CTU Journal of Innovation and Sustainable Development
ISSN 2615-9422 | e-ISSN 2815-5602

DOI:10.22144/ctujoisd.2023.029

Similarity join over multiple time series under Dynamic Time Warping

Bui Cong Giao*

Faculty of Electronics and Telecommunications, Saigon University, Ho Chi Minh City, Viet Nam

**Corresponding author (bcgiao@sgu.edu.vn)*

Article info.

Received 11 Jul 2023
Revised 10 Sep 2023
Accepted 23 Sep 2023

Keywords

Data normalization, Dynamic Time Warping, similarity join

ABSTRACT

Similarity join over multiple time series is an interesting task of data mining. This task aims at identifying couples of similar subsequences from multiple time series and the two subsequences might have any length and be at any position in the time series. However, the task is extremely challenging since the computational time to search for couples of similar subsequences from two time series is very large. Moreover, the task needs to normalize two subsequences before conducting a distance measure on the normalized subsequences to consider the similar degree of the original subsequences. To address the problem, this paper proposes a method of similarity join over two time series under Dynamic Time Warping (DTW), supporting z-score normalization. The proposed method utilizes both a suite of state-of-the-art techniques for computing the DTW distance and a technique of incremental z-score normalization to reduce the computational costs. The method employs multithreading to improve runtime performance. If similar subsequences from two time series may not pair up because they are too far apart, the method might use a sliding window to constrain a scope for coupling similar subsequences. The experiments on the proposed method show that the method could return similar subsequences quickly and incur no false dismissals.

1. INTRODUCTION

Similarity join or subsequence join over multiple time series is a major task in time-series data mining, which aims at identifying couples of similar subsequences from the time series under a certain measure. There are numerous methods of similarity join over multiple time series under the Euclidean distance (Lian & Chen, 2009; Chatzigeorgakidis et al., 2018; Yeh et al., 2018; Zhu et al., 2018; Wang et al., 2019; Yeh et al., 2022), or the Dynamic Time Warping (DTW) distance (Chen et al., 2009; Lin & McCool, 2010; Vinh & Anh, 2016), or the Fréchet distance (Ding et al., 2008), or Pearson's correlation coefficient (Mueen et al., 2014; Mollah et al., 2021). It is noteworthy that the task is very complicated because two similar subsequences constituting a combinable couple might have any length, and be at any position in the time series. A naïve process to conduct the task would consider all couples of subsequences from the time series to evaluate the similar degree of each two subsequences under a predefined distance measure, thereby incurring huge computational costs. In addition, to obtain accurate results, the distance measure should be conducted on normalized subsequences rather than their original ones. However, the operation of data normalization incurs additional computational costs to the task.

The problem of similarity join over multiple time series takes place in a wide spectrum of applications including analysing stocks (Lian & Chen, 2009), identifying spatial proximity and time series similarity over geolocated time series (Chatzigeorgakidis et al., 2018), synthesizing

motion (Ding et al., 2008; Chen et al., 2009), and searching for similarly fluctuant patterns in exchange rates (Mueen et al., 2014). This evidence shows that the intricate problem is extremely interesting, drawing much attention from researchers specializing in data mining on time series. For this reason, many research studies (Ding et al., 2008; Chen et al., 2009; Lian & Chen, 2009; Lin & McCool, 2010; Mueen et al., 2014; Vinh & Anh, 2016; Chatzigeorgakidis et al., 2018; Yeh et al., 2018; Zhu et al., 2018; Wang et al., 2019; Mollah et al., 2021; Yeh et al., 2022) have been investigating solutions to the problem.

In time-series data mining, the DTW distance (Berndt & Clifford, 1994) is suitable for matching two time series representing multimedia data such as audio and video. For this reason, DTW is often used in applications involving multimedia data processing, such as speech processing and gait recognition although the distance measure is of high time complexity. As for the data normalization of time series, there are two types for the operation: *min-max* and *z*-score. The latter is preferred in data mining on time series since *z*-score normalization preserves the shape of a normalized sequence more closely that of the original sequence.

After referring to the research studies (Ding et al., 2008; Chen et al., 2009; Lian & Chen, 2009; Lin & McCool, 2010; Mueen et al., 2014; Vinh & Anh, 2016; Chatzigeorgakidis et al., 2018; Yeh et al., 2018; Zhu et al., 2018; Wang et al., 2019; Mollah et al., 2021; Yeh et al., 2022), we note that so far, no methods of similarity join over multiple time series under DTW, supporting *z*-score normalization have been introduced yet. It is likely that due to the huge computational costs deriving from computing both the DTW distance and *z*-score normalization, the researchers used other ways to solve the problem of similarity join over multiple time series at computational costs of rationality and inexpensiveness. Motivated by the above observation, this paper proposes a method of similarity join over two time series under DTW, supporting *z*-score normalization. The requirements for the proposed method are no false dismissals and quick responses to inquiries of the similarity join. The experimental evaluation of the method is drawn from experiments on the method with real time series. Concretely, the method is evaluated in terms of accuracy and execution time.

