IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 2, 2017, accepted November 27, 2017, date of publication December 8, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2781464

HybridFTW: Hybrid Computation of Dynamic Time Warping Distances

MINWOO LEE[1], SANGHUN LEE[2], MI-JUNG CHOI[2], (Member, IEEE), YANG-SAE MOON[2], (Member, IEEE), AND HYO-SANG LIM[3], (Member, IEEE)

[1]NHN Entertainment, Seongnam 13487, South Korea
[2]Department of Computer Science, Kangwon National University, Chuncheon 24341, South Korea
[3]Computer and Telecommunications Engineering Division, Yonsei University, Wonju 26493, South Korea

Corresponding author: Hyo-Sang Lim (hyosang@yonsei.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion, funded by the Korea Government, under Grant R7117-17-0214, in part by the Development of an Intelligent Sampling and Filtering Techniques for Purifying Data Streams, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science, ICT & Future Planning, under Grant NRF-2017R1A2B4008991.

ABSTRACT In this paper, we propose an efficient approach that computes the dynamic time warping (DTW) distance in time-series similarity search. The DTW distance is known to offer the high accuracy in similarity search, but it has difficulty in supporting the large database due to its high computational complexity. Recently, FastDTW and FTW have been proposed for efficient computation of DTW distances, but they have still performance limitations. In this paper, we propose a hybrid approach, called *HybridFTW*, which combines the advantages of both FastDTW and FTW. First, HybridFTW takes the advantage of FastDTW that provides fast computation through the limitation of allowable ranges. We call these allowable ranges *dynamic (warping) bands*, which reduce the computation spaces on the fly, and we reanalyze previous FastDTW and FTW in the viewpoint of *static* and *dynamic* bands. Second, HybridFTW also takes the advantage of FTW that exploits the early abandon effect by using the segment-based tight lower bound. To maximize the synergy of combining two methods, we obtain the dynamic band of FastDTW during the process of computing the lower bound in FTW. Using HybridFTW, we next propose range search and k-NN search algorithms and prove their correctness through formal theorems. Experimental results on real and synthetic data sets show that HybridFTW improves the search performance by up to 38 times over FastDTW and by up to 12 times over FTW.

INDEX TERMS Time-series data, similarity search, data mining, dynamic time warping distance, similar sequence matching.

I. INTRODUCTION

In recent years, there have been a lot of research efforts on a large volume of time-series data [8], [21]. Time-series data are the sequences of real numbers representing values at specific time points. Typical examples of time-series data include climate changes, stock prices, biomedical measurements, image boundary values, document retrieval, and object trajectory data [1], [4], [9], [12], [15]. In this paper, we deal with the problem of similar sequence matching in time-series databases. *Similar sequence matching* can be classified into range and k-nearest neighbor (k-NN) searches that identify *data sequences* similar to the given *query sequence*. As the similarity measure of similar sequence matching, the Manhattan distance [7], [27], the Euclidean distance [3], [18], and the dynamic time warping (DTW) distance [2], [8], [9] have been widely used. Among these distances, we focus on the DTW distance since it is known to be robust against a variety of distortions [2], [9]. However, its time complexity, $\Theta(n^2)$ for sequences of length n, is much higher than $\Theta(n)$ of Manhattan or Euclidean distances, and thus, the efficient computation of DTW distances has been an important research issue for many recent years.

There have been many research efforts [24]–[26], [28] for efficient computation of DTW distances, and the very recent ones are FastDTW [26] and FTW [25]. The basic concept of these two methods is to transform high dimensional sequences to low dimensional sequences and obtain the DTW distance or its lower bound by using those low

dimensional sequences. That is, they try to improve the performance by using *cheap* low dimensional sequences rather than *expensive* high dimensional sequences. First, FastDTW by Salvador and Chan [26] uses piecewise aggregate approximation (PAA) [24] to transform high dimensional sequences to low dimensional sequences and constructs warping paths over those low dimensional sequences. Second, FTW by Sakurai *et al.* [25] divides a high dimensional sequence into multiple low dimensional segments, computes lower bounds from those segments, and uses those lower bounds to improve performance of similar sequence matching.

