inspected whether or not it is *too good* to be true by calling checkTrueMotif function. The checkTrueMotif function helps us find a fair pair of motifs that essentially belongs to the length of the sliding windows. The optimization could assuage the hot spot by finding the next pair until the *bsf* cannot be further updated, then RefreshPivot is called.

At line 6, *A* and *B* indicate the location of subsequence in time series. The function createGroup is called. The function creates a set $G = \{(\text{center}), \text{less}\}$ where center = $(A+B)/2$ and less is the minimum value of $\{A, B\}$.

At lines 7 – 12, in this stage, we have a group containing a pair of motifs. The NextNearestNeighbor takes a center of the group or cluster as a reference to find another neighbor *C*. Then, a neighbor *C* is added to the group. This loop continues until $bs \leq 0$ or there is no next neighbor left. Finally, every possible sliding window has its own total *bitsave* and stores it to a window (line 13). The windows are sorted to produce the *potential motif windows*.

## V. Experimental Results

We demonstrate the usefulness of minimum description length technique in our scoring scheme. Here are the extensive datasets in both planted datasets and real datasets. We use planted datasets for measuring accuracy and validity of the result because we know the predefined patterns and sizes. This section has three subsections, and is organized as follow: Gait Dynamic dataset, One-class dataset, and Multi-class dataset. The full datasets plotted in terms of bitsave and motif discovery in excel files are available at [20]. We also provide an open-source code in C++ language for reproducibility and for interested readers.

### *A. Gait Dynamic dataset*

Gait analysis is the systematic study of human motion. In particular, gait analysis is commonly used to assess, plan, and treat the individuals with conditions affecting their ability to walk. The raw data were collected by using force-senseitive resistors. (see Fig.4(a)). This experiment is set on the test on short data [3] (approximately 3,000 data points).The reason we select this dataset is to demonstrate the benefit of Minimum Description Length in ranking system as the scoring scheme. The human walk cycle is periodic as in Fig.4(a). The periodic nature of time series is difficult for many motif discovery algorithms because the proper length of the motif is not known in advance.

After running the algorithm, Fig.4 (b) shows the plot of total *bitsave* for each window size. Notice that the optimal or peak point is at position 279. This means the *potential motif window* (Definition 15) size is 279. The key point is that there are typically motifs of a specific length which repeatedly occur. So, they are the *potential candidate* (Definition 7). Fig.4 (a) annotates the discovered motifs. The very first pair of motifs found in this length is annotated in red. This pair is called a *potential candidate*. The group is created by a potential candidate and *bitsave* is calculated. As a next step, the potential candidate looks for neighbors as shown in colored highlight. These neighbors are added to an existing group and then the *bitsave* is calculated. The result shows that all patterns of length 279 are discovered.

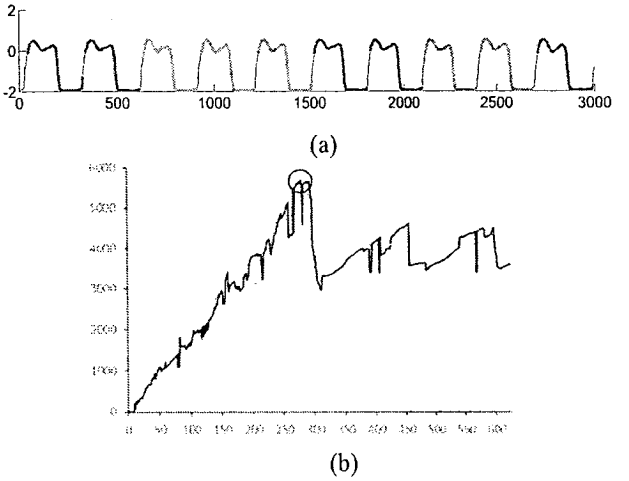



Figure 4. (a) A left foot signal dataset recorded at gait speed of 1.4 m/s. The result of motifs discovered at a window of length 279. The colored highlights.are all discovered motifs. (b) The typical plot of total bitsave at each window size. We omit the size of more than 500 for brevity. The x-axis represents length of sliding windows. The y-axis represents the total bitsave. The top peak is at window of length 279, as highlighted in red circle.

### *B. One-class dataset*

This dataset is planted with patterns of length 470 from the beef dataset [4]. The graph of bitsave for each sliding window is plotted. According to Fig.5(b), a peak of this plot occurs at the window length of 474 which is close to the planted size of 470. The highlights indicating various groups of motifs confirm correctness of the result, see Fig.5(a).

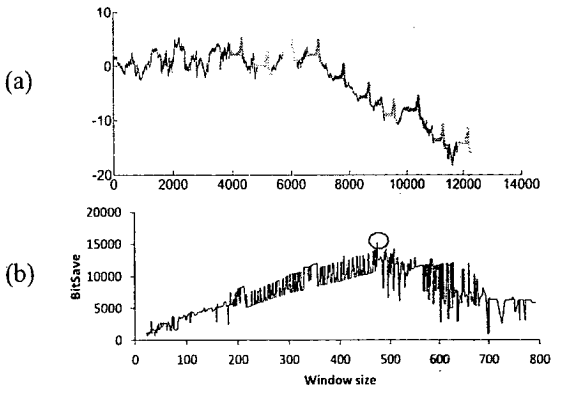



Figure 5. (a) A beef pattern that were planted in this dataset. The result of the motifs discovered at window of length 474. The colored highlights are all discovered motifs. (b) The typical plot of total bitsave at each window size. The top peak is at window of length 474 as highlighted in red circle.

### C. *Multi-class dataset*

This dataset is planted with two classes of patterns: The length of 150 data points from gun-point patterns and 275 data points for trace patterns (see Fig.6(a)) [4]. The graph of bitsave for each sliding window is plotted. According to Fig.6(b), two peaks occur at the window lengths of 151 and 276, which are very close to the planted patterns of sizes 150, and 275, respectively. The colored highlights indicating various groups of motifs confirm correctness of the result. Fig.6(c) shows the plot of motif discovered at length of 151. The second peak occurs at window of length 276, as shown in Fig.6(d). Our algorithm manages to find all of the motifs, as they are considered *potential candidates* (Definition 7).

## VI. SCALABILITY

Even though the worst case time complexity of the algorithm is $O(n^3)$, equaling that of brute force manner, we manage to optimize this complexity by the use of lower bound [2]. In essence, the complexity is reduced to be essentially quadratic. Moreover, the algorithm to check for a true motif is designed to be parallelizable. Specifically, each of the window length in the outer *for* loop is independent. This is, in fact, the model of SIMD (Single Instruction Multiple Data). The parallelism nature of the algorithm has a significant role in achieving a speedup. Lastly, the algorithm is compatible to the quantum computers due to the design that allows the quantum parallelism [17][18][19], and thus is worth exploring on the future work.