The main contribution in this paper is a novel method of similarity join over multiple time series under DTW, supporting *z*-score normalization. The proposed method employs the following accelerative techniques:

(i). A suite of state-of-the-art techniques for computing the DTW distance and a technique of incremental *z*-score normalization to decrease computational costs, and
(ii). Multithreading to subdivide the process performing the similarity join into many threads executing this task simultaneously, thereby shortening execution time, and
(iii). A sliding window to constrain a scope for coupling similar subsequences from two time series, thereby reducing the number of the DTW distance calculations.

The rest of the paper is organized as follows. Section 2 provides notations and definitions necessary for the problem. Section 3 presents supporting techniques for the proposed method. Section 4 proposes the method. Section 5 gives an overview of the experimental evaluation. Finally, Section 6 provides conclusion and future work directions.

2. NOTATIONS AND DEFINITIONS

The problem of similarity join over multiple time series is formally posed under the following definitions.

Definition 1. (*Time series*) Time series X is a series of real values collected in chronological order. For simplicity, $X = \{x_1, x_2, \ldots, x_n\}$ for $i = 1 \ldots n$ and $x_i \in \mathbb{R}$; hence, $|X| = n$. Besides, x_i is regarded as the data point ith of X.

A subsequence C of X is a chronologically ordered subset of X. That means $C=\{x_p, x_{p+1}, \ldots, x_q\}$ for $1 \leq p$ and $q \leq n$. For the sake of simplicity, let X_p^q denote $\{x_p, x_{p+1}, \ldots, x_q\}$; therefore, $C = X_p^q$. After X_p^q is normalized, we have its normalized time series; that is $NX_p^q = \{\overline{x_p}, \overline{x_{p+1}}, \ldots, \overline{x_q}\}$ where $\overline{x_\iota}$ is the normalized data point of x_i, and $i = p \ldots q$.

Table 1 depicts some notations, which are derived from two time series X and Y, necessary for the next definitions.

Definition 2. (*Similarity join over multiple time series*) Given a distance threshold $\varepsilon = \alpha \times \ell(l_{Xab}, l_{Ypq})$, the problem of *similarity join over multiple time series* is to search for all the couples of subsequences (X_a^b, Y_p^q) such that $dist(NX_a^b, NY_p^q) \leq \varepsilon$. If the inequality is satisfied, (X_a^b, Y_p^q) is a *cross-similar* couple.

Table 1. Notations

Notation	Meaning
minLength	The minimum length of interested subsequences
X_a^b, X_c^d	Two subsequences of *X*
Y_p^q, Y_u^v	Two subsequences of *Y*
l_{Xab}	The length of X_a^b, i.e., $l_{Xab} = \|X_a^b\| = b - a + 1$
l_{Xcd}	The length of X_c^d, i.e., $l_{Xcd} = \|X_c^d\| = d - c + 1$
l_{Ypq}	The length of Y_p^q, i.e., $l_{Ypq} = \|Y_p^q\| = q - p + 1$
l_{Yuv}	The length of Y_u^v, i.e., $l_{Yuv} = \|Y_u^v\| = v - u + 1$
NX_a^b, NY_p^q	The normalized time series of X_a^b, and Y_p^q, respectively
dist(x, y)	A function measures the distance between two subsequences *x* and *y*
α	A tuning parameter
$\ell(l_x, l_y)$	A function justifies that the similarity evaluation of two subsequences *x* and *y*, is significantly depended on their lengths, i.e., l_x and l_y

Definition 3. (*Overlap elimination*) The problem of similarity join over multiple time series should eliminate cross-similar couples which overlap enormously together not to have trivial results.

Two time-series subsequences overlap enormously together if they share a large portion of the common data points. Let β denote a ratio of enormous overlap, $0 < \beta \leq 1$. Assume that (X_a^b, Y_p^q) and (X_c^d, Y_u^v) are two cross-similar couples. X_a^b and X_c^d overlap enormously if they share at least $\lceil \beta \times l_{Xab} \rceil$ data points of X_a^b or $\lceil \beta \times l_{Xcd} \rceil$ data points of X_c^d. If X_a^b and X_c^d overlap enormously, and Y_p^q and Y_u^v does the same, the two cross-similar couples overlap enormously together. To obtain meaningful results, we must rule out one of the two couples in accordance with the following rules.

1. The first couple is removed if $l_{Xab} + l_{Ypq} < l_{Xcd} + l_{Yuv}$; otherwise, the second one is eliminated. The couple pruned off is regarded as a trivial result.
2. If $l_{Xab} + l_{Ypq} = l_{Xcd} + l_{Yuv}$, we check $dist(NX_a^b, NY_p^q) < dist(NX_c^d, NY_u^v)$. If the inequality is satisfied, (X_c^d, Y_u^v) is removed, and vice versa. In these ways, i.e., Rules 1 and 2, we can find out one cross-similar couple whose total length is largest or normalized subsequences are closer.
3. A frequent case of the enormous overlap is that (X_a^b, Y_p^q) covers (X_c^d, Y_u^v) entirely; that is $X_a^b \supseteq X_c^d$ and $Y_p^q \supseteq Y_u^v$. Accordingly, the second couple is eliminated.

3. SUPPORTING TECHNIQUES

The proposed method employs three techniques to improve runtime performance: UCR-DTW (Rakthanmanon et al., 2012) to check the similarity of two time series under DTW, incremental *z*-score normalization (Giao & Anh, 2016), and multithreading.