Even though FastDTW and FTW improve the efficiency of computing DTW distances, we find critical limitations from both of them. We thoroughly analyze two methods step by step and present their pros and cons as follows. First, FastDTW has an advantage of improving the performance by limiting the allowable ranges of warping paths. On the other hand, it has a disadvantage of continuing the complex computation until the final DTW distance is obtained even if the intermediate distance exceeds the given query tolerance. Second, FTW has an advantage of exploiting the early abandon effect by using the segment-based lower bound. On the other hand, it has a disadvantage of not having any performance gain if the lower bound is smaller than the query tolerance. Based on these analytical observations, we raise a question: can we adopt their advantages only? As an answer of this question, in this paper we propose a new hybrid approach, called *HybridFTW*, for more efficient computation of DTW distances.

As the hybrid approach of FastDTW and FTW, HybridFTW limits the allowable ranges like FastDTW and at the same time exploits the early abandon effect like FTW. To combine two methods, we first present a novel concept of *dynamic (warping) bands*, which reduce the computation spaces of the DTW distance on the fly. We then reanalyze the previous computation methods in the viewpoint of *dynamic* and *static* bands. Based on the analysis, HybridFTW combines FastDTW and FTW as follows: it first takes the computation procedure of FTW as its basic skeleton for exploiting the early abandon effect; it then puts the limitation scheme of dynamic bands, a major merit of FastDTW, to the basic skeleton. In particular, to maximize the synergy of combining two methods, we obtain the dynamic band of FastDTW on the process of computing the lower bound in FTW.

We can also explain HybridFTW on the perspective of computation steps. First, it obtains lower bounds from the segment-based low dimensional sequences like FTW. In more detail, we adopt two key techniques from FTW: (1) the segmentation scheme of high dimensional sequences and (2) the computation scheme of lower bounds from the segmented low dimensional sequence. Second, HybridFTW obtains the dynamic bands (of FastDTW) from the warping paths which are previously constructed in computing the segment-based lower bound (of FTW). HybridFTW iterates these two computation steps for judging the DTW distance-based similarity among sequences. Please note that use of dynamic bands comes from FastDTW while that of segmentation and its lower bound comes from FTW. Likewise, by combing advantages of both methods, HybridFTW significantly improves the computing performance. This is because it can efficiently reduce the dynamic bands in the process of computing lower bounds as in FastDTW, and at the same time it can exploit the early abandon effect in an early stage as in FTW. Experimental results on real and synthetic data sets showcase that the proposed HybridFTW improves the similar sequence matching performance by up to 38 times and 12 times over FastDTW and FTW, respectively.

Contributions of the paper can be summarized as follows. First, we present advantages and disadvantages of FastDTW and FTW by analyzing their working mechanisms in detail. Second, we present a novel notion of dynamic (warping) bands and discuss how to efficiently use the dynamic bands by reanalyzing the previous computation methods. Third, we propose HybridFTW that combines FastDTW and FTW to take their efficiency merits and formally describe its framework with the detailed working steps. Fourth, we present HybridFTW-based range and k-NN search algorithms and prove their correctness through formal theorems. Fifth, through extensive experiments on real and synthetic data sets, we show that HybridFTW significantly outperforms FastDTW and FTW in evaluating range and k-NN queries.

The rest of the paper is organized as follows. Section II describes the related work on similar sequence matching and DTW distances. Section III formally explains concepts and working steps of FastDTW and FTW as the technical background of HybridFTW. In Section IV, we first propose HybridFTW and next present HybridFTW-based range and k-NN search algorithms. Section V shows the experimental results that compare HybridFTW with FastDTW and FTW. Finally, Section VI concludes the paper with a brief discussion of future work.

