Noname manuscript No.
(will be inserted by the editor)

Asymptotic Dynamic Time Warping Calculation with Utilizing Values Repetition

Anooshiravan Sharabiani · **Houshang Darabi** ·
Samuel Harford · **Elnaz Douzali** · **Fazle Karim** ·
Hereford Johnson · **Shun Chen**

Received: date / Accepted: date

Abstract Dynamic Time Warping (DTW) is a popular method for measuring the similarity of time series. It is widely used in various domains. A major drawback of DTW is that it has a high computational complexity. To address this problem, pruning techniques to calculate the exact DTW distance, as well as DTW approximation methods, have become important approaches. In this paper, we introduce Blocked Dynamic Time Warping (BDTW), a new similarity measure which works on run-length encoded time series representation. BDTW utilizes any repetitive values (zero and non-zero) in time series to reduce DTW computation time. BDTW closely approximates DTW distance, and it is significantly faster than traditional DTW for time series with high levels of values repetition. Moreover BDTW can be combined with time series representation methods which provide constant segments, to serve as a close approximation method even for the time series without values repetition. Constrained BDTW, BDTW upper bound and BDTW lower bound are discussed as variations of BDTW. BDTW upper bound and BDTW lower bound are presented as a new DTW upper bound and lower bound which can be efficiently applied on time series with high levels of values repetition for pruning unhopeful alignments and matches in the exact DTW calculation. We show the effectiveness of BDTW and its variations on different applications using the following datasets: Almanac of Minutely Power, Refit Smart Homes, as well as the 85 datasets from the University of California, Riverside time series classification archive (UCR archive).

Keywords Dynamic Time Warping · Computational complexity · Similarity search · Classification · Lower bounds · Upper bounds

Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, IL 60607 USA.
A. Sharabiani E-mail: ashara3@uic.edu.com
Tel.: +1-312-996-6593
Fax: +1-312-996-6593
· H. Darabi E-mail: hdarabi@uic.edu
· S. Harford E-mail: sharfo2@uic.edu
· E. Douzali E-mail: edouza2@uic.edu
· F. Karim E-mail: karim1@uic.edu
· H. Johnson E-mail: hjohns23@uic.edu
· S Chen E-mail: shun6266@gmail.com

1 Introduction

When exploring the world of time series similarity measurements, one method that is predominantly used is Dynamic Time Warping (DTW) [2]. Ubiquity of DTW in time series analysis is conspicuous. DTW is used in many different applications [3–8]. Exhaustive literature searches and comprehensive experimental researches have shown that Nearest Neighbor DTW (NN-DTW) is exceptionally hard to beat [9, 14, 15]. Where NN-DTW can be beaten, it is usually by a very small margin and at the cost of huge processing time.

The core drawback of DTW is its computational complexity. There are two important approaches to address this problem: pruning techniques in calculation of the exact DTW, and DTW approximation. Lower bounding methods [19, 20, 29, 30] in similarity search, and using Euclidean distance as an upper bound of DTW to prune unpromising warping paths in the DTW matrix [10], are examples of pruning techniques in calculation of the exact DTW. Different approximation methods have been also developed to make DTW faster while keeping its accuracy level [25–27, 31]. Incorporating constraint bands to limit the search area in DTW matrix (Constrained warping), and using DTW on reduced representation of the time series are examples of these approximation methods.

In 2016, a fast warping distance (AWarp) was introduced which is considered a new powerful approximation method for calculation of DTW distance. Implementation of this method is limited to sparse time series- a special case of time series which contain repetitions of zero (runs of zeros) as well as non-zero observations. [11]. AWarp takes advantage of repetition of zeros in time series, however there are many real time series with repetition of values that the repetitive values are non-zero. In these cases AWarp is not useful anymore. Fig. 1. shows some examples of time series with no-zero values repetition from UCR time series archive [12]. In this figure, whenever a time series has a constant segment, it means a value is repeated during that period. The time series with more and longer constant segments have higher repetition rates.

In this paper, we propose Blocked Dynamic Time Warping (BDTW) as a new warping distance and its variants which work on any time series with repetition of any values (zeros or non zeros).

Our contributions in this paper are

- We present BDTW, an algorithm that generalizes the utilization of the repetition of any values in time series to calculate exact DTW for *any-two-valued* time series, and to calculate a close approximation of DTW for *more-than-two-valued* time series with repetition of *any values*.
- We present a new DTW approximation method for *all type of time series (with or without repetition of values)* based on combination of Adaptive Piecewise Constant Approximation (APCA) and BDTW method.
- We present BDTW Upper Bound (BDTW_UB) and BDTW Lower Bound (BDTW_LB) as new upper and lower bounds which are competitive with famous upper and lower bound techniques in pruning power for time series with high level of value repetitiveness.
- We show that BDTW a) significantly reduces the processing time of DTW calculation for time series with high repetition rate b) combined with APCA, serves as a DTW approximation method that beats traditional Piecewise Aggregate Approximation (PAA)-DTW-Projection approach in accuracy. c) is extendable to Constrained warping and Constrained BDTW processing time is significantly less than Constrained DTW processing time for time series with high value repetition rate.

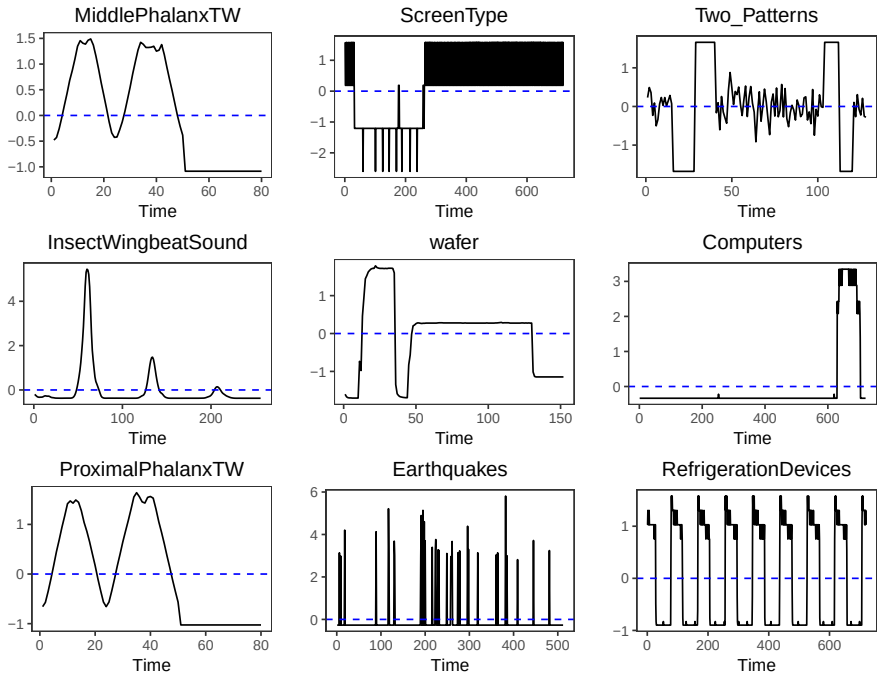


Fig. 1 Example of time series with values repetition

The remainder of paper is organized as followed: In Sect. 2, traditional DTW, lower bound techniques, time series representation and DTW approximation methods are briefly explained as the background. In Sect. 3, Blocked DTW is introduced with its variants and applications. BDTW algorithms performance are evaluated and analyzed in Sect. 4. Next, conclusion is presented in Sect. 5.

2 Background and Related Work

2.1 Euclidean Distance and Dynamic Time Warping

A time series $T = o_1, o_2, \cdots, o_n$ is defined as an ordered list of observations made at successive equally spaced points in time. For two time series of $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_m$ of length n and m, the most common method to measure their distance is based on Euclidean distance (ED). This is derived as the difference between the points that lie on X and Y where n is equal to m:

$$ED(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

Dynamic Time Warping (DTW) algorithm is used to non-linearly align the points of two time series and calculate the optimal warping path with the shortest distance between them. DTW distance calculation can be presented in a matrix view. Each cell of n-by-m DTW

matrix represents an alignment between two points of two time series X and Y. Permissible alignments (paths) bound to the following conditions [1]:

- **Boundary condition**: The paths should start from alignment of the beginning of time seires (x_1, y_1) and end at alignment of the ending of time series (x_n, y_m).
- **Continuity condition**: The paths should have no jumps. The next step for each point (i,j) in the path must be $(i+1,\ j)$, $(i,\ j+1)$ or $(i+1,\ j+1)$.
- **Monotonicity condition**: The warping paths can only go forward in time.