## VII. CONCLUSIONS

In this work, we introduce a parameter-free motif discovery algorithm that utilizes the minimum description length (MDL) technique. Our algorithm is specially designed to be further enhanced for speed by the use of parallel computing technique because each of the window lengths can be executed independently. The outcome of the *potential motif window* depends on both similarity and frequency of the motifs. This algorithm discovers the proper length of the sliding windows on various datasets including periodic datasets. We are also exploring the approach to other applications.

## ACKNOWLEDGMENT

We would like to thank all the donors of datasets. Also, we thank Dr. Vit Niennattrakul for his assistance and review of this work.

This research is partially supported by Returning Student Scholarship; the Thailand Research Fund (Grant No. MRG5380130), Chulalongkorn University Dutsadi Phiphat Scholarship; the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0319/2551 to S. Rodpongpun); and CU Graduate School Thesis Grant.

## REFERENCES

[1] A. Mueen, E. J. Keogh, and N. B. Shamlo, "Finding Time Series Motifs in Disk-Resident Data," ICDM, 2009, pp. 367-376.

[2] A Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact Discovery of Time Series Motifs," in Proceedings of the Ninth SIAM International Conference on Data Mining, 2009.

[3] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *http://www.physionet.org/physiobank/database/*

[4] E. Keogh, X. Xi, L. Wei, C. A. Ratanamahatana, "The UCR time series classification/clustering homepage," 2008, *http://www.cs.ucr.edu/~eamonn/time_series_data/*

[5] H. Li and N. Abe, "Clustering Words with the MDL Principle," Proc. of the 16th Int' Conf' on Computational Linguistics, 1996, pp. 5-9.

[6] H. Tang and S. S. Liao, "Discovering original motifs with different lengths from time series," Knowledge-Based Systems. vol. 21, pp. 666-671, 2008.

[7] J. Meng, J. Yuan, M. Hans, and Y. Wu, "Mining Motifs from Human Motion," in Proceedings of the 29th Annual Conference of the European Association for Computer Graphics Crete, Greece, 2008.

[8] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in Proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 53–68.

[9] P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana, "Parameter-free motif discovery for time series data," ECTI-CON, 2012, pp. 1 – 4.

[10] T. Rakthanmanon, E. Keogh, S. Lonardi, S. Evans, "Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data," ICDM, 2011, pp. 547-556.

[11] S. Rodpongpun, V. Niennattrakul, and C. A. Ratanamahatana, "Selective Subsequence Time Series clustering," Knowledge-Based Systems, 2012.

[12] Y. Tanaka, K. Iwamoto, and K. Uehara, K. "Discovery of time-series motif from multi-dimensional data based on MDL principle," Machine Learning, vol. 58, no. 2, 2005.

[13] Y. Li, J. Lin, "Approximate variable-length time series motif discovery using grammar inference," MDMKDD, 2010, pp. 1–9.

[14] B. Hu, T. Rakthanmanon,Y. Hao et al., "Discovering theIntrinsic Cardinality and Dimensionality of Time Series Using MDL," in Proceedingsof the 2011 IEEE 11th International Conference on Data Mining, 2011, pp.1086-1091.

[15] H. T. Lam, T. Calders, and N. Pham,"Online Discovery of Top-k Similar Motifs in Time Series Data," in SDM, 2011,pp. 1004-1015.

[16] H. Tang, and S. S. Liao, "Discoveringoriginal motifs with different lengths from time series," Know.-Based Syst., vol. 21, no. 7, pp. 666-671, 2008.

[17] B. Ömer. A procedural formalism for quantum computing. Master's thesis, Department of Theoretical Physics, Technical University of Vienna, 1998.

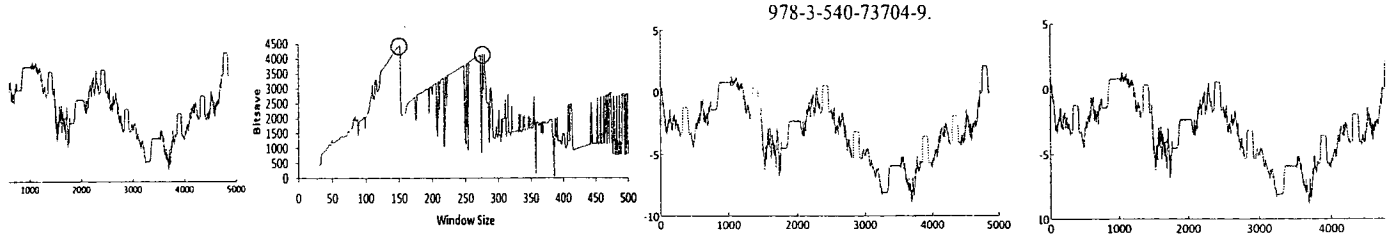
[18] B. Ömer. Quantum programming in QCL. Master's thesis, Institute of Information Systems, Technical University of Vienna, 2000.

[19] B. Ömer. Structured Quantum Programming. PhD thesis, Technical University of Vienna, 2003.

[20] Support webpage, *https://sites.google.com/site/dmdhlab/plmda*

[21] Y. Lin, M. D. McCoo, A. A. Ghorbani, "Motif and Anomaly Discovery of Time Series," IMECS, 2010, pp. 481-486

[22] Goise, Francois & Olla, Stefano (2008). Entropy methods for the Boltzmann equation: lectures from a special semester at the Centre Émile Borel, Institut H. Poincaré, Paris, 2001. Springer. p. 14. ISBN 978-3-540-73704-9.




(a) (b) (c) (d)

Figure 7. (a) The gun and trace pattern that were planted in this dataset. (b) The typical plot of total bitsave at each window size. The x-axis represents length of sliding windows. The y-axis represents the total bitsave. (c) The result of motifs discovered at window of length 151. (d) The result of motifs discovered at window of length 276.

# Multiple Shape-based Template Matching for Time Series Data

Warissara Meesrikamolkul, Vit Niennattrakul and Chotirat Ann Ratanamahatana
Department of Computer Engineering, Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok, Thailand 10330
{g53wms, g49vnn, ann}@cp.eng.chula.ac.th

*Abstract*—**1-Nearest Neighbor classification with Dynamic Time Warping distance measure is mainly used for time series classification. In large datasets, major concerns for the classification problem are CPU time and storage requirement. Recently, Shape-based Template Matching Framework (STMF) was proposed to resolve these problems by constructing a template as a representative for each class of the data, and then STMF uses these templates to classify a query sequence. However, a single template per class may not well represent the overall characteristic of the data. In this paper, we propose a new method called Multiple Shape-based Template Matching (MSTM) extended from STMF. Our method constructs multiple templates by clustering each class of data and also learning the global constraint to increase the accuracy. In the experiment, we evaluate by comparing with STMF which uses only one template per class and the original 1-NN classification with global constraint. Our proposed method also minimizes the number of templates and still classifies the query sequence effectively.**

*Keywords-Shape Averaging; Template Matching; Time Series Data Classification;*

## I. INTRODUCTION

Time series classification [3] is one of the major time series mining tasks and has been applied to a variety of domains and applications, especially when a wide range of real world data can be meaningfully transformed into time series such as biometric [9] and multimedia data [10]. For time series classification, Dynamic Time Warping (DTW) distance [1] measure, a well-known similarity measure for time series data, has been proven to be a suitable choice since it allows more flexibility in sequence alignment than the Euclidian distance metric.