II. RELATED WORK

Time-series matching is the problem of finding data sequences similar to the given query sequence. It is also known as *time-series similarity search* or *similar sequence matching*, and we use those terms interchangeably throughout the paper unless confusion occurs. Time-series matching has a variety of applications including stock price prediction [5], biometric similarity search [6], query by humming [14], [30], sensor data prediction [29], and boundary image matching [13], [19]. Queries of time-series matching can be classified into range search and k-NN search. First, the range search finds data sequences whose distances from the query sequence are within the given search range ϵ, called *tolerance* [20], [23]. Second, the k-NN search finds k data sequences that are k-nearest neighbors to the query sequences [3], [9]. In general, the performance of range and k-NN searches largely depends on the input parameters, ϵ and k, respectively. In this paper, we consider both of these

range and k-NN searches as the target search algorithms of time-series matching.

Various similarity measures have been used in time-series matching, and the representative ones are the Euclidean distance and the DTW distance [2], [17]. In this paper, we use the DTW distance, which is known to be more robust than the Euclidean distance, and we explain it in detail. First, we define the DTW distance between two sequences as follows [8]. In general, DTW handles unequal length sequences as well as equal length sequences. However, as in previous works [25], [26], we deal only with equal length sequences to simplify the problem. Extending this to unequal length sequences is straightforward.

Definition 1: For two sequences $Q(= \{q_1, q_2, \ldots, q_n\})$ and $C(= \{c_1, c_2, \ldots, c_n\})$ of length n, their DTW distance $DTW(Q, C)$ is defined as Eq. (1).

$$\begin{aligned} &DTW(\{\}, \{\}) = 0; \\ &DTW(Q, \{\}) = DTW(\{\}, C) = \infty; \\ &DTW(Q, C) \\ &= \sqrt{dist(q_1, c_1) + \min \begin{cases} DTW(\{q_2, \ldots, q_n\}, \{c_2, \ldots, c_n\}) \\ DTW(\{q_2, \ldots, q_n\}, C) \\ DTW(Q, \{c_2, \ldots, c_n\}) \end{cases}}. \end{aligned} \tag{1}$$

In Eq. (1), {} is a null sequence, and $dist(q_1, c_1)$ is the squared Euclidean distance between q_1 and c_1, that is, $dist(q_1, c_1) = |q_1 - c_1|^2$. □

Since Eq. (1) has the exponential time complexity of $O(3^n)$, all the recent methods use the dynamic programming technique whose complexity is $\Theta(n^2)$. Definition 2 shows how to construct a two-dimensional matrix for dynamic programming with the related terminologies.

Definition 2: For two sequences Q and C of length n, a two-dimensional matrix $\gamma(1..n, 1..n)$ for computing the DTW distance can be obtained by Eq. (2), which is a recurrence equation to fill the matrix in dynamic programming.

$$\begin{aligned} \gamma(i, j) = &\, dist(q_i, c_j) \\ &+ \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \end{aligned} \tag{2}$$

We call the matrix $\gamma(1..n, 1..n)$ of Eq. (2) as the *warping matrix* and the path that provides the optimal (minimal) distance in the warping matrix as the *warping path*. □

If we construct the warping matrix by Definition 2, the DTW distance is $\gamma(n, n)$, i.e., $DTW(Q, C) = \gamma(n, n)$, and it corresponds to the Euclidean distance computed along with the warping path.

Figure 1 depicts the process of computing the DTW distance in dynamic programming. As shown in the figure, we first construct a two-dimensional warping matrix, and we then fill the matrix from the lower-left entry $\gamma(1, 1)$ to the upper-right entry $\gamma(n, n)$ using Eq. (2). Time complexity of the dynamic programming is $\Theta(n^2)$, which is much efficient compared to the recursive computation of Eq. (1). The complexity of $\Theta(n^2)$, however, is not sufficiently efficient for a huge database containing millions, billions, or even trillions of data sequences.