P is defined as a continuous path of cells in the matrix (alignments) from (x_1, y_1) to (x_n, y_m). The s^{th} element of P is defined as $p_s = d(i,j)_s$; where, $d(i,j) = (x_i - y_j)^2$, S is the length of P and

$$P = p_1, p_2, \ldots, p_s, \ldots, p_S$$

DTW must find the optimal path which minimizes the path warping distance:

$$DTW(X,Y) = \min\left\{\sqrt{\sum_{s=1}^{S} p_s}\right\}$$

The optimal path can be calculated through the application of dynamic programming and a recursive function as:

$$DTW(X,Y) = \sqrt{D(i,j)}$$

where $D(i,j)$ is calculated by:

$$\begin{aligned} D(i,j) = (x_i - y_j)^2 + min[D(i-1,j-1), D(i,j-1), D(i-1,j)] \\ D(0,0) = 0, D(0,1:m) = \infty, D(1:n,0) = \infty \\ (i = 1,2,\cdots,n; j = 1,2,\cdots,m) \end{aligned} \tag{2}$$

D(i,j) is the cumulative distance from the first cell to cell i and j. The exact DTW distance is the square root of $D(n,m)$. Here we omit the step of taking the square root since it has no influence on the resulting ordered distance of paired time series, and reduces the computational cost of the technique.

Warping constraint is a constraint which is usually used in DTW calculation to reduce the processing time. It limits the allowed alignments based on the time difference of observations. By using this constraint, only the cells close to the diagonal of the matrix and within the limitations of a window (warping window) are considered in path calculation. Unfortunately, if outside the window, the globally optimal path will not be found, however, in these cases warping constraint still provides a close approximation of the exact DTW, and in general, it is commonly used in different applications. Several studies show that a warping window of 10% (or less) of the length of time series usually provides better results in NN-DTW classification [32].

2.2 Motivation: Warping Distance for Sparse Time Series

In 2016, the AWarp algorithm was introduced to approximate DTW on sparse time series [11]. Awarp is the motivation of this paper. Some of most common sparse time series consists of zeroes and numerical observations that are spread out disproportionately. AWarp proposes an upper bound and a lower bound of DTW for sparse time series. BDTW incorporates the idea of taking advantage of value repetition form AWarp. We emphasize that while

AWarp focuses only on spare time series with runs of zeros, BDTW is comprehensive since it considers runs and repetitions of all values.

Table 1, summarizes the comparison of the application of AWarp and BDTW. In binary-valued time series AWarp and BDTW both obtain exact DTW distance. However, Awarp only reduces the length of time series based on runs of zeros. In AWarp, runs of non-zero values (runs of ones) in binary-valued time series does not help in reducing the length of time series in encoding and reduction of processing time of the exact DTW calculation. On the other hand, BDTW utilizes the repetition of both zeros and ones. AWarp obtains exact DTW only for binary-valued time series, but BDTW for any two-valued time series provides the exact DTW. For more-than-two-valued times eries, AWarp is only useful when there is repetition of zeros. On the other hand BDTW is helpful when there is repetition of any values. When BDTW is combined with APCA method it is useful on time series even without value repetition.

Another advantage of BDTW over AWarp is its compatibility to z-normalization. If we consider a pair of sparse time series with runs (repetition) of zero, it is very typical that after z-normalization runs of zeros would change to runs of non-zero values and in this case, AWarp algorithm is not applicable anymore. In contrary, BDTW does not suffer from this problem, and after z-normalization we would still be able of apply it. This feature is important because in many cases z-normalization is a necessary step in preprocessing of classification to avoid amplitude and offset invariance of time series.

Table 1 The comparison of the application of AWarp and BDTW

Time series type	Usable method		Output
Two valued- binary-with repetition of zeros	AWarp	BDTW	exact DTW
Two valued- binary-with repetition of on-zero values	-	BDTW	exact DTW
Two valued-non binary-with repetition of any values	-	BDTW	exact DTW
More than two valued-with repetition of zeros	AWarp	BDTW	DTW upper and lower bounds
More than two valued-with repetition of any values	-	BDTW	DTW upper and lower bounds
More than two valued- without any values repetition	-	BDTW+APCA	DTW approximation

2.3 Data Reduction and DTW Approximation

One of the problems that similarity search algorithms face is handling time series with huge length. Many of the most effective and popular resolutions for similarity search incorporate time series representation. Thus, there have been multiple contributions which focus on time series representation/approximation with reduction of time series length. Here, we review two popular methods which give the constant segments: Piecewise Aggregate Approximation (PAA) [13], and Adaptive Piecewise Constant Approximation (APCA) [18].

2.3.1 Piecewise Aggregate Approximation

Let n be the length of a time series $X = x_1, x_2, \cdots, x_n$, and w be the length of the transformed time series where w is a factor of n. Essentially, to reduce the time series length from n to w , begin by dividing the time series into w equal segments. The mean value of the points that lie within each segment is calculated as:

$$\overline{X_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{(\frac{n}{w})i} (x_j) \tag{3}$$

Note that when $w = n$, the transformed representation remains like the original time series, and when $w = 1$, the transformed sequence is the mean of the original time series.
In PAA, ith segment can be considered as $\overline{X_i}$ value being repeated $\frac{n}{w}$ times on that segment.

2.3.2 Adaptive Piecewise Constant Approximation

Adaptive Piecewise Constant Approximation (APCA) attempts to improve the performance quality of the PAA approximation by permitting the segments to have arbitrary lengths as opposed to fixed [13, 18]. To achieve this, APCA stores two numbers: the value of the mean of all points in each segment as well as the length of the segment. For a time series $T = \{t_1, t_2, \cdots, t_n\}$, APCA, with w segments is presented as:

$$T = \{< tv_1, tr_1 >, < tv_2, tr_2 >, \cdots, < tv_w, tr_w >\} \qquad tr_0 = 0 \tag{4}$$

Where tv_i is the segmented average of the ith region, and tr_i is the right most data point of the ith region. The length of each segment can be calculated by the difference of each endpoint: Length of segment $i = tr_i$ - tr_{i-1}

In APCA, ith segment can be considered as tv_i value, being repeated $(tr_i - tr_{i-1})$ times on that segment.

2.3.3 PAA Based DTW Approximation

Due to the known efficacy of DTW, and the fact that the traditional DTW algorithm is inherently sluggish, there have been efforts to improve its performance through approximation. One of the famous approximation approaches is applying DTW on a reduced representation of the time series in particular, PAA [25–27]. First PAA transforms the time series to a higher level of abstraction (lower resolution), then DTW is used to find the optimal path on that reduced representation and finally the optimal path is mapped (projected) to the original time series. Note that in this approach, PAA is only usable (not APCA) because of the equal segment lengths in the abstraction. In Sect. 3, we discuss application of BDTW (combined with PAA or APCA) for DTW approximation in competition with traditional PAA-DTW-Projection approach for time series even without value repetition.

2.4 Pruning Techniques to Accelerate Exact DTW Calculation

2.4.1 Lower Bounding

Lower bounding is used to avoid the expensive full calculation DTW matrix, when a candidate time series has no chance of being the best match in similarity search. Using lower bounding accelerates the nearest neighbor search by pruning off unhopeful candidates [19, 20, 29, 30]. LB- Kim, LB- Yi and LB- Keogh are popular lower bounds for DTW. For a candidate time series $C(c_1, \cdots, c_i, \cdots, c_n)$, and query time series $Q(q_1, \cdots, q_i, \cdots, q_n)$, LB-Kim [21] is calculated as:

$$LB-Kim(Q,C) = \sqrt{(c_1-q_1)^2 + (c_n-q_n)^2 + (Min_{i=1}^n(c_i) - Min_{i=1}^n(q_j))^2 + (Max_{i=1}^n(c_i) - Max_{i=1}^n(q_i))^2} \tag{5}$$

LB - Yi [22] is calculated as:

$$LB-Yi(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (Min_{i=1}^n(q_i) - c_i)^2 & if \quad c_i < Min_{i=1}^n(q_i) \\ (c_i - Max_{i=1}^n(q_i))^2 & if \quad c_i > Max_{j=1}^n(q_i) \\ 0 & otherwise \end{cases}}, \tag{6}$$

LB- Keogh [23, 24] is a powerfull lower bound but it can only be used on constrained warping window DTW.

2.4.2 Using an Upper Bound to Speed up All-Pairwise DTW Matrix Calculation

Recently, Silva and Batista proposed PrunedDTW which is an exact approach for accelerating all-pairwise DTW matrix calculation [10]. All-pairwise DTW is used for clustering and classification of time series. PrunedDTW method uses an upper bound to prune unhopeful warping alignments inside DTW matrix. The idea behind PrunedDTW is based on a simple observation: in DTW matrix, usually the value of the cells around the optimal path are relatively low. Therefore, if a cell has a relatively high value it is not very likely that optimal path passes through that cell. PrunedDTW uses Euclidian Distance (ED), which is an upper bound of DTW, as a threshold to prune the cells (alignments) that have no chance of being on the optimal path.

The advantage of using ED as upper bound is that ED is linear in time complexity. However it is not a tight lower bound and its pruning power in all-pairwise DTW matrix calculation is limited. Finding a better upper bound increases the efficiency of PrunedDTW.

3 Blocked Dynamic Time Warping

3.1 Time Series Encoding

The first step starts by defining the length-encoding method that we use for reducing the size of time series with repetitive values. The quantity of sequential run lengths of a value within a time series is represented inside a parenthesis.