1-Nearest Neighbor (1-NN) classification based on DTW distance is considered one of the most accurate techniques to classify time series data. Unfortunately, in large datasets, this classic 1-NN may not be practical since it needs to compute DTW distance between the query sequence and every single candidate sequence in the entire dataset, making it very computationally expensive. Although recent research works have tried to speed this up using many techniques such as lower bounding [4] and indexing [7], they are still considered impractical for applications with limited storage.

Template matching is an approach that can solve both computational complexity and storage problems; one template for each class can be used as its representative, thus reducing the size of the training set down to just a number of classes to produce its template is a sensible approach. Shape averaging must be used instead of amplitude scaling to be able to capture the actual features of all the data. Prioritized Shape Averaging (PSA) algorithm [6] has been proposed to produce one template for each class using shape averaging scheme. Fig. 1 demonstrates the idea of the template approach. In particular, each pair of sequences is averaged in an order according to a hierarchical clustering result. However, since PSA algorithm uses uniform scaling to reduce the length of the averaged sequence, some errors increase in the resulting template. Then Shape–based Template Matching Framework (STMF) [8] was proposed in an attempt to reduce this error by averaging most similar pair of the sequences using Cubic-spline Dynamic Time Warping (CDTW). CDTW uses cubic spline interpolation [8] to re-sample the averaged sequence instead of the uniform scaling. Even though the cubic spline is a better approach, the accuracy of 1-NN classification using this template matching is much lower comparing to the classification accuracy when the entire training dataset is used since one single template may not be able to represent the overall characteristic of data within the class.

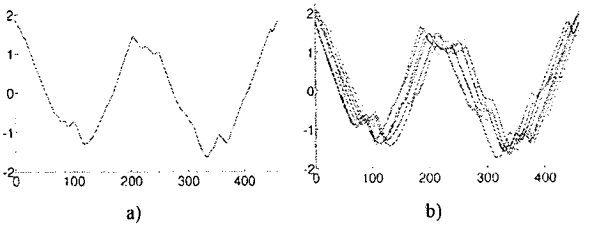



Figure 1. a) The template obtained from a sample of data b) from Fish dataset.

In this work, we extend the idea of CDTW further and introduce a novel framework called Multiple Shape-based Template Matching (MSTM). Specifically, we cluster each class separately using hierarchical clustering, validate the cluster quality with Silhouette index [12], and create multiple templates for each class. We will demonstrate that our multiple templates can better represent the data characteristics especially when subclasses are present. We also utilize a global constraint [11] that is the Sakoe-Chiba band [13] which is learned to find an appropriate warping window size for DTW distance measure, which can boost up the accuracy of classification. Our experiments will compare our method with STMF/CDTW technique in 1-NN time series classification problem. The result shows that our method can improve the accuracy effectively while greatly reduce the size of the training data.

## II. BACKGROUND

### A. Dynamic Time Warping (DTW) distance measure

DTW distance measure [1] is the similarity measurement which is typically used for time series data. DTW distance can discover an optimal alignment between two sequences using the following equation.

$$dist(q_i, c_j) = dist(q_i, c_j) + min \begin{cases} dist(q_{i-1}, c_j) \\ dist(q_i, c_{j-1}) \\ dist(q_{i-1}, c_{j-1}) \end{cases} \quad (1)$$

Given two time series $Q$ and $C$ where $q_i$ and $c_j$ are their elements, respectively. DTW can discover a minimum cumulative distance between $Q$ and $C$ using dynamic programming. As shown in Fig. 2, the warping path can be found by tracing back from the final cell's alignment (top right) down to the first (bottom left) element after the minimum cumulative distance calculation is completed.

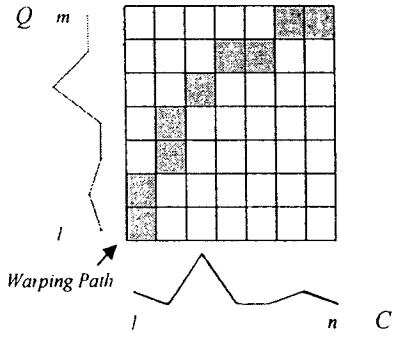



Figure 2. The warping path between two time series sequences.

### B. Hierarchical Clustering

Hierarchical Clustering [2] is a method to group similar objects using DTW distance as a similarity measurement and using the maximum distance between objects of each cluster to merge similar clusters. In this work, complete linkage inter-cluster distance is used. Merging is made in a bottom-up fashion until a desired number of clusters is reached.

### C. Silhouette Index

Silhouette index [12] is a well-known technique to validate the quality of clustering using the distance among data object within the same cluster and across difference clusters.

$$s(i) = \frac{b(i) - a(i)}{max\{b(i), a(i)\}} \quad (2)$$

$$S = \frac{1}{n} \sum_{i=1}^{n} s(i) \quad (3)$$

In equation (2), $s(i)$ is the Silhouette index of time series data object $i$, where $a(i)$ is an average value of all distances between instance $i$ and every time series object within the same class, and $b(i)$ is a minimum average value of distances between object $i$ and every time series object in different classes. The Silhouette index, $S$, of a dataset can be computed according to Equation (3), where $n$ is number of time series instances in the dataset and the maximum index gives the best clustering separation.

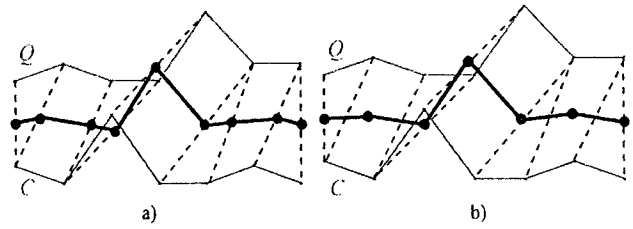



Figure 3. The averaged sequence a) before re-sampling and b) after re-sampling using cubic spline interpolation.

### D. Cubic-Spline Dynamic Time Warping (CDTW)

CDTW [8] has been introduced to average two time series sequences with an additional feature of maintaining their original shape using cubic spline interpolation. Suppose we are to average the sequences $Q$ and $C$. Firstly, we find the warping path ($W$) between these two sequences. Each $w_k$ element consists of the value $i$ and $j$ which are the indices of the element $q_i$ and $c_j$, respectively. The average sequence $Z(z_k(x), z_k(y))$ is calculated as follows.

$$z_k(x) = \frac{\omega_q q_i + \omega_c c_j}{\omega_q + \omega_c} \quad (4)$$

$$z_k(y) = \frac{\omega_q q_{i_k} + \omega_c c_{j_k}}{\omega_q + \omega_c} \quad (5)$$

In equations (4) and (5), $\omega_q$ and $\omega_c$ are weights of the sequences $Q$ and $C$. Since the resulting sequence will be longer than or at least have the same length as the original ones (as shown in Fig.3 (a)), the cubic spline interpolation is used to re-sample the number of points of an averaged sequence (as shown in Fig.3 (b)).