3.1. UCR-DTW

Dynamic Time Warping (DTW) is a pervasive distance measure for two time series. The distance measure may map one data point of a time series to many data points of another to search for a minimum distance. In this way, the sum of distances of such couples of data points is minimum and is the DTW distance between two time series. It is obvious that DTW outperforms the Euclidean measure in terms of capability of detecting the similarity of two time series despite having a phase difference between them. Furthermore, DTW can work over two time series of different lengths. DTW is very suitable for multimedia data, such as text, voice signal, and video. Yet, DTW is of high time complexity, $O(n^2)$.

UCR-DTW is a suite of state-of-the-art techniques to compute the DTW distance between two time series. UCR-DTW could alleviate the high time complexity of DTW using the Sakoe-Chiba band with a width of *w* to constrain the scope for mapping from one data point of a time series to many data points of another in the course of searching for a minimum distance. This is suitable for reality because one data point of a time series may hardly map to one data point, which is too far, of another. Moreover, UCR-DTW employs cheap-to-compute lower bounding functions to help relieve the high computational cost of DTW. The lower bounding functions enable UCR-DTW to prune off the dissimilar time series early. The evidence is proven as follows. Let *LB*(*C*, *Q*) be a lower-bound function of two time series *C* and *Q*, and *DTW*(*C*, *Q*) be the DTW distance between the two time series, we have *LB*(*C*, *Q*) ≤ *DTW*(*C*, *Q*). Given ε be a distance

threshold, if $LB(C, Q) > \varepsilon$ is satisfied, C is certainly dissimilar to Q due to $DTW(C, Q) > \varepsilon$.

The lower bounding functions in UCR-DTW are LB_{Kim} (Kim & Park, 2001) of the time complexity $O(1)$, LB_{Keogh} (Keogh & Ratanamahatana, 2004) of $O(n)$, and reversed LB_{Keogh} of $O(n)$. The lower bounding functions are arranged in UCR-DTW such that front lower bounding functions of lower time complexities eliminate most couples of dissimilar time series. If the three lower bounding functions cannot filter out couples of dissimilar time series, the direct DTW computation will be performed to confirm whether the time-series couples are similar. Since UCR-DTW employs the above speedup techniques in the conjunctive manner, computing DTW is of early abandoning. As a result, the computational time of DTW using UCR-DTW is reduced dramatically.

UCR-DTW needs to build the envelope of a time series so that LB_{Keogh} in UCR-DTW can operate. The envelope of time series X comprises two time series U and L standing for upper and lower bounds of X. Given the Sakoe-Chiba band whose width is w, the data points of U and L are computed as below.

$$\begin{aligned} u_i &= \max(x_{i-w} : x_{i+w}) \\ l_i &= \min(x_{i-w} : x_{i+w}). \end{aligned} \tag{1}$$

Note that UCR-DTW uses the one-pass method of Lemire (Lemire, 2009) to construct the envelop of a time series.

3.2. Incremental *z*-score normalization

The technique of incremental *z*-score normalization is presented as follows.

Z-score normalization maps a value x of time series $X = \{x_1, x_2, \ldots, x_n\}$ to x_{norm} by computing

$$x_{\text{norm}} = \frac{x - \mu}{\sigma} \tag{2}$$

where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3}$$

and

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \mu^2. \tag{4}$$

Let define

$$\hat{x} = \sum_{i=1}^{n} x_i^2. \tag{5}$$

Equation (4) can be expressed as

$$\sigma^2 = \frac{\hat{x}}{n} - \mu^2. \tag{6}$$

Henceforth we regard μ and σ as *z*-score coefficients of time series.

Assume that X slides forward, thereby causing $X = \{x_2, x_3, \ldots, x_{n+1}\}$. The new *z*-score coefficient μ_{new} of X is computed as below.

$$\begin{aligned} \mu_{\text{new}} &= \mu + \frac{x_{n+1} - x_1}{n} \text{ and} \\ \hat{x}_{\text{new}} &= \hat{x} + x_{n+1}^2 - x_1^2. \end{aligned} \tag{7}$$

σ_{new} is then derived from (4) and (7).

$$\sigma_{\text{new}}^2 = \frac{\hat{x}_{\text{new}}}{n} - \mu_{\text{new}}^2. \tag{8}$$

In this way, we do not need to compute the new *z*-score coefficients of X from scratch. Thanks to this incremental computation, the time complexity of *z*-score normalization declines from $O(n)$ to $O(1)$ when X slides forward step by step with each data point.

3.3. Multithreading

Multithreading is the ability of computing systems to handle simultaneously many tasks of a job executed by a process. To fulfil this ability, the process must create multiple threads of concurrent execution and each execution thread takes charge of a task. After all execution threads have completed, their results are aggregated to return the ultimate results for the process.

It is worth noting that multithreading can be easily implemented in state-of-the-art processors and programming languages. Yet, to employ the advantages, the tasks should be less dependent on one another, and their loads should be distributed evenly to optimize the performance of processors. Furthermore, the number of execution threads must be approximate to that of threads of processors. For example, CPU AMD Ryzen 3 3250U has 2 cores and 4 threads so the number of execution threads of a program using multithreading should be approximate 4. If the number of execution threads is slightly larger than that of threads of processors, the program could perform a little faster, but not significantly. If there are too many execution threads, the program will work slower because the computing system needs to spend more time switching between the contexts of the execution threads.