For example, for time series $T = \{1,3,3,2,3,3,3,3,3,1,1,2,2,2,2,1\}$, the encoded format $(T_{Encoded})$ is $\{1(1),3(2),2(1),3(5),1(2),2(4),1(1)\}$. Note that in this example time series T size (length) is 16 but after encoding the size reduces to 7. Since DTW path should start from alignment of the beginning of time seires and end at alignment of the ending of time series, we need to have the first value and the last value of the encoded time series with one repetition. Therefore if a time series start with a repetitive value we split that repetition into two parts, first part is with one repetition and the second part is with remaining of repetition. Moreover if a time series ends with a repetitive value, then after split the second part is with one repetition and the first part is with rest of repetitions. For example assume the original time series is $\{3,3,3,3,3,1,1,1,2,2,2,2,2,2\}$, after encoding it is presented as $\{3(1),3(4),1(3),2(5),2(1)\}$.

The level of repetitiveness of a time series is given by a ratio named time series repetition ratio. For one time series, if the original length is l and encoded length is l', the time series repetition ratio (trr) can be defined as:

$$trr = 1 - \frac{l'}{l} \tag{7}$$

Assume that for a dataset, D, with K time series each with length of l_i (where i is from 1 to K), the length of ith time series after the encoding is l'_i. Then the repetitiveness ratio (rr) of D can be define as:

$$rr = 1 - \frac{\sum_{i=1}^{K} l'_i}{\sum_{i=1}^{K} l_i} \tag{8}$$

If the original time series have equal length of n, the formula changes to:

$$rr = 1 - \frac{\sum_{i=1}^{K} l'_i}{Kn} \tag{9}$$

	1	0	0	1	0	1	1	1	0
0	1	1	1	2	2	3	4	5	5
1	1	2	2	1	2	2	2	2	3
1	1	2	3	1	2	2	2	2	3
1	1	2	3	1	2	2	2	2	3
1	1	2	3	1	2	2	2	2	3
0	2	1	1	2	1	2	3	3	2
1	2	2	2	1	2	1	1	1	2
0	3	2	2	2	1	2	2	2	1
0	4	2	2	3	1	2	3	3	1
0	5	2	2	3	1	2	3	4	1
1	5	3	3	2	2	1	1	1	2

a) Traditional DTW matrix

	1(1)	0(2)	1	0	1(3)	0(1)
0(1)	1	1	2	2	5	5
1(4)	1	3	1	2	2	3
0(1)	2	1	2	1	3	2
1(1)	2	2	1	2	1	2
0(3)	5	2	3	1	4	1
1(1)	5	3	2	2	1	2

b) Block DTW matrix

Fig. 2 Traditional DTW matrix and Block DTW matrix for two binary time series (X and Y)

3.2 Blocked Dynamic Time Warping Structure

Definition A block is a group of cells in a DTW matrix that correspond to alignments of repetitive values of two time series. Fig. 2 illustrates an example of a DTW matrix of two time series (with binary values) in which blocks are represented in different colors. Each block in the conventional DTW matrix, Fig. 2-a, is represented by one cell in our contribution, the Blocked Dynamic Time Warping matrix in Fig.2-b.

The Blocked Dynamic Time Warping (BDTW) algorithm (Algorithm 1) reduces the computational complexity of calculating the optimal warping path on encoded time series. The input of this algorithm include two encoded time series. Assume that after encoding the length of two time series are l'_x and l'_y.Then an l'_x-by-l'_y matrix is created. Within this matrix, each cell of this matrix corresponds to a block in a traditional DTW matrix.

For example assume $[a(A)]_i$ is the ith point on encoded time series x and $[b(B)]_j$ is the jth point on encoded time series y where a and b are values and A and B are the number of repetition of these values. For simplicity we disregard the indexes and refer to these points as $a(A)$ and $b(B)$. The intersection of $a(A)$ and $b(B)$ on the cost matrix corresponds to a block in a traditional cost matrix where a and b are repeated A and B times, respectively. Fig. 3, shows the structure of BDTW cost matrix.

$D_{i,j}$ (the accumulate distance in intersection of $a(A)$ and $b(B)$), is calculated based on the distance of the points in that cell as well as previous accumulative costs in top $(D_{i-1,j})$, left$(D_{i,j-1})$ or diagonal cells $(D_{i-1,j-1})$. $D_{i,j}$ can be derived from specific patterns which use: the values of the time series, their repetitive numbers, and the direction of the warping path.

If the direction of the path is coming from the top, the resulting distance will be $A*(a-b)^2$ and if the direction is coming from left, the distance will be $B*(a-b)^2$. Finally, if the path is coming from the diagonal direction, the resulting distance will be $max(A,B)*(a-b)^2$. After calculating the distance in each case, the accumulative cost for cell $D_{i,j}$ is the minimum of the accumulative cost from the three possible directions. For example, in case of a and b greater than one $D_{i,j}$ is calculated as:

$$D_{i,j} = min\{D_{i-1,j} + A*(a-b)^2, D_{i,j-1} + B*(a-b)^2, D_{i-1,j-1} + max(A,B)*(a-b)^2\}$$

		1	...	j	...	l'_y
	$y \rightarrow$	...	...	$b(B)$	...	
1	...	...	...	...		
...	...	...	$D_{i-1,j-1\searrow}$	$D_{i-1,j\downarrow}$		
i	$a(A)$	...	$D_{i,j-1\rightarrow}$	$D_{i,j}$		
...	...					
l'_x						
	$x \uparrow$					

Fig. 3 The structure of the BDTW cost matrix

3.3 Using BDTW for DTW Calculation of Different Time Series

3.3.1 BDTW on Two-Valued Time Series with Values Repetitions

When BDTW is applied on two-valued time series the optimal warping path results in the exact DTW distance. One special case of two-valued time series is when the values are zero and one (binary-valued time series). Example: Given original time series $X = \{0,1,1,1,1,0,1,0,0,0,1\}$ with length of 11, and $Y = \{1,0,0,1,0,1,1,1,0\}$ with length of 9, we begin by the transforming X and Y to encoded time series: $X_{Encoded} = \{0(1),1(4),0(1),1(1),0(3),1(1)\}$ and $Y_{Encoded} = \{1(1),0(2),1(1),0(1),1(3),0(1)\}$ reducing both original time series to a length of 6. The encoded time series then fill the outer shell of the BDTW cost matrix. Next, apply the BDTW algorithm, the resulting matrix is represented in Fig. 2-b.The conventional DTW cost matrix is presented in Fig. 2-a. The values in the BDTW cost matrix are identical to the bottom-right values of the corresponding blocks of the DTW matrix. Note that in this example the length time series after encoding are equal, but BDTW Algorithm also works similarly on the encoded time series of different length.

3.3.2 BDTW on More-Than-Two-Valued Time Series with Values Repetitions

In binary-valued time series, the repetitive values play identical roles in the calculation of the block distance, however, dependence assumptions of the accumulative distance of each block is not necessarily correct for more-than-two-valued time series. For example, Fig. 4 shows a traditional DTW matrix for two time series that one of them has more than two values. In the blue block, the cumulative distance (6), cannot be calculated solely based on previous cumulative distance measures of the neighboring blocks, values (3 and 1), and their repetition (3 and 1). The second 3 on the time series also influences the distance of the blue block, which results in that block having a smaller distance (6) than the BDTW distance (7) represented in Fig. 4-b.

	1	**3**	**3**	**3**	**2**
1	**0**	4	8	**12**	**13**
2	**1**	1	2	**3**	**3**
1	**1**	5	5	**6**	**4**
2	**2**	2	3	4	**4**

a) Traditional DTW matrix

	1(1)	**3(3)**	**2(1)**
1(1)	**0**	**12**	**13**
2(1)	**1**	**3**	**3**
1(1)	**1**	**7**	**4**
2(1)	**2**	4	**4**

b) Block DTW matrix

Fig. 4 Traditional DTW matrix and Block DTW matrix for more-than-two-valued time series (X and Y)

Due to this behavior, the final distance calculated with the BDTW algorithm will always be greater than (or equal to) the true DTW distance. Therefore, we can conclude that BDTW is the upper bound of the true DTW distance. Accordingly, we refer to BDTW (algorithm 1) for more-than-two-valued time series as BTDW_UB. Note that it is not possible to estimate the error (the difference betweenBTDW_UB and exact DTW) for not binary time series. Because in exact DTW, an optimal warping alignment in each block may be obtained from different values of repetitive values, however BDTW treats all repetitive values as identical. Since it is not possible to detect which values between repetitive values play a role in the exact DTW calculation, it is not possible to estimate the error.

Algorithm 1 Blocked_DTW_UB(series1, series2)

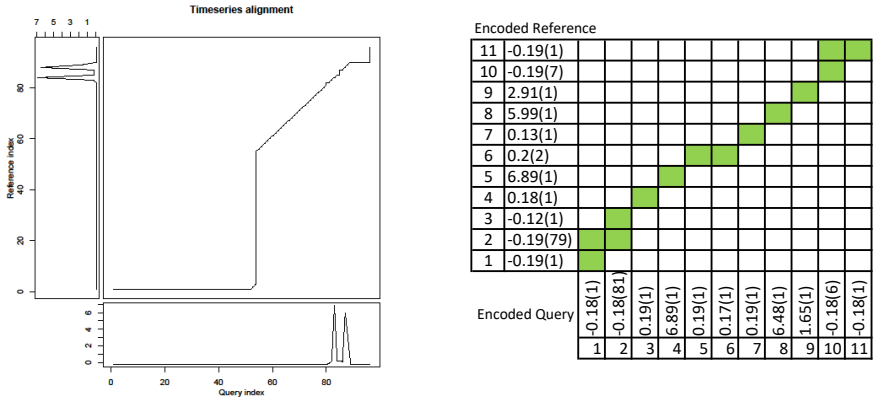
Require:
Series 1 & series 2 ← encoded time series in two-row data frames (values, number of repetitions).
Note: a and b represent the values and, A and B are the number of times these values are repeated.