### E. Global Constraint

For some applications such as speech recognition [13], the global constraint is typically used to limit the warping window of sequence alignment to increase the accuracy of the classification. Additionally, the global constraint with the help of lower bounding can help reduce both CPU and I/O times [4]. Sakoe-Chiba band [13], one of the well-known global constraints, was introduced by Sakoe and Chiba in speech recognition community, and has been practically used in various time series mining problems [11].

Suppose there are two time series sequences to be averaged using Sakoe-Chiba band. The sequences simply are allowed to align only within the limited window (as shown in Fig.4). The width of the warping window is defined by $r$ which is a percentage of the time series' length, specifying how much warping is allowed above and to the right of the diagonal cells.

## III. MULTIPLE SHAPE-BASED TEMPLATE MATCHING

STMF [8] constructs one single template from every time series sequence within the same class as its representative. However, this single template may not be able to preserve all of the characteristics of every time series sequences within that

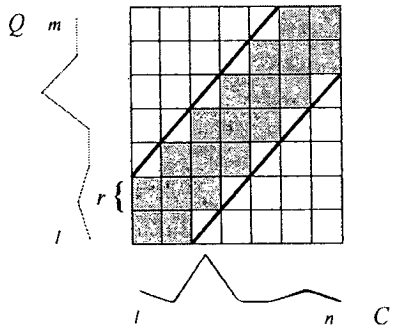



Figure 4. The Sakoe-Chiba band, global constraint to limit the warping path.

class. In this work, we propose a novel framework called Multiple Shape-based Template Matching (MSTM) to increase the number of templates for each class by clustering the data within each class into subclasses and constructing one template for each subclass. We will later demonstrate that these templates better represent the characteristics of overall data within the class.

TABLE I. AN ALGORITHM TO CLUSTER EACH CLASS OF DATA

```
Algorithm Clustering(D_C)
1.  D_C  is the set of time series data of class C
2.  Num_C is the number of time series in class C
3.  num  is the number of subclasses in class C
3.  for n = 1:Num_C
4.     HierarchicalClustering(D_C)to n subclasses
5.     S = SilhouetteIndex(D_C)
6.     if S_n is less than S_n-1
7.        num = n-1
8.        break
9.     end if
10. end for
11. return num
```

Table I shows our clustering algorithm which separates each class into subclasses using hierarchical clustering and validates clusters with the Silhouette index. After we get the number of subclasses, we construct one template for each subclass. The algorithm is shown in Table II.

We construct templates from every subclass following an approach proposed in [8]. We continually average the most similar pair of data sequences until we get one final template. Once completed, we obtain multiple templates for the training dataset which are used to classify the query sequences.

TABLE II. AN ALGORITHM TO CONSTRUCT TEMPLATES WITH GLOBAL CONSTRAINT

```
Algorithm Templates(D_T)
1.  D_T is the set of training data
2.  D_C is the set of data in class C
3.  N_C is the number of classes in training data
4.  D_S is the set of data of subclass S
5.  T  is the set of templates
6.  N_S is the number of subclasses in class C
7.  r  is the size of warping window
8.  N_S = Clustering(D_C)
9.  for i = 1:N_C
10.    for j = 1: N_S
11.       T = STMF(D_Sij)
12.    end for
13. end for
14. r = LearningConstraint(T)
15. return T, r
```

Additionally, we utilize the global constraint (in line 14.) to increase the classification accuracy. The best warping window ($r$) will be returned as a result from template learning, as shown in Table III.

TABLE III. AN ALGORITHM TO LEARN GLOBAL CONSTRAINT

```
Algorithm LearningConstraint(T)
1.  T  is the set of templates
2.  D_T is the set of training data
3.  r  is the size of warping window
4.  acc = accuracy of 1-NNClassification
5.  for i = 1:100
6.     acc = 1NNClassification(D_T, T, i)
7.     if acc_i is more than acc_i-1
8.        r = i
9.     end if
10. end for
11. return r
```

## IV. EXPERIMENTS AND RESULTS

We evaluate our proposed method, MSTM with global constraint, by comparing the accuracy with STMF, which uses single template and the full 1-NN classification. The experiment uses ten datasets from the UCR datasets classification/clustering archive [5] as shown in Table IV.

In the experiment, we construct templates from the training data and evaluate our templates with the test data. Table V compares the 1-NN classification accuracies of STMF (one template per class), our MSTM (multiple templates per class), and the full 1-NN classification with no templates. The best warping path sizes ($r$) are also reported. Note that warping window constraint learning is applied in the full 1-NN to give their optimal accuracies. In most datasets, our MSTM gives better classification accuracy than the STMF, and also gives comparable accuracy to the full 1-NN classification.

In Table VI, we report the number of templates for each datasets, along with the average number of templates per class, which is very few compared with the size of the training dataset; our templates can classify the query sequence practically and also require much less storage. Fig. 5 shows two templates obtained from one of the classes of Fish dataset.

TABLE IV. THE DETAILS OF DATASETS

| Datasets | Number of classes | Length of data | Size of training set | Size of test set |
|---|---|---|---|---|
| Synthetic Control | 6 | 60 | 300 | 300 |
| Trace | 4 | 275 | 100 | 100 |
| Gunpoint | 2 | 150 | 50 | 150 |
| Lightning-2 | 2 | 637 | 60 | 61 |
| Lightning-7 | 7 | 319 | 70 | 73 |
| ECG | 2 | 96 | 100 | 100 |
| Olive Oil | 4 | 570 | 30 | 30 |
| Fish | 7 | 463 | 175 | 175 |
| CBF | 3 | 128 | 30 | 900 |
| Swedish leaf | 15 | 128 | 500 | 625 |
| Two Patterns | 4 | 128 | 1000 | 4000 |

TABLE V. THE ACCURACIES COMPARED WITH RIVAL METHODS

| Datasets | 1-NN Classification Accuracy | | |
|---|---|---|---|
| | STMF | MSTM with Learning Constraint ($r$) | 1-NN with Learning Constraint ($r$) |
| Synthetic Control | **0.97** | 0.95(7) | 0.98(6) |
| Trace | 0.98 | **1.00(13)** | 0.99(3) |
| Gunpoint | 0.64 | **0.90(2)** | 0.91(0) |
| Lightning-2 | 0.56 | **0.79(25)** | 0.87(6) |
| Lightning-7 | 0.66 | **0.71(17)** | 0.71(5) |
| ECG | **0.70** | 0.69(11) | 0.88(0) |
| Olive Oil | **0.80** | **0.80(1)** | 0.87(1) |
| Fish | 0.58 | **0.70(3)** | 0.84(4) |
| CBF | 0.96 | **0.98(25)** | 1.00(6) |
| Swedish leaf | 0.70 | **0.76(4)** | 0.83(2) |
| Two Patterns | **0.97** | 0.95(43) | 1.00(4) |