The problem of similarity join over time series might be subdivided into many execution threads. Each execution thread searches for cross-similar couples within a range of lengths. These ranges of lengths are separate and predefined by data analysts.

4. PROPOSED METHOD

Given *minLength*, α, and $\ell(l_x, l_y)$ described in Table 1, *w* being the width of the Sakoe-Chiba band, and β being the ratio of enormous overlap, the proposed method of similarity join over multiple time series under DTW is illustrated by Algorithm SJ-DTW.

There are some things to explain Algorithm SJ-DTW as follows.

– In the beginning, the total data points of *X* and *Y*, *n* and *m*, are retrieved at Lines 1 and 2, respectively. The length region of the subsequences of *X* and *Y*, which are valid for the task of similarity join, is from *minLength* to *maxLength*. In the algorithm, *maxLength* is the minimum value of *n* and *m*. Note that *maxLength* might be an arbitrary value from *minLength* to min(*n*, *m*). The smaller *maxLength* is, the faster the task of similarity join performs because the number of searches done by UCR-DTW at Line 17 is fewer.

Algorithm SJ-DTW(*X* and *Y* are two time series)

```
1.  n ← |X|
2.  m ← |Y|
3.  maxLength ← min(n, m)
4.  Compute the z-score coefficients, μ and σ, of X_1^minLenth and Y_1^minLenth
5.  Construct the envelopes of X and Y
6.  for len ← minLength to maxLength do
7.      if len > minLength then
8.          Update μ and σ of X_1^len and Y_1^len incrementally
9.      for i ← 1 to n − len + 1 do
10.         if i > 2 then
11.             Update μ and σ of X_i^(i+len−1) incrementally
12.         NX_i^(i+len−1) ← Normalize X_i^(i+len−1) using μ and σ
13.         Sort the indices of data points of X_i^(i+len−1) based on the absolute
values of data points of NX_i^(i+len−1)
14.         for j ← 1 to m − len + 1 do
15.             if j > 2 then
16.                 Update μ and σ of Y_j^(j+len−1) incrementally
17.             if UCR-DTW(NX_i^(i+len−1), NY_j^(j+len−1)) ≤ α × ℓ(l_x, l_y) then
18.                 if couple (X_i^(i+len−1), Y_j^(j+len−1)) does not overlap enormously with
cross-similar couples found beforehand then
19.                     (X_i^(i+len−1), Y_j^(j+len−1)) is a cross-similar couple
```

– The *z*-score coefficients of the first subsequences of *X* and *Y* are computed at Line 4. Henceforth, the *z*-score coefficients of the subsequent subsequences of *X* and *Y* are updated incrementally at Lines 8, 11, and 16 to ease the computational costs of *z*-score normalization.

– Using the width *w* of the Sakoe-Chiba band, the algorithm constructs the envelopes of *X* and *Y* at Line 5 because LB_{Keogh} and reversed LB_{Keogh}, the two lower bounding functions used in UCR-DTW, need them to work. The lower bounding functions work with the descending order of the absolute values of data points of $NX_i^{i+len-1}$ rather than the original order of the data points of the normalized subsequence so the absolute values are sorted and then the indices of data points after sorting are recorded at Line 13. LB_{Keogh} and reversed LB_{Keogh} work with such order of the data points of the normalized subsequence in order to accelerate UCR-DTW.

– At Line 17, $NY_j^{j+len-1}$ is computed on the fly in performing UCR-DTW along with the *z*-score

coefficients of $Y_j^{j+len-1}$ computed incrementally at Line 16. Moreover, because UCR-DTW works only with two time series of the same length, the function $\ell(l_x, l_y)$ has $l_x = l_y = len$.

– β is used to check whether two couples overlap enormously together at Line 18.

– The algorithm is of brute force approach that searches for cross-similar couples from all subsequences couples deriving from *X* and *Y*, not incurring false dismissals.

– Because of the negligible computational cost of UCR-DTW, the time complexity of Algorithm SJ-DTW is $O(n^3)$.

There are two ways to accelerate the proposed method:

1. Multithreading is used to improve the runtime performance of the method. Assume that the computing system can create *k* execution threads to handle simultaneously similarity join over two time series *X* and *Y*. The length region [*minLength*, *maxLength*] is subdivided into *k* separate subregion. The size of each subregion is $e = \left\lceil \frac{maxlength - minlength + 1}{k} \right\rceil$ and the final subregion might be less than *e* elements. Execution thread *i* takes charges of the subregion of [$minlength_i$, $maxlength_i$]. Therefore, Lines 6 and 7 of Algorithm SJ-DTW for execution thread *i* changes as below.

```
6.for len ← minLength_i to maxLength_i
    do
7.    if len > minLength_i then
```

   Such *k* execution threads can theoretically speed up the method to *k* times.

2. It is worth noting that in fact a subsequence $X_i^{i+len-1}$ could hardly similarly join with a subsequence $Y_j^{j+len-1}$ where the two subsequences are too far apart. That means if $(X_i^{i+len-1}, Y_j^{j+len-1})$ is a cross-similar couple, then we have $|i - j| \leq r$ where *r* is the predetermined half width of a window which constrains possibility to couple two similar subsequences deriving from *X* and *Y*. Therefore, for each value of *i* at Line 9, there is a constraining window in *Y* so that $Y_j^{j+len-1}$ slides within this window. Line 14 manifests the constraining window as follows.

```
14.        for j ← max(1, i − r) to
       min(m − len + 1, i + r)  do
```

   As a result, the number of invoking UCR-DTW at Line 17 slumps significantly.