Ensure:
Output the distance d between series 1 and series 2

$l'_x \leftarrow$ length(series1), $l'_y \leftarrow$ length(series2)
$\mathrm{D}_{1:l'_x,1:l'_y} \leftarrow \infty$
$\mathrm{D}_{1,1} \leftarrow (a_1\text{-}b_1)^2$
for $i \leftarrow 2$ to l'_x **do**
    $\mathrm{D}_{i,1} \leftarrow \mathrm{D}_{i-1,1} + A_i(\mathrm{a}_i - b_1)^2$
end for
for $j \leftarrow 2$ to l'_y **do**
    $\mathrm{D}_{1,j} \leftarrow \mathrm{D}_{1,j-1} + B_j(\mathrm{a}_1 - b_j)^2$
end for
for $i \leftarrow 2$ to l'_x **do**
    for $j \leftarrow 2$ to l'_y **do**
        top $\leftarrow \mathrm{D}_{i-1,j} + A_i(a_i - b_j)^2$
        diagonal $\leftarrow \mathrm{D}_{i-1,j-1} + (\max(A_i, B_j))(a_i - b_j)^2$
        left $\leftarrow \mathrm{D}_{i,j-1} + B_j(a_i - b_j)^2$
        $\mathrm{D}_{i,j} \leftarrow$ min(top, diagonal, left)
    end for
end for
return $\mathrm{D}_{l'x,l'y}$

In Fig. 5, DTW and BDTW UB alignments are graphically compared. Two time series are used as reference and query which are selected from row 1 and row 2 of "ElectricDevices" rain dataset from UCR archive. The length of these time series is 96 and they both have repetitive values. The left figure shows the original time series and DTW optimal path (alignments) after calculating 96-by-96 DTW matrix. The exact DTW distance is 1.858176. In Fig. 5, right figure, time series are encoded and the length of them after encoding is 11. Using BDTW_UB algorithm, the cells in 11-by-11 matrix are calculated and the warping path is determined. The BDTW distance is 1.858309. As expected it is greater than DTW distance. However, it is very close. In general BDTW is a very accurate approximation of DTW. Note that if there is no repetition in two time series the result of BDTW will be same as DTW.

When the direction of the path to each block is from diagonal we can change the distance formula to make the total distance of the matrix, the lower bounding of DTW. If in each block the repetition of one the values is 1, we calculate the distance from diagonal using



DTW alignments BDTW alignments

Fig. 5 Graphical comparison of DTW and BDTW_UB alignments

$(a-b)^2$ and if both values have more than 1 repetition we can use $min(A,B)*(a-b)^2$. This will guaranty that the distance of the block is less than or equal to any optimal path in the block. These changes are applied in Algorithm 2. The output of this algorithm is a distance which is lower bound of DTW and we will refer to it as BDTW_LB.

Algorithm 2 Blocked_DTW_LB(series1, series2)

Require:
Series 1 & series 2 ← encoded time series in two-row data frames (values, number of repetitions).
Note: a and b represent the values and, A and B are the number of times these values are repeated.
Ensure:
Output the distance d between series 1 and series 2

$l'_x \leftarrow$ length(series1), $l'_y \leftarrow$ length(series2)
$D_{1:l'_x,1:l'_y} \leftarrow \infty$
$D_{1,1} \leftarrow (a_1\text{-}b_1)^2$
for $i \leftarrow 2$ to l'_x **do**
  $D_{i,1} \leftarrow D_{i-1,1} + A_i(a_i - b_1)^2$
end for
for $j \leftarrow 2$ to l'_y **do**
  $D_{1,j} \leftarrow D_{1,j-1} + B_j(a_1 - b_j)^2$
end for
for $i \leftarrow 2$ to l'_x **do**
  for $j \leftarrow 2$ to l'_y **do**
    top $\leftarrow D_{i-1,j} + A_i(a_i - b_j)^2$
    diagonal $\leftarrow D_{i-1,j-1} + (\min(A_i, B_j))(a_i - b_j)^2$
    left $\leftarrow D_{i,j-1} + B_j(a_i - b_j)^2$
    $D_{i,j} \leftarrow$ min(top, diagonal, left)
  end for
end for
return $D_{l'x,l'y}$

3.3.3 Constrained Blocked Dynamic Time Warping

Constrained DTW with a user defined warping window is widely used in time series data mining applications. While Constrained DTW does not guarantee the exact DTW distance between two time series, it accelerates the calculations and in some cases it results in better accuracy. Constrained warping is applicable on BDTW. Algorithm 3 represents constrained BDTW for two encoded time series. In line 12,$|ta_i - tb_j|$, ta_i and tb_j are the timestamp of a_i and b_j and condition in line 13 guarantees that if the difference of time stamps is larger than w (defined warping window), then the value of the cell would be infinity to keep the warping path within the boundaries.

Algorithm 3 Constrained_BDTW(series1, series2, w)

Require:
Series 1 & series 2 ← encoded time series in two-row data frames (values, number of repetitions).
Note: a and b represent the values and, A and B are the number of times these values are repeated.
Ensure:
Output the distance d between series 1 and series 2

1: $l'_x \leftarrow$ length(series1), $l'_y \leftarrow$ length(series2)
2: $D_{1:l'_x,1:l'_y} \leftarrow \infty$
3: $D_{1,1} \leftarrow (a_1\text{-}b_1)^2$
4: **for** $i \leftarrow 2$ to l'_x **do**
5: $\quad D_{i,1} \leftarrow D_{i-1,1} + A_i(a_i - b_1)^2$
6: **end for**
7: **for** $j \leftarrow 2$ to l'_y **do**
8: $\quad D_{1,j} \leftarrow D_{1,j-1} + B_j(a_1 - b_j)^2$
9: **end for**
10: **for** $i \leftarrow 2$ to l'_x **do**
11: $\quad$ **for** $j \leftarrow 2$ to l'_y **do**
12: $\quad\quad$ gap $\leftarrow |ta_i - tb_j|$
13: $\quad\quad$ **if** gap $>$ w &&(tb$_{j-1}$ − ta$_i$ $>$ w || ta$_{i-1}$ − tb$_j$ $>$ w) **then**
14: $\quad\quad\quad D_{i,j} \leftarrow \infty$
15: $\quad\quad$ **else**
16: $\quad\quad\quad$ top $\leftarrow D_{i-1,j} + A_i(a_i - b_j)^2$
17: $\quad\quad\quad$ diagonal $\leftarrow D_{i-1,j-1} + (\min(A_i, B_j))(a_i - b_j)^2$
18: $\quad\quad\quad$ left $\leftarrow D_{i,j-1} + B_j(a_i - b_j)^2$
19: $\quad\quad\quad D_{i,j} \leftarrow$ min(top, diagonal, left)
20: $\quad\quad$ **end ifelse**
21: $\quad$ **end for**
22: **end for**
23: **return** $D_{l'x,l'y}$

Each cell in BDTW matrix represent a block in traditional DTW and when we accept that a cell in BDTW is within a boundary, it is like we are assuming that all cells in the corresponding block (of traditional DTW) are within that boundary. This assumption is not necessarily true for all cases. Because of this, constrained BDTW does not result in exact constrained DTW output. Constrained BDTW can be used as an approximation of BDTW_UB and usually it will be faster than BDTW_UB. The small warping window will cause faster path calculation because search area will be limited more.

3.3.4 Using BDTW for DTW Approximation of Any Time Series

We discussed that BDTW is a close approximation method which speeds up DTW calculation for time series with repetitive values. A valid question is if BDTW can also be used for time series without (or with low level of) repetition. The answer is yes. BDTW can be used on top of the representation methods, which transform time series to sequences of constant segments (i.e. PAA or APCA).

As discussed in Sect. 2.3.3, the DTW traditional approximation methods are based on PAA, DTW on reduced presentation and projection. Here we propose a new approach. Since each segment of PAA or APCA representation consist of repetitive values, we can use BDTW to calculate the distance between time series on these representations and there is no need for projection any more (Fig. 6). The input to PAA and APCA is a time series and the number of segments that we want after transformation (resolution). For example in Fig. 6, the number of segments is 7. The number of segments also represents the repetition rate after run-length encoding. For example, assume that the length of a time series is 150. For APCA (or PAA) we define the number of segments at 10% of the total length, then we would have 15 segments. In each of these segments a value is repeated several times (equal to the length of the segment). Therefore, the repetition rate of that time series would also be 10%. For each time series pair, the error of approximation distance can be calculated as:

$$Distance\ \ Error = \frac{|ApproximateDistance - TrueDTWDistance|}{TrueDTWDistance} \tag{10}$$

Using this formula we can evaluate and compare the performance of APCA-BDTW (or PAA-BDTW) with the traditional approximation methods based on PAA, DTW and projection.