TABLE VI. THE NUMBER OF TEMPLATES AND DATA REDUCTION

| Datasets | Size of training sets (number of classes) | Number of templates | Average templates per class | Reducing size of training sets (%) |
|---|---|---|---|---|
| Synthetic Control | 300(6) | 12 | 2 | 96.00 |
| Trace | 100(4) | 8 | 2 | 92.00 |
| Gunpoint | 50(2) | 4 | 2 | 92.00 |
| Lightning-2 | 60(2) | 4 | 2 | 93.33 |
| Lightning-7 | 70(7) | 15 | 2.14 | 78.57 |
| ECG | 100(2) | 4 | 2 | 96.00 |
| Olive Oil | 30(4) | 13 | 3.25 | 56.67 |
| Fish | 175(7) | 16 | 2.29 | 90.86 |
| CBF | 30(3) | 12 | 4 | 60.00 |
| Swedish leaf | 500(15) | 33 | 2.2 | 93.40 |
| Two Patterns | 1000(4) | 8 | 2 | 99.80 |

In most datasets, our MSTM with global constraint is more accurate than STMF because multiple templates can preserve the characteristics of data better than a single template. Moreover, our method can even give better accuracy than the full 1-NN classification in some datasets because the templates can decrease noise in the data. However, the accuracy of our method is slightly less than that of STMF in some datasets since the nature of data is too noisy; therefore, the noisy data can be clustered into subclasses. We would like to point out that even though multiple templates are used, our computational time still is quite comparable to that of the single template because global constraint is used.

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel framework called Multiple Shape-based Template Matching (MSTM) with global constraint. Our method clusters each class into subclasses and utilizes shape-based averaging to construct multiple templates as representatives for each class. We also use the global constraint to improve the 1-NN classification accuracy. Our multiple-template method in general yields higher accuracy compared with the single- template approach

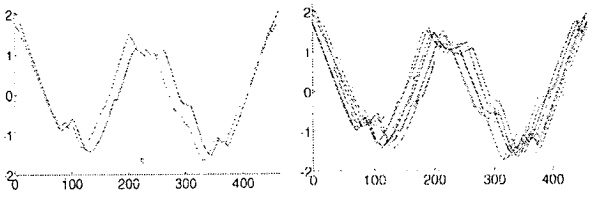



Figure 5. a) The multiple templates obtained from a sample of data b) from Fish dataset.

and can reduce size of the training data by a large margin.

For future work, if we can improve shape-based averaging algorithm to provide a real averaged sequence, it will be useful for many other applications.

## ACKNOWLEDGMENT

This research is partially supported by the Thailand Research Fund (Grant No. MRG5380130), the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0141/2549 to V. Niennattrakul).

## REFERENCES

[1] D. Berndt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," *In Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1994, pp. 359-370.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed, Morgan Kaufman, 2006, pp. 408-416.

[3] E. Keogh and M. J. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," *In Proceedings of 4th International Conference of Knowledge Discovery and Data Mining*, New York, USA, 1998, pp. 239–243.

[4] E. Keogh and C. A. Ratanamahatana, "Exact Indexing of Dynamic Time Warping," *Knowledge and Information Systems*, vol. 7, 2005, pp. 358–386.

[5] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "The UCR Time Series Classification/Clustering Homepage," 2008; www.cs.ucr.edu/~eamonn/time_series_data/.

[6] V. Niennattrakul and C. A. Ratanamahatana, "Shape Averaging under Time Warping," *In 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI-CON 2009)*, Pattaya, Thailand, 2009.

[7] V. Niennattrakul, P. Ruengronghirunya and C. A. Ratanamahatana, "Exact Indexing for Massive Time Series Databases Under Time Warping Distance," *Data Mining and Knowledge Discovery*, vol 21, November 2010, pp. 509-541.

[8] V. Niennattrakul, D. Srisai, and C. A. Ratanamahatana, "Shape-based Template Matching for Time Series Data," under review.

[9] V. Niennattrakul, D. Wanichsan, and C. A. Ratanamahatana, "Hand Geometry Verification Using Time Series Representation," *In Proceedings of Knowledge-Based Intelligent Information and Engineering Systems Lecture Notes in Computer Science*, 2007, pp. 824-831.

[10] C. A. Ratanamahatana and E. Keogh, "Multimedia retrieval using time series representation and relevance feedback," *In Proceedings of 8th International Conference on Asian Digital Libraries (ICADL'2005)*, Bangkok, Thailand, 2005, pp. 400-405.

[11] C. A. Ratanamahatana and E. Keogh, "Making Time-Series Classification More Accurate Using Learned Constraints," *In Proceedings of SIAM International Conference on Data Mining (SDM'04)*, Lake Buena Vista, FL, USA, 2004, pp. 11–22.

[12] P. J. Rousseeuw, "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Journal of Computational and Applied Mathematics*, vol. 20, 1987, pp. 53-65.

[13] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, 1978, pp. 43–49.

# Discovery of Variable Length Time Series Motif

Pawan Nunthanid, Vit Niennattrakul, and Chotirat Ann Ratanamahatana
Department of Computer Engineering, Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok Thailand 10330
{g53pnn, g49vnn, ann}@cp.eng.chula.ac.th

*Abstract*— **One significant task in time series mining research area is motif discovery which is the first step needed to be done in finding interesting patterns in time series sequence. Recently, many motif discovery algorithms have been proposed in place of the untenable brute-force algorithm, to improve its time complexity. However, those motif discovery algorithms still need a predefined sliding window length that must be known a priori. In this paper, we present a novel motif discovery algorithm that requires no window length parameter. This sliding window length is sensitive in that a small difference in the value can lead to huge difference of motif results. The proposed algorithm automatically returns suitable motif lengths from all possible sliding window lengths; in other words, our algorithm efficiently reduces a large set of possibilities of the sliding window lengths down to a few truly-interesting variable-length motifs.**

*Keywords-Motif discovery; Time series; Variable length*

## I. INTRODUCTION

Research area in time series mining [12][17][15][13][3] is increasingly important for a wide variety of applications including prevalent electrocardiogram in medical field [5] and stock market prediction [7]. Motif discovery [12] which focuses on finding patterns occurred repeatedly, is among the most common time series mining tasks. Time series motif is a pair of the most similar subsequences in time series. An example illustrated in Figure 1 is a motif in gun-point dataset [6]; a motif of length 150 data points is discovered at locations 450 to 599 and 1,647 to 1,796, respectively. A straightforward approach is a "brute-force" method that applies a sliding window of a predefined length of 150 to search for the best matched subsequences. However, the problem here is twofold; the length of sliding window must be known or chosen a priori, and the brute-force approach is simply untenable in practice. To the best of our knowledge, almost all of the works on time series motif discovery are focusing on the latter problem of increasing efficiency especially its time complexity or speed of motif discovery algorithm to replace the brute-force approach, but almost none has mentioned about the choice of the sliding window length. In the past decade, many motif discovery algorithms have been proposed. In the beginning, there were several fast approximate algorithms [1][2][4][10][11][16][18] to discover motifs faster, with some tradeoffs of increased error rates. Recently, the latest algorithm [12], Mueen-Keogh (MK), has been proposed; it is the first and only exact algorithm which uses triangular inequality to prune off unnecessary calculations. It is however still based on a fixed length motif scheme; so, the problem still remains.