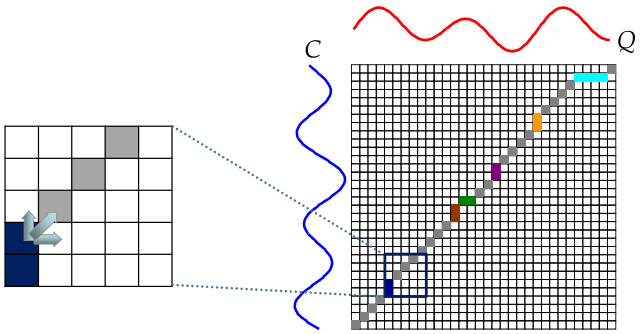

FIGURE 1. Warping matrix for computing the DTW distance in dynamic programming.

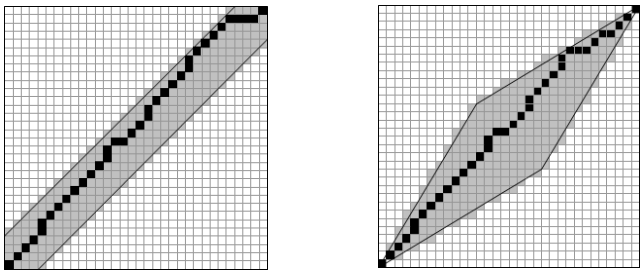

FIGURE 2. *Static bands* for limiting warping paths. (a) Sakoe-Chiba band. (b) Itakura-Parallelogram band.

To improve the efficiency of computing the DTW distance, many novel methods have been proposed recently [23]–[26]. First, we can use an allowable range of the warping path, where the warping path should be inside of the allowable range. We call this allowable range as the *static (warping) bands*, where we use the term 'static' since the band is fixed for all sequences. Figure 2 shows the representative static bands, Sakoe-Chiba band [24] and Itakura Parallelogram band [23]. As shown in the figure, when constructing the warping matrix, we consider the band only, a small part of the matrix. In other words, we enforce the warping path should be inside of the given band. Simply using the band we can reduce the computing space and improve the performance. These band-based methods, however, may not obtain the actual DTW distance since the warping path for the actual distance can be outside of the band.

Second, we can use the lower dimensional transformation to achieve the high efficiency. The representative ones are FastDTW [26] and FTW [25]. The basic procedures of these two methods are as follows: they first transform high dimensional sequences to low dimensional sequences; they then obtain the DTW distance or its lower bound from those low dimensional sequences. In this paper, we use these two methods as the major technical background of the proposed HybridFTW, and we will explain them in detail in Section III.

III. BACKGROUND AND MOTIVATION

In this section, we explain FastDTW and FTW in detail. We also explain the motivation why we try to integrate those two methods for performance improvement.

As the first technical background, we explain FastDTW that narrows down the warping band to the final warping path by exploiting PAA. FastDTW consists of three major steps: *Coarsening*, *Projection*, and *Refinement*. More precisely, it executes the Coarsening step just once and iterates the Projection and Refinement steps until getting the DTW distance. These three steps work as follows.

- In the Coarsening step, it uses PAA to map a high dimensional sequence to multiple low dimensional sequences, where the lowest dimension is given by a user parameter.
- In the Projection step, it computes the first warping path from the lowest dimensional sequences. After then, it constructs a warping band for the next low dimensional sequences by projecting the previously obtained warping path to the next low dimension.
- In the Refinement step, it computes a warping path again in the previously constructed warping band.
- It repeats the Projection and Refinement steps until it obtains the final warping path for high dimensional sequences and computes the actual DTW distance from the final warping path.

We here note that FastDTW gradually shrinks the warping band by repeating the Projection and Refinement steps. In other words, FastDTW dynamically changes, actually reduces, the warping band while the traditional Sakoe-Chiba and Itakura-Paralleogram bands are not changed. To emphasize this dynamic property, in this paper we call the warping bands of FastDTW the *dynamic (warping) bands* to distinguish from static bands of [10] and [24].