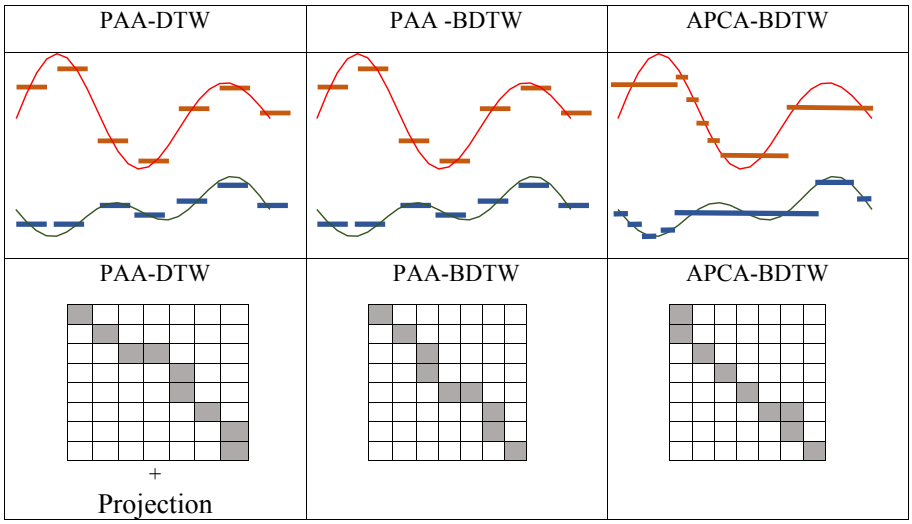


Fig. 6 Graphical comparison of approximation methods (PAA-DTW-Projection/PAA-BDTW/APCA-BDTW)

3.4 Using BDTW as a Pruning Technique to Calculate Exact DTW

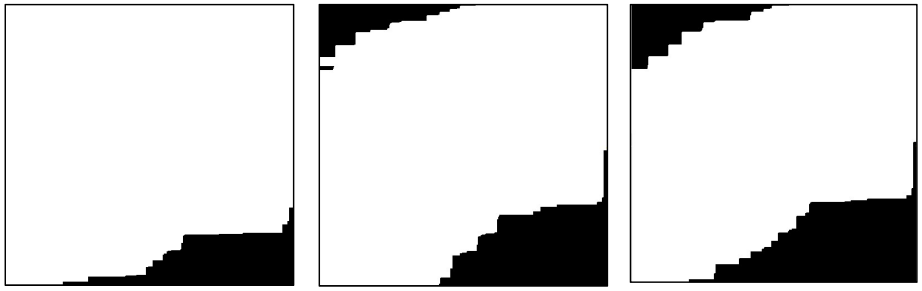
3.4.1 Using BDTW_ LB as a Lower Bound Technique

In Sect. 3.3, we saw that BDTW_UB provides an approximation of DTW, while reducing the computation time for time series with high level of repetition. However, if a user only wants to have the exact DTW distances rather than approximation in classification or clustering of time series, we can still apply BDTW_LB as a useful lower bound to reduce the processing time. Lower bounds performance can be evaluated based on their pruning power and also their own computation time. BDTW_LB's computation complexity is quadratic on the encoded (reduced) time series length, but it is a very tight lower bound with high power of pruning. In experiment result we show that on time series with high level of repetition, BDTW_LB can significantly reduce the classification time. BDTW_LB is competitive to famous lower bounds like Kim_LB or Yi_LB.

3.4.2 Using BDTW_UB as an Upper Bound in PrunedDTW

PrunedDTW uses squared ED as an upper bound to prune unhopeful warping alignments inside DTW matrix in the process of all-pairwise exact DTW calculation. We propose using BDTW_UB as the alternative upper bound instead of ED in PrunedDTW method. Even though the time complexity of BDTW_UB is higher than ED, its pruning power will compensate the difference in time for high repetitive time series. This is due to the fact that it provides a tighter lower bound.

Fig. 7, shows an example of pruning unpromising alignments on DTW matrix using different upper bounds including ED, BDTW and exact DTW (OracleDTW). OracleDTW is DTW pruned by the exact DTW distance as the upper bound. In practice we do not have the exact DTW distance to use it as the upper bound for pruning. We assume that it is imaginary given to us just to evaluate the highest possible pruning power. Therefore, OracleDTW presents the best possible performance of PrunedDTW using the optimal upper bound. The black areas present the pruned alignments (cells) and the white areas present the alignments that their distance should be calculated (not pruned). For this example in Fig. 7, BDTW performs better than ED. It gives almost the same pruning power when compared to using the exact DTW as the upper bound. We should also consider the overhead of calculating BDTW. However, for high repetitive time series BDTW calculation will be fast enough to justify this overhead.



a) UB: ED b) UB:BDTW c) UB:Exact DTW

Fig. 7 An example of pruning unpromising alignments using different upper bounds (ED, BDTW and exact DTW)

4 Experiments

In this section, the BDTW distance is evaluated based on the case studies and UCR benchmark datasets [12]. First in the case studies, the performance of BDTW is compared with the performance of traditional DTW and AWarp. Then using UCR benchmark datasets, the performance of BDTW and its variants are evaluated on 5 different application areas: First, using BDTW_UB as an approximation method of DTW for time series with high repetition rate. Secondly, using BDTW_UB on top of APCA representation method to approximate DTW distance for any time series (even without or low value repetition). Thirdly, using BDTW_LB as a tight lower bound for pruning the time series that could not be best match in order to accelerate searches . Fourthly, using BDTW_UB for pruning unhopeful warping alignments to speed up the all pairwise DTW matrix calculation and finaly the evaluation of Constrained BDTW and comparing it with regular Constrained DTW.

4.1 Case Studies

4.1.1 Case Study 1: Power Consumption Classification

To illustrate the effectiveness of BDTW over AWarp and traditional DTW methods, we use a case study to determine the sources of power consumption. The Almanac of Minutely Power dataset Version 2 (AMPds2) is utilized as the data source for this study. AMPds2 is a dataset made available by [16] to researches to test models, systems and algorithm on real world data. Available datasets have been pre-cleaned to provide comparable accuracy and speed results amongst different researchers. The dataset of interest for this case study is the Electricity-P data. Over a two-year period, data points have been collected for a single house in Canada. These data points include information about electricity, water, and natural gas usage. For the proposes of this study, only the data points related to electrical power consumption are used. Since this dataset contains information for a two-year period, each component can be broken into 730 time series, one per day. This process is followed for the components of interest (B1E, B2E, BME, CDE, CWE, DNE, DWE, EBE, EQE, FGE, FRE, HPE, HTE, OFE, OUE, TVE, UTE, WOE). These components correspond to different sources of electrical consumption such as dining room, furnace, outside, etc. Fig. 8, shows the example of some of these time series. The sample series used in this figure, are the first rows of the training sets. Note that since in some periods of time some components are not active, their power consumption are zero. Therefore time series commonly have repetition of zero and non-zero values and it makes sense to apply AWarp and BDTW on them. Based on this information, the goal is to be able to determine the source of a series. For demonstration purposes, we limit our testing to 30 days worth of information for each component. The data is tested using Leave Out One Cross-Validation. The classification problem is run using BDTW, AWarp, DTW, Euclidian Distance, Constrained DTW, and Constrained BDTW. The result is presented in Table 2.

The result shows that ED accuracy (0.561) is much lower than BDTW accuracy (0.807). BDTW accuracy (0.807) is exactly same as DTW accuracy (0.807) and slightly better than AWarp (0.806). BDTW is roughly 10 times faster than DTW and 3.7 times faster than AWarp. The comparison of CDTW and CBDTW with the same percentage of warping window (10%), shows that CBDTW is almost 3 times faster than CDTW with slightly better accuracy.