5. EXPERIMENTAL EVALUATION

The section represents experiments on the proposed method to empirically evaluate the method in terms of accuracy and execution time. The experiments were carried out with a PC of Intel Core i7 6600U, and 16GB RAM. Note that the CPU has 2 cores and 4 threads. The method was implemented in Microsoft C#.

The parameters are set in the proposed method as follows. The width of the Sakoe-Chiba band in UCR-DTW is $w = 5$. The ratio of enormous overlap is $\beta = 0.9$. With respect to the distance threshold, $\varepsilon = \ell(l_x, l_y) \times \alpha$, we use $\ell(l_x, l_y) = \sqrt{\frac{l_x + l_y}{2}} = \sqrt{l_x}$. Regarding α and *minLength*, they are specifically determined for each experiment to reply to practical inquiries and be suitable for the concise visualization of cross-similar couples. Moreover, let $(i : j)$ denote a subsequence from index *i*th to index *j*th of a time series to clarify the presentation of the experiments.

The experiments on the proposed method are conducted with two pairs of real time series. The first is two time series of exchange rates, and the second is two time series of biosignal. These time-series datasets may be downloaded from (Giao, Time-series datasets, 2022).

5.1. Exchange rates of Australian Dollar (AUD) and Canadian Dollar (CAD) against United States Dollar (USD)

The two time series, the monthly exchange rates of *USD/AUD* and *USD/CAD* from January 1971 to June 2023, are collected from (Pele, 2023). Each time series has 530 data points. The tuning parameter α is 0.1 in the experiment. The inquiry, which is posed here for similarity join over *USD/AUD* and *USD/CAD*, is to search for similar fluctuations of the two exchange rates which happen during periods lasting at least one year, i.e., *minLength* = 12. After the experiment has finished, the primary results are obtained as follows. There are 4 cross-similar couples; however, there is one case in which two cross-similar couples enormously overlap together. The trivial couple in this case is thus eliminated.

Table 2. Details of the resulting couples

Couple	*USD/AUD*		*USD/CAD*		Length	ED	DTW	ε
	Subsequence Start	End	Subsequence Start	End				
1	(122:153) Feb-81	Sep-83	(145:176) Jan-83	Aug-85	32	0.967	0.564	0.566
2	(363:417) Mar-01	Sep-05	(180:234) Dec-85	Jun-90	55	1.347	0.741	0.742
3	(378:395) Jun-02	Nov-03	(378:395) Jun-02	Nov-03	18	0.707	0.415	0.424

Table 2 reveals the details of the three resulting couples identified by the proposed method. For the first cross-similar couple, the table indicates that the monthly exchange rates of *USD/AUD* from February 1981 to September 1983 and those of *USD/CAD* from January 1983 to August 1985 fluctuated nearly the same. The period of the event lasted 32 months. The DTW distance of the two normalized subsequences, which are (122:153) and (145:176), corresponding to the period is less than the distance threshold ε, i.e., $0.564 < 0.1 \times \sqrt{32} \approx 0.566$. However, if the Euclidean distance (ED) is used, the couple is not cross similar because of 0.967 > 0.566. The two remaining cross-similar couples are reasoned likewise. The result of the experiment also implies that if similarity join over *USD/AUD* and *USD/CAD* with *minLength* = 12 and $\varepsilon = 0.1 \times \sqrt{l_x}$ is conducted under ED, no cross-similar couples are found.

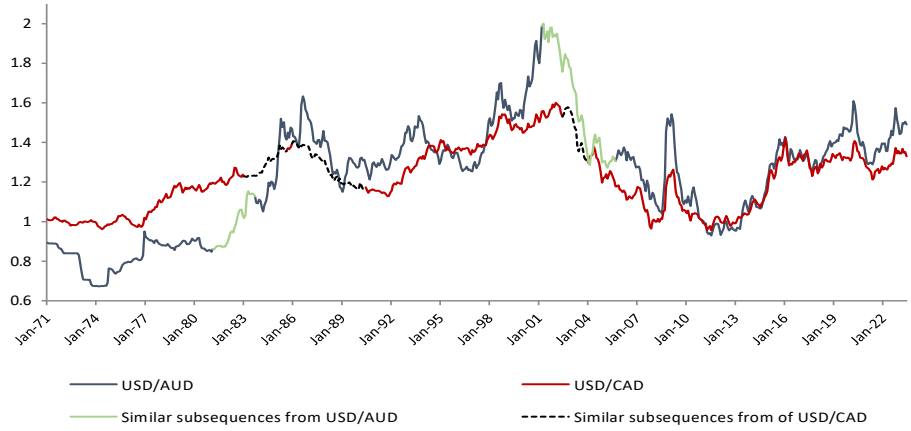

Figure 1. Visualization of *USD/AUD* and *USD/CAD* and their similar subsequences

Figure 1 illustrates the two time series and the similar subsequences detected by the proposed method. It is difficult to view the cross-similar couples because their subsequences have different altitudes and often occur at different positions. Figure 2 enables viewers to see the three cross-similar couples more clearly after their subsequences are normalized. Note that the period of the first subsequence (363:417) of the second couple covers completely that of the first subsequence (378:395) of the third couple; however, the period of the second subsequence (180:234) of the second couple are entirely separate from that of the second subsequence (378:395) of the third couple. The two cross-similar couples are thus not trivial. In addition, the two subsequences of the third couple take place during the same period so the lines showing their normalized subsequences coincide as in Figure 2.