We would like to demonstrate how important the variable length problem is and see what would happen if the length of sliding window is varied. It is quite a challenging problem; other than adding more computational complexity to the problem itself, its exact definition of an optimal sliding window size is still unclear. In typical time series motif discovery problem, the length of the sliding window must be predefined, usually by domain experts, by trial and errors, or simply by inspection of the time series plot. We will show that the choice of this parameter is in fact crucial and quite sensitive, and thus should not be neglected; only a small difference in the window length can cause a big change in the discovered motifs as illustrated in Figure 2.

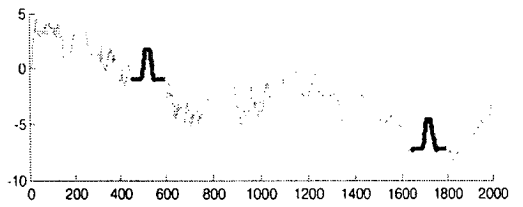



Figure 1. A motif in gun-point dataset [6] with 2,000 data points. The motif of length 150 are located at data points 450 to 599 and 1,647 to 1,796.

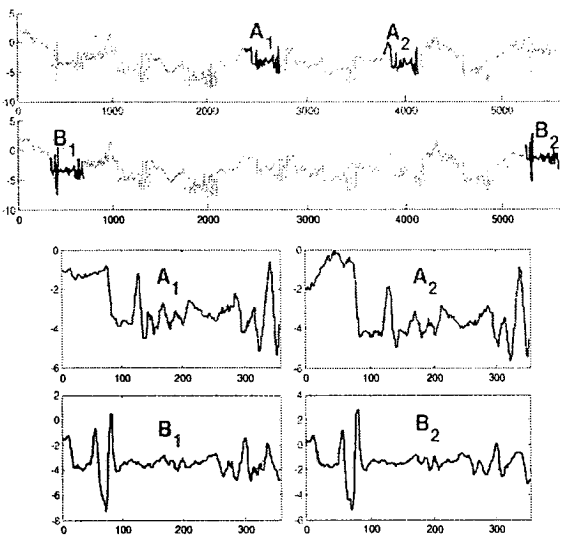



Figure 2. Two completely different motifs of lengths 352 and 353 data points discovered from the face-four dataset of 5,600 data points in length. The first motif locations $A_1$ and $A_2$ are at 2,376 to 2,727 and 3,777 to 4,128 while the second motif locations $B_1$ and $B_2$ are at 341 to 693 and 5,239 to 5,591.

In this work, Variable-Length Motif Discovery (VLMD) algorithm is proposed to automatically find a set of suitable variable-length motifs. Specifically, our VLMD algorithm iteratively separates different-length motifs into groups based on sensitivity. Within a group, a motif representative with minimum normalized distance between subsequence pairs is selected. Finally, VLMD returns a set of useful motif representatives which is extremely small comparing with all possibilities of sliding window lengths.

The rest of the paper is organized as follows. Related work, definitions, and background are provided in Section II. Our proposed algorithm is described in Section III, and in Section IV, our proposed method is evaluated. Finally, Section V concludes our work.

## II. Related Work, Definitions, and Background

### A. Related Work

To the best of our knowledge, two algorithms for time series variable length discovery are *k*-motif-based algorithm [19] and grammar-inference-based algorithm [8]. *K*-motif-based algorithm utilizes Motif Concatenation which concatenates motifs into one segment. This process analyzes a collision matrix and puts the matrix as an input for the algorithm before using three predefined parameters to constrain a motif searching range for concatenation. Unlike *k*-motif-based algorithm, grammar-inference-based algorithm uses the idea of grammar induction to find approximate motif length. In addition, grammar-inference-based algorithm is designed specifically for discrete data; therefore, it uses Symbolic Aggregate Approximation (SAX) [9] to first discretize time series into strings before applying grammar induction on the strings, and then converts the strings back to time series subsequence. The grammar-inference-based algorithm needs the initial variable length, the numbers of SAX segments, and alphabet size. However, both works require many additional parameters, especially an initial sliding window length, where in this work, no parameter is required.

### B. Definitions

All definitions we use in this work will be stated in this section.

**Definition 1**: Time series $T$ is a sequence of real values continuous in series, defined as $T = \{t_1, t_2 \ldots, t_n$

**Definition 2**: Subsequence $S_i^w$ is a smaller sequence of time series $T$, where $i$ is its starting point and $w$ is a sliding window length, defined as $S_i^w = \{t_{i\ \ +1}, \ldots, t_{i\ \ w-1}\}$, where $w > 1$.

**Definition 3**: Euclidean distance $\mathrm{EUC}(S_i^w, S_j^w)$ is a distance between two subsequences $S_i^w$ and $S_j^w$, where $w$ is a sliding window length, $1 \le i \le n$, and $1 \le j \le n$, defined as $\mathrm{EUC}(S_i^w, S_j^w) = \sqrt{\sum_{k=1}^{w} (t_{k\ \ i-1} - t_{k+j-1})^2}$.

**Definition 4**: Time series motif $M_w$ is a pair of the most similar subsequence in time series $T$ with specified sliding window length $w$. It is defined as $M_w = (L_1\ L_2$ , where $L_1$ and $L_2$ are the starting locations of subsequences in time series $T$, $L_1 < L_2$, and *MDist* is the normalized Euclidean distance between $S_{L_1}^w$ and $S_{L_2}^w$ calculated by $MDist = \mathrm{EUC}(S_{L_1}^w, S_{L_2}^w)/w$.

**Definition 5**: Two motifs $M_i$ and $M_j$ are overlapped when ($M_i.L_1 \le M_j.L_1 < M_i.L_1\ \ i$ or $M_jL_1 \le M_iL_1 < M_jL_1+j$) and ($M_i.L_2 \le M_j.L_2 < M_i.L_2+i$ or $M_jL_2 \le M_iL_2 < M_jL_2+j$).

**Definition 6**: Motif group *MG* is a group of motifs, where two contiguous motifs are overlapped, and is defined as $MG = \{M \mid M_i$ and $M_j$ are overlapped, $1 \le i < j \le n/2\}$.

**Definition 7**: Motif group representative $R$ is a time series motif which has minimum *MDist* among all motifs in a given motif group *MG*.

**Definition 8**: Overlapping length of two motifs $M_i$ and $M_j$ is an average value between percentage of overlapping sequence between $S_{L_1}^l$ and $S_{L_1}^l$ and percentage of overlapping sequence between $S_{L_2}^l$ and $S_{L_2}^l$.