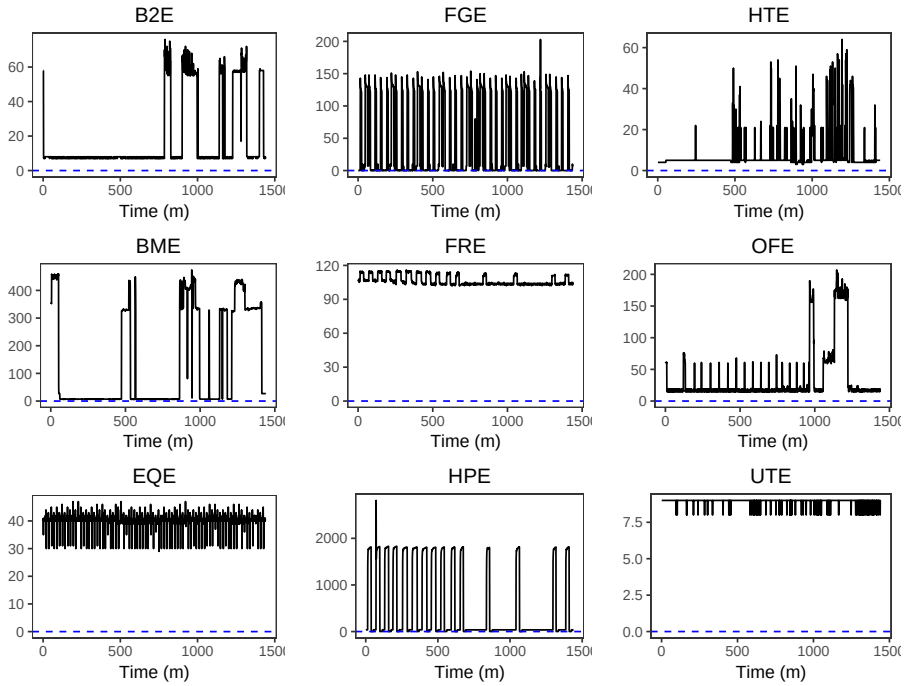


Fig. 8 The example of some of electricity components time series

Table 2 The comparison of accuracy and processing time- power consumption case

Method	**ED**	**DTW**	**Awarp**	**BDTW**	**CDTW**	**CBDTW**
Accuracy	0.561	0.807	0.806	**0.807**	0.761	0.776
Processing time (s)	51.96	8514.20	3237.53	**877.57**	936.56	307.50

4.1.2 Case Study 2: Household Electrical Measurements

To further show the performance and application of the BDTW method, we use data collected on household electrical measurements form [17]. This dataset contains information about the electrical usage of different appliances within 21 houses. For the purposes of this case study, we focus on two households, houses 3 and 8. To get extract the information for each household from the original data we use the supplied UNIX time to split the data into time that represent a full day. The data is collected on average each day consists of 10,800 data points, which represents one reading every eight seconds. For this study, we limited the data to the first 100 full days for each household and just the data that occurred between 5:00pm and 11:00pm (consumption peak hours). Using this subset of data, we aim to demonstrate the capabilities of BDTW for classifying a household in two different contexts, aggregate and freezer power consumption. Fig. 9 shows two sample time series of aggregate and freezer power consumption. This figure shows that for aggregate time series there are constant segments (the repetition of values) but all repetition values are non-zero. On the other hand in Freezer time series we see both the repetition of zero and non-zero values.

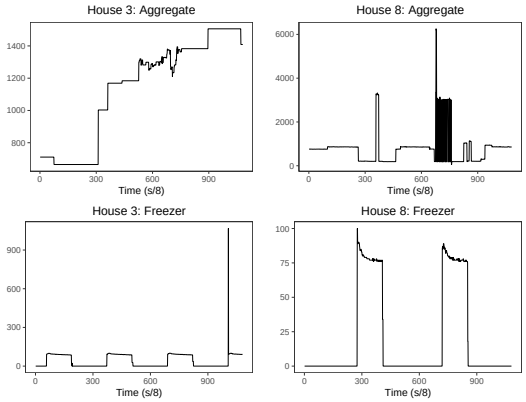


Fig. 9 Two sample time series of aggregate and freezer power consumption

The classification problem to detect the households based on the signals is run using BDTW, AWarp, DTW, Euclidian Distance, Constrained DTW, and Constrained BDTW. The data is tested using Leave Out One Cross-Validation. The result is presented in Table 3.

Table 3 The comparison of accuracy and processing time-household electrical measurements problem

Dataset	Repetition Rate	Method	ED	DTW	Awarp	BDTW	CDTW	CBDTW
Houses 3,8 Aggergate	49.80%	**Accuracy**	0.87	0.90	0.90	0.90	0.89	0.87
Houses 3,8 Aggergate	49.80%	**Processing time(s)**	5.67	143.85	174.98	57.41	71.93	33.20
Houses 3,8 Freezer	91.13%	**Accuracy**	0.64	0.86	0.86	0.86	0.73	0.77
Houses 3,8 Freezer	91.13%	**Processing time(s)**	17.62	1394.83	117.23	28.76	697.42	14.95

In both cases (Aggregate and Freezer), the accuracy of DTW, AWarp and BDTW are equal and higher than accuracy of ED. Regarding the processing time for Aggregate case, since there is no repetition of zero values in time series AWarp is not useful. AWarp processing time (174.98s) is even higher than DTW processing time (143.85s). BDTW processing time is 57.41s, which means BDTW is 3 times faster than AWarp and 2.5 times faster than DTW. In Freezer case, since there is repetition of zero value in the time series (along with repetition of non-zero values), AWarp is to some extend useful in reduction of processing time. However since BDTW utilizes the repetition of any values (zero and non-zero values), it is significantly faster than AWarp. BDTW is 48.5 times faster than DTW and 4 times faster than AWarp. The efficiency improvement of BDTW in Freezer case is more than Aggregate case, because the repetition rate of Freezer dataset (91.13%) is more than Aggregate repetition rate (49.87%). In both cases CBDTW also obtains similar or better accuracy than CDTW with less processing time. CBDTW is more than 2 times faster than CDTW in Aggregate case and 46.6 time faster than CDTW in Freezer case.

4.2 UCR Benchmark

During all the experiments for classification, a 1- Nearest Neighbor (1-NN) Classifier is used under different distance measures such as Euclidean distance, DTW distance, AWarp distance and our proposed BDTW distance. We use all available datasets in the UCR public benchmark datasets archive [12] for our empirical evaluation. In the UCR time series classification archive, there are 85 (including both real and synthetic) time series with dif-

ferent length, repetition level and number of classes. All experiments are run on the same computer. All the codes and detail of results are shared on:

http://prominent.mie.uic.edu/Prominent/BDTW

4.3 BDTW_UB as an Approximation Method of DTW

Fig. 10, shows the classification processing time of DTW over BDTW based on different repetition rates (the x axes). Each point in this figure represents a dataset from UCR Archive. This figure shows that the superiority of BDTW speed over DTW speed increases significantly when *rr* decreases. When a dataset's repetition rate is higher than 0.25 we can expect gain in processing time. There are 14 datasets with *rr* of higher than 0.25 which are listed in Table 4. In all of them BDTW is faster than traditional DTW. For the rest of datasets (with repetition rate of less than 0.25), the BDTW processing time is equal or slightly longer than DTW processing time. The highest repetition rate is for "SmallKitchen Appliances" which BDTW is approximately 3.5 times faster than DTW.

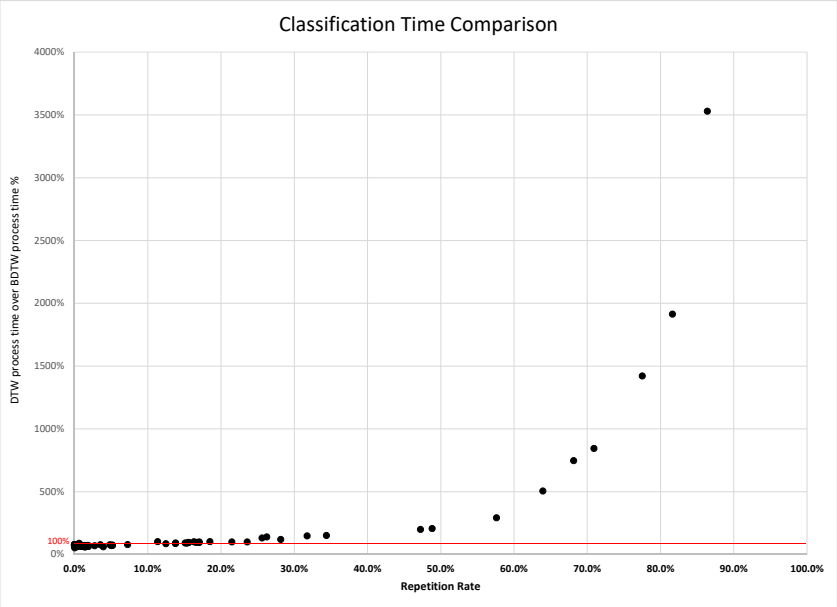


Fig. 10 Speed gain of BDTW for 85 time series datasets in UCR archive

Table 4 The list of datasets with shorter processing time using BDTW

Dataset	Repetition Rate	Speed up percentage	Dataset	Repetition Rate	Speed up percentage
SmallKitchenAppliances	86.4%	3529%	wafer	48.8%	206%
LargeKitchenAppliances	81.6%	1914%	FaceFour	47.2%	199%
ScreenTyp	77.5%	1421%	Two_Patterns	34.4%	151%
Computers	70.9%	844%	uWaveGestureLibrary_Y	31.8%	148%
Earthquakes	68.1%	747%	uWaveGestureLibraryAll	28.2%	120%
RefrigerationDevices	63.9%	505%	uWaveGestureLibrary_X	26.2%	139%
ElectricDevices	57.6%	292%	uWaveGestureLibrary_Z	25.6%	132%

Using a nonlinear exponential regression curve fitting method, the following equation can be used to estimate the processing time gain based on the repetition rate:

$$Speed\ \ Gain = e^{4.6348*(repetition\ \ rate)^2} \tag{11}$$

In Fig. 11, the comparison between BDTW and DTW in terms of classification error is presented. This figure shows that BDTW is a very close approximation of DTW. In many cases, (63 out of 85) both methods result in same classification error. In some case, (9 out of 85) BDTW classification error is slightly less than DTW and in 13 out of 85 datasets DTW has slightly less error. Using BDTW for time series with high repetition level (repetition rate), will have almost same accuracy but shorter processing time than traditional DTW.