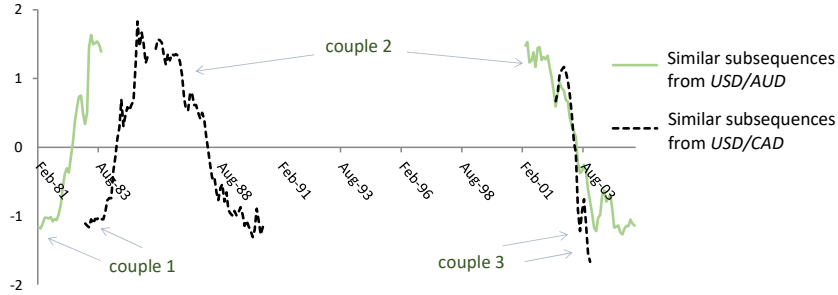

Figure 2. Cross-similar couples after the normalization

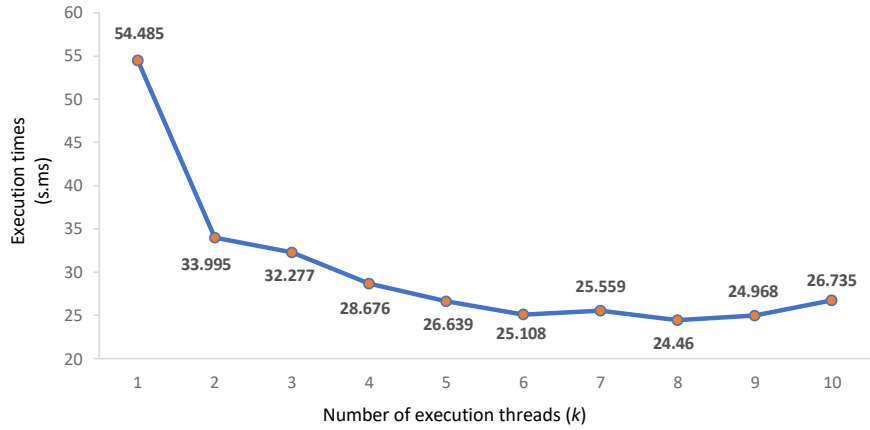

Figure 3. Execution times of the proposed method

Figure 3 shows the execution times of the proposed method using k execution threads, where k is from 1 to 10. For k = 1, the method works in the conventional fashion, i.e., single threading. When the method utilizes multithreading, i.e., k = 2, its runtime performance is improved dramatically. Since then, its runtime performance has a gradual improvement until k = 6. The runtime performance of the method has a tendency of a slight decline in case of k > 6. It is worth noting that every execution of the method with a specific value of k from 1 to 10 makes 79,250,570 calls to UCR-DTW and returns the resulting couples, as in Table 2.

Table 3. Details of the resulting couples

Couple	*infraredwave* Subsequence	*wirewave* Subsequence	Length	ED	DTW	ε
1	(418:979)	(425:986)	562	6.893	4.711	4.741
2	(2,549:3,058)	(2,560:3,069)	510	5.967	4.516	4.517
3	(2,637:3,207)	(2,646:3,216)	571	6.619	4.774	4.780

5.2. Time series of biosignal

The experiment works with the two time series, *infraredwave* and *wirewave* signals, from (Weigend, 2016). Each time series has 4,096 data points. In addition, the testbed for the experiment uses α = 0.2, and *minLength* = 500.

In the beginning, the proposed method is implemented in the conventional fashion, i.e., k = 1. After the execution has finished, three resulting couples are detected and detailed in Table 3. We consider couple 1. The DTW distance of the two normalized subsequences is less than ε, i.e., $4.711 < 0.2 \times \sqrt{562} \approx 4.741$; however, they are dissimilar in case ED is used due to 6.893 > 4.741. It is important to note that couple 2 overlaps couple 3; yet they overlap not too much. Their overlap ratio is less than β = 0.9. The consequence of the experiment also implies that if similarity join over *infraredwave* and *wirewave* accompanied with *minLength* = 500 and $\varepsilon = 0.2 \times \sqrt{l_x}$ is performed under ED, no cross-similar couples are found.

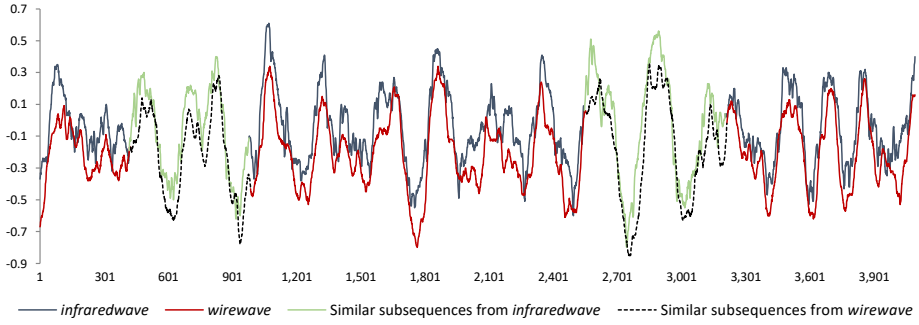

Figure 4. Visualization of *infraredwave* and *wirewave* and their similar subsequences

Figure 5. Cross-similar couples after the normalization

Figure 4 illustrates the two time series, *infraredwave* and *wirewave*, and their similar subsequences identified by the proposed method. The subsequences of these couples are normalized and then shown in Figure 5 to highlight their similarity.