**Definition 9**: Sensitivity of a motif $M_w$ is an averaged value of an overlapping length between $M_w$ and $M_{w-1}$ and an overlapping length between $M_w$ and $M_{w+1}$. This sensitivity ranges from zero to one, where the value is close to one is very sensitive.

### C. Background

Motif discovery is used to retrieve hidden relevant information. In addition, motif discovery is needed for other time series mining algorithms such as subsequence matching [14] and discord detection [3]. Intuitively, a motif discovery returns the most similar subsequences of a long time series sequence given a fixed-length sliding window, where Euclidean distance is typically used as a distance measure. In this work, we utilized MK Motif Discovery algorithm which is the fastest algorithm to discover the motif to date. However, other motif algorithms can simply be utilized as well. Due to space limitation, details on MK algorithm can be found in [12] for interested readers.

## III. Variable-Length Motif Discovery

In this work, we propose a novel parameter-free motif discovery algorithm called Variable-Length Motif Discovery (VLMD) that returns a small set of variable-length motifs. Given a time series sequence $T$ as an input, VLMD returns a set of motif results as an output, where the number of motif results is significantly smaller than the number of all possible motif lengths.

Specifically, VLMD consists of two steps. Firstly, VLMD finds a set of motif groups by searching all possible lengths of sliding window to get motifs with different lengths. If the current motif and the previous motif overlap, the current motif is added to the same motif group of the previous motif; otherwise, a new motif group is created. Then, for each motif group, a motif representative is selected from the motifs with minimum normalized motif distance.

Given a time series sequence $T$ of length $n$, the maximum possible motif length is $n / 2$ since the motif is defined as a pair of non-overlapping subsequences. In addition, the minimum possible motif length is 2 since from the definition, the sliding

window length must be larger than 1. VLMD iteratively finds a motif with length from 2 to $n / 2$. In each iteration, after a motif is discovered by an exact motif discovery algorithm, if the motif with a sliding window length $w$ overlaps with the previous motif of a sliding window length $w - 1$, the motifs $M_w$ and $M_{w-1}$ are assigned the same motif group $MG$; otherwise, a new motif group $MG$ is created by adding $M_w$ as the first member. In particular, $M_w$ and $M_{w-1}$ are considered overlapping if subsequence $S_{L_1}^{w}$ of $M_w$ overlaps with subsequence $S_{L_1}^{w-1}$ of $M_{w-1}$ and subsequence $S_{L_2}^{w}$ of $M_w$ overlaps with subsequence $S_{L_2}^{w-1}$ of $M_{w-1}$. All motif groups are stored in a motif group set $MGS$.

After all motifs generated from all possible sliding window lengths are retrieved, the motif representative $R$ is selected for each motif group, and is stored in a set of motif results $MRS$. The motif representative $R$ of a motif group $MG$ is a motif $M_w$ that has minimum $MDist$ in the group $MG$. The number of motif representative $R$ will be very small comparing to the number of all possible sliding window lengths from $w = 2$ to $n / 2$. The pseudo code of VLMD is provided in Table I.

TABLE I. VARIABLE LENGTH MOTIF DISCOVERY ALGORITHM

| | [*MRS*] VARIABLELENGTHMOTIFDISCOVERY (*T*) |
|---|---|
| 1. | Let $n$ be the length of time series $T$ |
| 2. | Let *MGS* be a set of motif groups |
| 3. | Let *MRS* be a set of motif results |
| 4. | Let $i$ be a group count |
| 5. | $i = 0$ |
| 6. | For ($w = 2$ to $n / 2$) |
| 7. | $M_w$ = MK_MOTIF_DISCOVERY($T$, $w$) |
| 8. | If ($M_w$ overlaps with $M_{w-1}$) |
| 9. | $MG_i$ <- $M_w$ |
| 10. | Else |
| 11. | $i$++ |
| 12. | End If |
| 13. | End For |
| 14. | For each *MG* in *MGS* |
| 15. | $R = M_w$ with minimum *MDist*, $M_w \in MG$ |
| 16. | *MRS* <- $R$ |
| 17. | End For |
| 18. | Return *MRS* |

## IV. EXPERIMENT

We use gun-point dataset and face-four dataset from the UCR Clustering/Classification archive [6] in our experiments. First, we interleave each of the 150 gun-point patterns with 150 data points in length into a random-walk data of 22,500 data points in length, giving a time series of 45,000 data points. Then, we further divide it into 15 sets of data, each with 3,000 data points. Second, 88 Face-Four patterns of 350 data points in length are similarly interleaved into a random-walk data of 30,800 data points in length, giving a time series of 61,600 data points. Then, we further divide it into 11 sets of data, each with 5,600 data points. Examples of both datasets are shown in Figure 3.

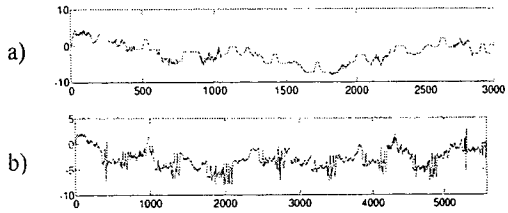



Figure 3. Examples of a) Gun dataset and b) Face Four dataset.

Our VLMD is mainly proposed to eliminate a sliding window length selection in motif discovery problem. As a result, it returns a very small set of motif representatives. Table II shows that VLMD can significantly reduce as much as 99.13% in a set of answers for the Gun dataset and as much as 99.29% for the Face Four dataset.

TABLE II. VLMD CAN SIGNIFICANTLY REDUCE A SET OF POSSIBLE ANSWERS

| Dataset | Gun (total $w$ = 1499) | | Face Four (total $w$ = 2799) | |
|---|---|---|---|---|
| | Number of motif representatives | Reduced percentage | Number of motif representatives | Reduced percentage |
| 1 | 23 | 98.47% | 27 | 99.04% |
| 2 | 25 | 98.33% | 19 | 99.32% |
| 3 | 26 | 98.27% | 25 | 99.11% |
| 4 | 19 | 98.73% | 32 | 98.86% |
| 5 | 23 | 98.47% | 32 | 98.86% |
| 6 | 13 | 99.13% | 29 | 98.96% |
| 7 | 21 | 98.60% | 40 | 98.57% |
| 8 | 17 | 98.87% | 25 | 99.11% |
| 9 | 26 | 98.27% | 22 | 99.21% |
| 10 | 30 | 98.00% | 26 | 99.07% |
| 11 | 34 | 97.73% | 20 | 99.29% |
| 12 | 20 | 98.67% | | |
| 13 | 19 | 98.73% | | |
| 14 | 26 | 98.27% | | |
| 15 | 22 | 98.53% | | |

In VLMD algorithm, motif groups are constructed based on sensitivity of motifs as defined in Definition 9. In other words, our proposed algorithm, VLMD, returns an exact solution to the variable-length motif discovery problem. We plot the sensitivity of motifs from some sliding window lengths in Figure 4. The dashed vertical lines in the figure show the separator between motif groups. This means VLMD can correctly group motifs according to sensitivity value. In addition, ten motif representatives of the 23 discovered motif representatives in Dataset 1 of Gun dataset are also shown in Figure 5.