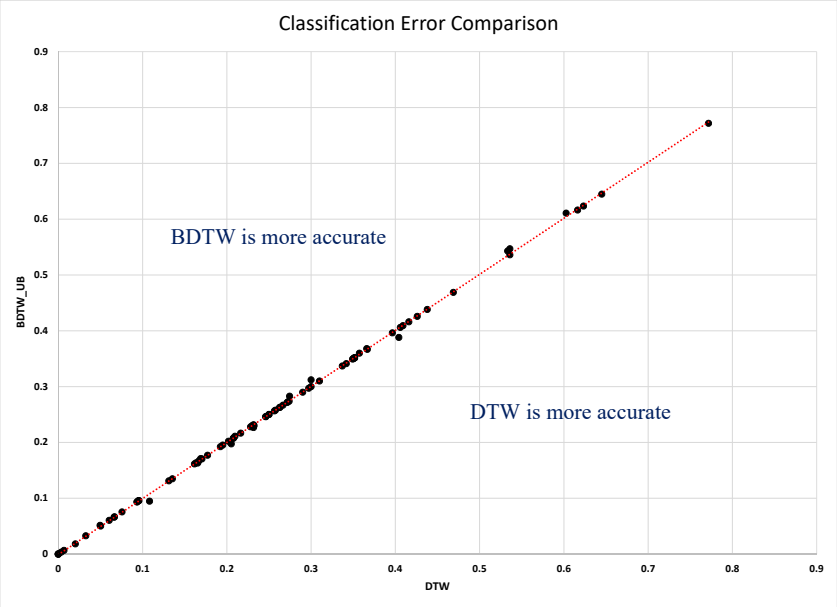


Fig. 11 The comparison BDTW and DTW in terms of classification error for 85 time series datasets in UCR archive

4.4 Using BDTW_UB on Top of APCA as an Approximation Method for Any-Valued Time Series

From each datasets in UCR Archive 20% of all-pairwise time series are randomly selected and for each time series pair, true DTW distance and approximation distances are calculated. To make the processing time linear, for calculating DTW approximation we use square root of n as the number of segments in time series representation (where n is the length of time series). Using formula (10), we calculate the error between true DTW distance and approximation methods. The approximation methods that we compare are abstraction (using PAA-DTW and projection) and APCA-BDTW_UB. The result is presented in Fig. 12. The x axes is the datasets and the y axes is the mean of error. This Figure shows that in most datasets (61 out of 75) APCA-BDTW_UB outperforms PAA-DTW based approximation method (PAA-DTW and projection).

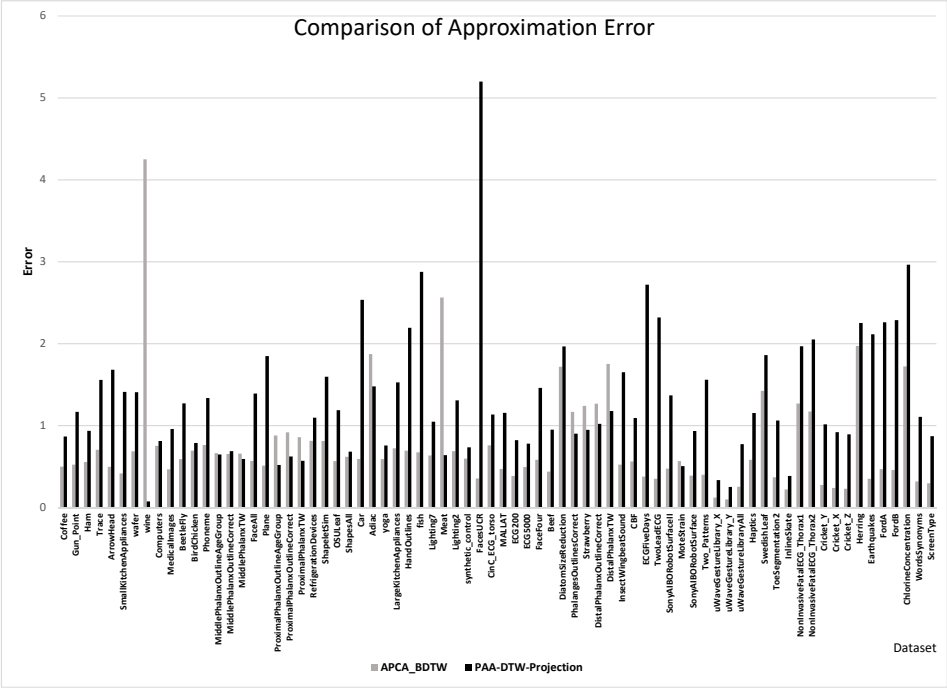


Fig. 12 Comparison of approximation error between APCA_BDTW and PAA-DTW-Projection

4.5 BDTW_LB: a New Lower Bound to Increase the Efficiency of DTW

BDTW_LB can be used as lower bound to accelerate the similarity search when we want to calculate the exact DTW. In Table 5, the power of BDTW_LB, in speed up of DTW, is compared with Kim, Yi and Keogh lower bounds on 6 datasets with highest repetition rate in UCR Archive.When a speed up is less than 1, it means using lower bound does not help in reducing the classification processing time. In smallkitchenAppliances dataset, using BDTW_LB, results in more than 16 time faster similarity search than using traditional DTW.

BDTW_LB performs better than both Kim_LB, Yi_LB and Keogh_LB in datasets with relatively high repetition rate. Usually, as repetition rate of a dataset increases, BDTW_LB performance increases. Here Keogh_LB performance is evaluated with warping windows of 10%.

Table 5 Comparison the efficiency of BDTW lower bound with Kim, Yi and Keogh lower bounds on time series with high repetition rate

Dataset	Repetition Rate	Kim_LB speed-up	Yi_LB speed-up	Keogh_LB speed-up	BDTW_LB speed-up
SmallKitchenAppliances	86%	0.89	1.25	1.49	16.87
LargeKitchenAppliances	82%	0.99	1.36	1.72	10.84
ScreenType	78%	0.95	2.13	1.82	5.24
Computers	71%	0.88	1.65	1.73	4.59
Earthquakes	68%	0.67	0.59	0.63	2.32
RefrigerationDevices	64%	0.88	1.03	0.88	2.25

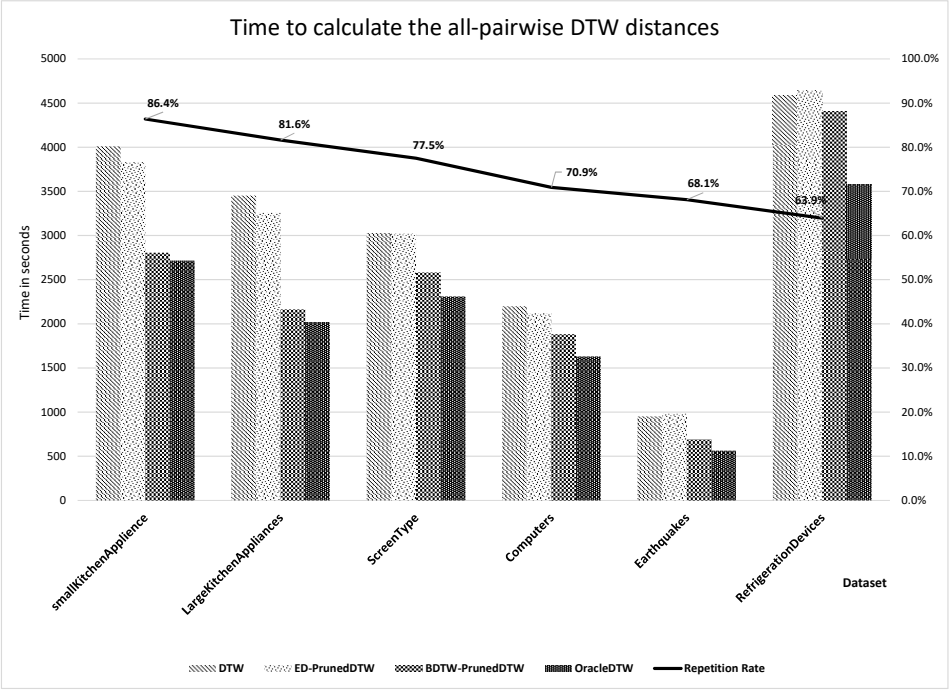


Fig. 13 Comparison processing time to calculate the all-pairwise DTW distances using DTW without pruning, DTW pruned by ED as the upper bound, DTW pruned by BDTW_UB as the upper bound and OracleDTW

4.6 Using BDTW_UB as a New Upper Bound to Speed Up All-Pairwise DTW Matrix Calculation

PrunedDTW code is used to measure the performance of BDTW_UB as the applied upper bound for pruning unpromising alignments in all-pairwise DTW matrix calculation [10]. In Fig. 13, the processing time of DTW without pruning, DTW pruned by ED (as the upper bound), DTW pruned by BDTW_UB (as the upper bound) and OracleDTW are compared on 6 datasets with high repetition rate. In this figure, the line shows the repetition rate of the datasets and bars show the processing time. This figure shows that for high repetition rate problems, BDTW_UB outperforms DTW and DTW pruned by ED. The higher repetition rate in a problem results in the better performance of BDTW_UB. When repetition rate increases to higher than 0.8, BDTW_UBs performance gets very close to OracleDTW's performance (the best possible performance of PruedDTW).