The proposed method is also implemented using multithreading with various numbers of execution threads. Note that all these executions of the method using multithreading return the resulting couples as the same as those obtained by the execution of the method using single threading. Each execution of the method makes 15,519,622,195 calls to UCR-DTW and handle 34,585 cases of enormous overlaps, thereby processing a huge bulk of load. Hence, besides multithreading, another accelerative technique should be used.

Table 4. Execution times of the proposed method in various sceneries

#Execution thread (k)	No constraining windows (d.h:m:s)	Constraining window with r = 15 (h:m:s)
1	2.05:02:23	3:08:34
2	1.18:02:19	2:03:00
3	1.09:12:35	1:43:23
4	1.05:27:53	1:32:45
5	1.03:55:56	1:25:24
6	1.03:04:02	1:25:01
7	1.02:17:24	1:24:46
8	1.02:25:26	1:24:46
9	1.01:32:30	1:24:43
10	1.01:31:27	1:24:32

We use a *constraining window* with r =15 to reduce the number of UCR-DTW calls and this setting still enables the proposed method to return the resulting couples as in Table 3. The width of the constraining window is thus $2 \times r + 1 = 31$. The method using the constraining window invokes UCR-DTW 199,738,933 times and handles 34,585 cases of enormous overlaps. Therefore, the constraining window and no constraining windows give the same number of enormous overlaps; however, the number of UCR-DTW calls in case of using the constraining window decreases by 99%. The workload of the method using the constraining window decreases dramatically, so the method returns results faster.

Table 4 indicates the execution times of the proposed method in various sceneries. These sceneries are the method using single threading or multithreading, without constraining windows or using the constraining window with r = 15. The table shows that in case of single threading and no constraining windows, the execution time of the method is extremely huge, exceeding 2 days. Using multithreading and no constraining windows also speeds up the method; however, the execution times of the method are still over 1 day. If the method takes advantage of a constraining window with r = 15, its execution times shortens significantly, about several hours.

Figure 6 presents the speedups of the proposed method in various sceneries. The method using the constraining window with r = 15 has extremely good runtime performance. Notice that this happens for both similarity join over *infraredwave* and *wirewave*. When this task is conducted over other time-series pairs, we need to scrutinize the value of r to prevent false dismissals. For example, if the constraining window with r = 15 takes effect in the experiment presented in Subsection 5.1, the method will miss out couples 1 and 2 (see Table 2). Figure 6 also reveals that the runtime performance of the method is negligibly enhanced in case of $k > 6$. It is noteworthy that the experiment is performed with the CPU of 4 threads, so the optimal value of k might be greater than 6 when the experiment is conducted with CPUs of many more threads.

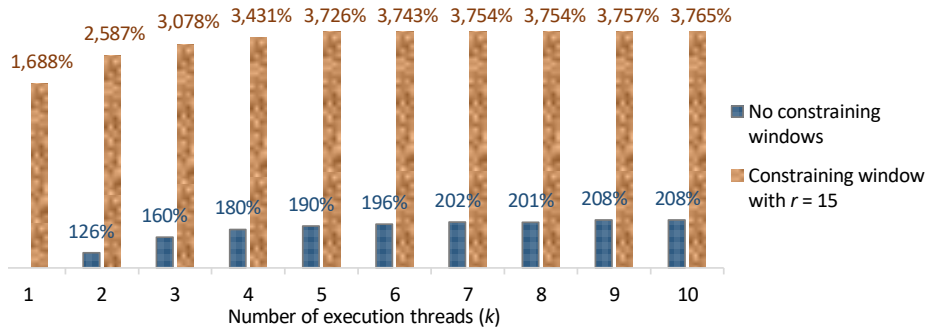

Figure 6. Speedups of the proposed method in various sceneries

6. CONCLUSION AND FUTURE WORK

This paper has proposed a method of similarity join over multiple time series under DTW, supporting *z*-score normalization. The manifestation of the method is Algorithm SJ-DTW searching for cross-similar couples from all couples of subsequences of two time series. For this reason, the method does not get false dismissals. To accelerate the similarity join, the algorithm uses UCR-DTW for computing the DTW distance between two subsequences from the time series to check the property of cross similarity. Since UCR-DTW works with normalized subsequences and various lower bounding functions, a working environment suitable for UCR-DTW needs establishing carefully. The algorithm facilitates data normalization by the technique of incremental *z*-score normalization and constructs the two envelopes of the time series once before scanning the two time series to search for cross-similar couples.