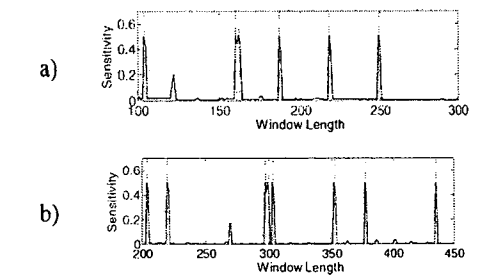



Figure 4. Sensitivities of a) Dataset 1 of Gun dataset and b) Dataset 1 of Face Four dataset.

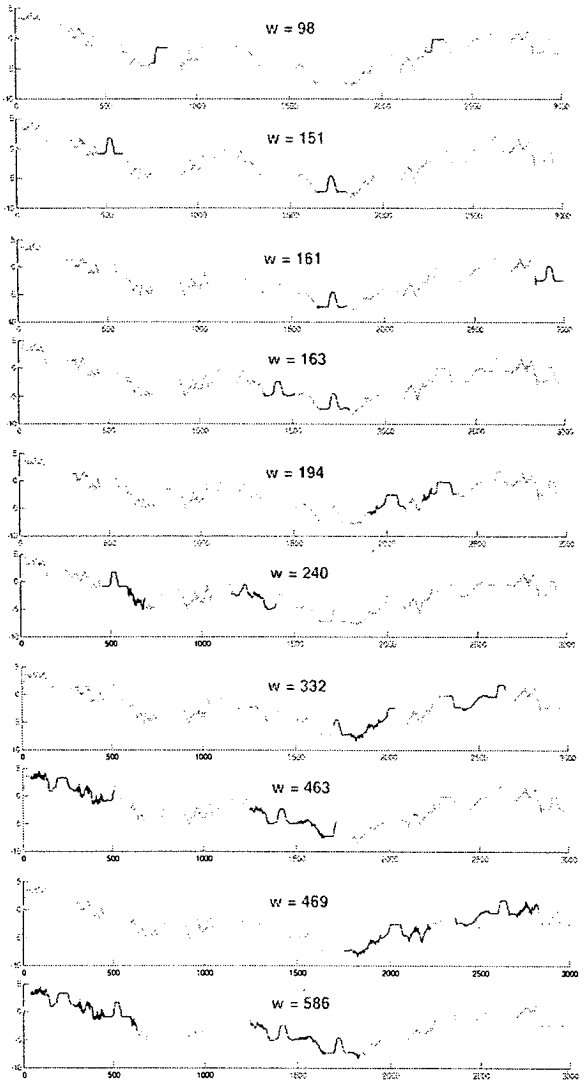



Figure 5. Ten motif representatives from Dataset 1 of Gun dataset.

## V. CONCLUSION AND FUTURE WORK

In this work, we introduce a novel parameter-free variable-length motif discovery called VLMD which returns a small set of motifs from a given time series sequence. VLMD eliminates the need of a predefined crucial parameter, the sliding window length, of the typical motif discovery algorithm. We have demonstrated that this particular parameter is very sensitive; a small difference of sliding window may lead to huge difference in the discovered motifs. VLMD models the sensitivity of the sliding window length, and accordingly returns a set of most sensible motifs. VLMD can prune off a large possibility of the length of sliding window up to 99.29%, while being able to return an exact solution to the motif discovery problem. We can further extend this work by devising some ranking scheme to the algorithm such that suggestion of the most useful/interesting motifs is returned to the users.

## ACKNOWLEDGMENT

This research is partially supported by the Thailand Research Fund (Grant No. MRG5380130), the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0141/2549 to V. Niennattrakul).

## REFERENCES

[1] P. Beaudoin, S. Coros, M. V. Panne, and P. Poulin, "Motion-Motif Graphs," in *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, 2008.

[2] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic Discovery of Time Series Motifs," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[3] A. W. Fu, O. T. Leung, E. Keogh, and J. Lin, "Finding Time Series Discords Based on Haar Transform," in *Proceedings of the Advanced Data Mining and Applications, Second International Conference* Xi'an, China, 2006.

[4] T. Guyet, C. Garbay, and M. Dojat, "Knowledge Construction from Time Series Data Using a Collaborative Exploration System," *Journal of Biomedical Informatics*, vol. 40, pp. 672–687, 2007.

[5] E. Keogh, J. Lin, A. Fu, and H. V. Herle, "Finding the Unusual Medical Time Series: Algorithms and Applications," in *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems* Dublin, Ireland, 2005.

[6] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "The UCR Time Series Classification/Clustering Homepage," 2008; www.cs.ucr.edu/~eamonn/time_series_data/

[7] B. LeBaron, W. B. Arthur, and R. Palmer, "Time Series Properties of an Artificial Stock Market," *Journal of Economic Dynamics & Control*, vol. 23, pp. 1487-1516, 1999.

[8] Y. Li and J. Lin, "Approximate Variable-Length Time Series Motif Discovery Using Grammar Inference," in *Proceedings of the Tenth International Workshop on Multimedia Data Mining* Washington DC, U.S.A., 2010.

[9] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a Novel Symbolic Representation of Time Series," in *Data Mining and Knowledge Discovery Journal*, 2007.

[10] J. Meng, J. Yuan, M. Hans, and Y. Wu, "Mining Motifs from Human Motion," in *Proceedings of the 29th Annual Conference of the European Association for Computer Graphics* Crete, Greece, 2008.

[11] D. Minnen, C. L. Isbell, I. Essa, and T. Starner, "Discovering Multivariate Motifs Using Subsequence Density Estimation and Greedy Mixture Learning," in *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, 2007.

[12] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact Discovery of Time Series Motifs," in *Proceedings of the Ninth SIAM International Conference on Data Mining*, 2009.

[13] V. Niennattrakul and C. A. Ratanamahatana, "On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping," in *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering* Seoul, Korea, 2007.

[14] V. Niennattrakul, D. Wanichsan, and C. A. Ratanamahatana, "Meaningfulness Subsequence Matching on Data Stream under Time Warping Distance," in *The 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining* Bangkok, Thailand, 2009.

[15] C. A. Ratanamahatana and E. Keogh, "Making Time-series Classification More Accurate Using Learned Constraints," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004.

[16] S. Rombo and G. Terracina, "Discovering Representative Models in Large Time Series Databases," in *Proceedings of the 6th International Conference on Flexible Query Answering Systems*, 2004.

[17] P. Sajjipanon and C. A. Ratanamahatana, "A Novel Fractal Representation for Dimensionality Reduction of Large Time Series Data," in *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining* Bangkok, Thailand, 2009.

[18] Y. Tanaka, K. Iwamoto, and K. Uehara, "Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle," *Machine Learning*, vol. 58, pp. 269-300, 2005.

[19] H. Tang and S. S. Liao, "Discovering original motifs with different lengths from time series," *Knowledge-Based Systems*, vol. 21, pp. 666–671, 2008.