The processing time of BDTW_UB is summation of two times: the time for calculating BDTW_UB distance and the time for pruning and calculating DTW matrix. Since BDTW_UB is a close approximation of exact DTW, its required time for pruning and matrix calculation is very close to OracleDTW. However the required time for calculating BDTW_UB distance is the overhead that adds to processing time. When repetition rate is high enough this overhead will be justified in total processing time because BDTW_UB distance calculation will be fast. In some problems (for example ScreenType, Earthquakes and RefrigerationDevices) using ED as the upper bound for pruning does not help in reduc-

Table 6 Comparison processing time and accuracy of DTW, BDTW, Constrained DTW and Constrained BDTW

	Repetition	Accuracy				Processing time (s)				CBDTW Speed-up ratio		
Dataset	Rate	DTW	CDTW	BDTW	CBDTW	DTW	CDTW	BDTW	CBDTW	DTW	CDTW	BDTW
SmallKitchenAppliances	86.4%	64%	61%	64%	73%	715.4	770.7	20.3	16.0	44.7	48.2	1.3
LargeKitchenAppliances	81.6%	79%	71%	80%	72%	718.9	571.4	37.6	22.8	31.5	25.1	1.6
ScreenType	77.5%	40%	43%	39%	39%	647.7	514.8	45.6	27.7	23.4	18.6	1.6
Computers	70.9%	70%	64%	69%	65%	312.9	297.5	37.1	24.7	12.6	12.0	1.5
Earthquakes	68.1%	74%	74%	74%	74%	124.7	118.5	16.7	11.1	11.2	10.6	1.5
RefrigerationDevices	63.9%	46%	46%	46%	51%	758.7	1,016.3	150.2	86.0	8.8	11.8	1.7
ElectricDevices	57.6%	60%	62%	61%	64%	6,172.2	4,568.5	2,111.5	1,087.6	5.7	4.2	1.9
wafer	48.8%	98%	98%	98%	100%	1,120.0	375.4	543.5	278.5	4.0	1.3	2.0
FaceFour	47.2%	83%	83%	83%	88%	2.7	3.0	1.4	0.5	5.3	5.9	2.7
Two-Patterns	34.4%	100%	100%	100%	100%	724.8	317.4	478.5	195.1	3.7	1.6	2.5
uWaveGestureLibrary-Y	31.8%	63%	66%	63%	70%	2,496.3	967.2	1,686.3	740.1	3.4	1.3	2.3
uWaveGestureLibraryAll	28.2%	89%	89%	91%	95%	26,655.0	28,464.3	22,290.9	13,844.7	1.9	2.1	1.6
uWaveGestureLibrary-X	26.2%	73%	74%	73%	80%	2,688.6	1,064.9	1,932.7	850.1	3.2	1.3	2.3
uWaveGestureLibrary-Z	25.6%	66%	67%	66%	70%	2,335.6	925.1	1,774.9	780.6	3.0	1.2	2.3

ing the processing time. Because for these problems ED is fast to calculate but, it does not provide a powerful and useful pruning level.

4.7 CBDTW_LB: Constrained Block DTW

In Table 6, the processing time and accuracy of DTW, BDTW, Constrained DTW (CDTW) and Constrained BDTW (CBDTW) with warping window of 10% of the length of time series are compared on the datasets with repetition rate of higher than 25%. The last 3 columns of the table shows the speed up of CBDTW in respect to DTW, CDTW and BDTW. The results show that for all time series with repetition rate of higher than 25%, CBDTW is faster than DWT, CDTW and BDTW with better or close level of accuracy.

5 Conclusion

This paper introduces Blocked Dynamic Time Warping (BDTW) as a new similarity measure which works on run-length encoded time series representation. BDTW takes advantage of values repetition in time series to shrink the size of DTW matrix for a reduction of processing time. BDTW algorithms offer an upper bound and a lower bound of DTW. BDTW upper bound is a close approximation of exact DTW and significantly reduces the processing time of calculations on time series with high levels of values repetition.

The Combination of BDTW upper bound and APCA provides a close approximation of DTW distance, even for time series with low levels of values repetition or without any vale repetition. APCA-BDTW outperforms the approximation method based on PAA-DTW-Projection. BDTW can also be used as a pruning technique to accelerate calculation of the exact DTW. BDTW lower bound can be used for pruning of unpromising candidates in similarity search and it performs better Kim and Yi lower bounds for time series with high levels of values repetition. BDTW upper bound can also be used for pruning of unpromising alignments in all-pairwise DTW calculation and performs better than ED (as the UB) for time series with high levels of values repetition.

In Constrained BDTW Algorithm, a warping constrant is used to limit the search area in BDTW matrix and for time series with high repetition rates, Constrained BDTW is faster than Constrained DTW with similar or better accuracy.
BDTW is also extendable to multidimensional time series warping. Investigating and ana-

lyzing the specification of multidimensional time series warping can be considered as the future work of this paper.

References

1. V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, "A multiresolution symbolic representation of time series." *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE*, 2005.
2. D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.
3. A. P. Shanker and A. Rajagopalan, "Off-line signature verification using dtw," *Pattern recognition letters*, vol. 28, no. 12, pp. 1407–1414, 2007.
4. J. B. Kruskal and M. Liberman, "The symmetric time-warping problem: from continuous to discrete," *Time warps, string edits and macromolecules: The theory and practice of sequence comparison*, pp. 125–161, 1983.
5. J. Aach and G. M. Church, "Aligning gene expression time series with time warping algorithms," *Bioinformatics*, vol. 17, no. 6, pp. 495–508, 2001.
6. Z. Bar-Joseph, G. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon, "A new approach to analyzing gene expression time series data," in *Proceedings of the sixth annual international conference on Computational biology*. ACM, 2002, pp. 39–48.
7. D. Gavrila, L. Davis *et al.*, "Towards 3-d model-based tracking and recognition of human movement: a multi-view approach," in *International workshop on automatic face-and gesture-recognition*. Citeseer, 1995, pp. 272–277.
8. T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003.
9. A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The Great Time Series Classification Bake Off: a Review and Experimental Evaluation of Recent Algorithmic Advances,"*Data Mining and Knowledge Discovery*, 2016.
10. D. F. Silva and G. E. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 837–845.
11. A. Mueen, N. Chavoshi, N. Abu-El-Rub, H. Hamooni, and A. Minnich, "Awarp: Fast Warping Distance for Sparse Time Series." in *Data Mining (ICDM)* IEEE, 2016, pp. 350–359.
12. E. Keogh and T. Folias, "The UCR time series data mining archive," *Computer Science & Engineering Department, University of California, Riverside, CA. http://www. cs. ucr. edu/eamonn/TSDMA/index. html*, 2002.
13. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.
14. X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. "Experimental comparison of representation methods and distance measures for time series data". *Data Mining and Knowledge Discovery*, 26(2):275 309, pp. 1–35, 2013.
15. H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures", *PVLDB* 1(2): 1542-1552, 2008
16. S. Makonin. "Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014". *Sci. Data 3:160037 doi: 10.1038/sdata.2016.37* (2016).
17. D. Murray and L. Stankovic. *Refit: Electrical load measurements.* http://www.refitsmarthomes.org/
18. E Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 151–162, 2001.
19. H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.
20. E. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, and P. Protopapas, "Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures," *The International Journal on Very Large Data Bases*, vol. 18, no. 3, pp. 611–630, 2009.
21. S. W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases." IEEE, 2001, pp. 607–614.
22. A. Fu, E. Keogh, L.Y. Lau, "Scaling and time warping in time series querying." *The International Journal on Very Large Data Bases*, vol. 17, no. 4, pp. 899–921, 2008.

23. T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 262–270.
24. G. E. Batista, X. Wang, and E. Keogh, "A complexity-invariant distance measure for time series." in *SDM*, vol. 11. SIAM, 2011, pp. 699–710.
25. S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.
26. E. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 285–289.
27. S. Chu, E. Keogh, D. M. Hart, M. J. Pazzani , "Iterative deepening dynamic time warping for time series." in *SDM*. SIAM, 2002, pp. 195–212.
28. F. Petitjean, "Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm," *Knowledge and Information Systems*, 2016, pp. 1–26.
29. E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping." *Knowledge and Information Systems*, 2005, pp. 358–386.
30. R. Tavenard and L. Amsaleg, *"Improving the efficiency of traditional DTW accelerators." Knowledge and Information Systems*, 2015, pp. 215–243.
31. A. Sharabiani, H.Darabi, A.Rezaei, S.Harford, H.Johnson and F. Karim, *"Efficient Classification of Long Time Series by 3-Dimensional Dynamic Time Warping." IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47 Issue: 8. doi:10.1109/TSMC.2017.2699333.
32. C.A. Ratanamahatana and E. Keogh, "Three Myths about Dynamic Time Warping Data Mining. *In Proceedings of the SIAM International Conference on Data Mining. SDM 05. SIAM*, 2005, pp. 506-510.