The proposed method employs multithreading to improve runtime performance. Each execution thread conducts similarity join on a length subregion which is separate with others. These execution threads work concurrently. In addition, regarding each case study of time series, the method might impose a constraining window on possibility of coupling two subsequences due to cross similarity, thereby having a fewer number of UCR-DTW calls. As a result, using a constraining window enables the method to enhance runtime performance. Combining these accelerative techniques makes the method return cross-similar couples quickly. The experimental evaluation of the method proves its effectiveness and efficiency in handling similarity join over multiple time series under DTW.

In future work, we plan to enhance the runtime performance of the proposed method using GPU.

ACKNOWLEDGMENT

This research is funded by Saigon University (SGU) under grant number CSB2022-47.

REFERENCES

Berndt, D. J., & Clifford, J. (1994). Using Dynamic Time Warping to find patterns in time series. AAAI-94 Workshop on Knowledge Discovery in Databases (pp. 359-370). Seattle, Washington, USA: AAAI Press. doi:10.5555/3000850.3000887

Chatzigeorgakidis, G., Patroumpas, K., Skoutas, D., Athanasiou, S., & Skiadopoulos, S. (2018). Scalable hybrid similarity join over geolocated time series. *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, (pp. 119-128). Seattle Washington, USA. doi: 10.1145/3274895.3274949

Chen, Y., Chen, G., Chen, K., & Ooi, B. C. (2009). Efficient processing of warping time series join of motion capture data. *2009 IEEE 25th International Conference on Data Engineering* (pp. 1048 - 1059). Shanghai, China: IEEE. doi:10.1109/ICDE.2009.20

Ding, H., Trajcevski, G., & Scheuermann, P. (2008). Efficient similarity join of large sets of moving object trajectories. *2008 15th International Symposium on Temporal Representation and Reasoning*, (pp. 79-87). Montreal, QC, Canada. doi:10.1109/TIME.2008.25

Giao, B. C. (2022). *Time-series datasets*. doi:0.13140/RG.2.2.12512.15360

Giao, B. C., & Anh, D. T. (2016). Similarity search for numerous patterns over multiple time series streams under dynamic time warping which supports data normalization. *Vietnam Journal of Computer Science, 3*(3), 181-196. doi:10.1007/s40595-016-0062-4

Keogh, E., & Ratanamahatana, C. A. (2004). Exact indexing of Dynamic Time Warping. *Knowledge and Information Systems, 7*(3), 358-386. doi:10.1007/s10115-004-0154-9

Kim, S. W., & Park, S. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. *Proceedings of the 17th IEEE International Conferenceon Data Engineering*, (pp. 607-614). Heidelberg, Germany. doi:10.1109/ICDE.2001.914875

Lemire, D. (2009). Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recognition, 42*(9), 2169-2180. doi:10.1016/j.patcog.2008.11.030

Lian, X., & Chen, L. (2009). Efficient similarity join over multiple stream time series. *IEEE Transactions on Knowledge and Data Engineering, 21*(11), 1544-1558. doi:10.1109/TKDE.2009.27

Lin, Y., & McCool, M. D. (2010). Subseries join: A similarity-based time series match approach. *14th Pacific-Asia Conference, PAKDD 2010* (pp. 238-245). Springer Berlin Heidelberg. doi:10.1007/978-3-642-13657-3_27

Mollah, M., Souza, V., & Mueen, A. (2021). Multi-way time series join on multi-length patterns. *2021 IEEE International Conference on Data Mining (ICDM)*, (pp. 429-438). Auckland, New Zealand. doi:10.1109/ICDM51629.2021.00054

Mueen, A., Hossein, H., & Trilce, E. (2014). Time series join on subsequence correlation. *Proceedings of 2014 IEEE International Conference on Data Mining* (pp. 450-459). Shenzhen, China: IEEE. doi:10.1109/ICDM.2014.52

Pele, L. (2023). *Currency converter using official exchange rates*. https://fxtop.com/

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., . . . Keogh, E. (2012). Searching and mining trillions of time series subsequences under Dynamic Time Warping. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)* (pp. 262-270). Beijing, China: ACM. doi:10.1145/2339530.2339576

Vinh, V. D., & Anh, D. T. (2016). Efficient subsequence join over time series under Dynamic Time Warping. *Recent Developments in Intelligent Information and Database Systems*, (pp. 41-52). doi:10.1007/978-3-319-31277-4_4

Wang, J., Li, Q., Li, Z., Wang, P., Wang, Y., Wang, W., . . . Chi, M. (2019). Similarity join on time series by utilizing a dynamic segmentation index. *Knowledge and Information Systems, 61*(2019), 1517-1546. doi:10.1007/s10115-018-1317-4

Weigend, A. S. (2016). *Time series prediction: Forecasting the future and understanding the past*. http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html. Accessed December 2016

Yeh, C. C. M., Zheng, Y., Wang, J., Chen, H., Zhuang, Z., Zhang, W., & Keogh, E. (2022). Error-bounded approximate time series joins using compact dictionary representations of time series. *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, (pp. 181-189). Alexandria, VA, USA. doi:10.1137/1.9781611977172

Yeh, C. C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., . . . Keogh, E. (2018). Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery, 32*(1), 83-123. doi:10.1007/s10618-017-0519-9

Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C.-C. M., Funning, G., Mueen, A., . . . Keogh, E. (2018). Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. *Knowledge and Information Systems, 54*(2018), 203-236. doi:10.1007/s10115-017-1138-x