The proposed HybridFTW adopts Projection and Refinement steps of FastDTW. To understand these two steps, we first redefine the dimension increasing parameter [10].

Definition 3: Suppose n-dimensional sequences $Q(=\{q_1, \ldots, q_n\})$, $C(=\{c_1, \ldots, c_n\})$ are transformed to f-dimensional sequences $Q^f(=\{q_1^f, \ldots, q_f^f\})$, $C^f(=\{c_1^f, \ldots, c_f^f\})$ and m-dimensional sequences $Q^m(=\{q_1^m, \ldots, q_m^m\})$, $C^m(=\{c_1^m, \ldots, c_m^m\})$, respectively, by PAA. In the Projection and Refinement steps, we increase the dimension from f to m, where f and m have the relationship of Eq. (3), and we call *radius* of Eq. (3) the *dimension increasing parameter*.

$$m = 2 \times f \times radius \tag{3}$$

□

In Definition 3, the dimension increasing parameter, *radius*, controls how fast we shrink and how often we make the dynamic band. That is, if *radius* is small, dynamic bands are constructed many times, and they are shrunk finely; on the other hand, if *radius* is large, the bands are constructed only a few times, and they are shrunk coarsely [26].

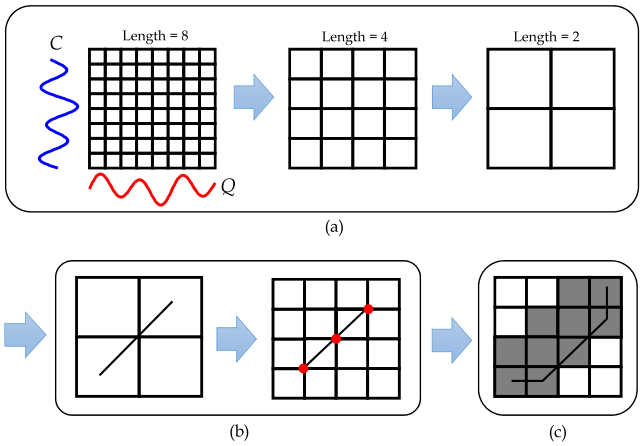

FIGURE 3. Process of computing the DTW distance in FastDTW. (a) Coarsening step. (b) Projection step. (c) Refinement step.

Figure 3 depicts how FastDTW computes the DTW distance for 8-dimensional sequences. In Example 1, we explain three steps of FastDTW in detail.

Example 1: In Figure 3, we use two sequences Q and C of length 8, and we set the lowest dimension to 2 and the dimension increasing parameter, *radius*, to 1. In the Coarsening step of Figure 3(a), we obtain 4- and 2-dimensional sequences by applying PAA to 8-dimensional sequences since the lowest dimension and *radius* are 2 and 1, respectively. In the Projection step of Figure 3(b), FastDTW first constructs a 2-dimensional warping array from 2-dimensional sequences and obtains a warping path in the array. After then, it projects the 2-dimensional warping path to the next 4-dimensional warping array (refer to the right array of Figure 3(b)). We note that the dimension is increased from 2 to 4 since *radius* is set to 1; if *radius* is set to 2, the dimension will be changed from 2 to 8 by Eq. (3).

Figure 3(b) shows how we construct the warping band dynamically. As shown in the figure, the 2-dimensional warping path passes through three points in the 4-dimensional warping array, and thus, we bound their adjacent cells as the dynamic warping band (see the shaded cells in Figure 3(c)). In the Refinement step of Figure 3(c), we next obtain the 4-dimensional warping path within the previously constructed 4-dimensional band. After then, we can get the 8-dimensional dynamic band from the 4-dimensional warping path. Likewise, we repeat these Projection and Refinement steps until we reach the original highest dimension and obtain the final warping path and its corresponding DTW distance. □

As explained in Example 1, we execute the Coarsening step just once to obtain low dimensional warping arrays and repeat the Projection and Refinement steps to obtain warping paths and dynamic bands.

As the second technical background, we explain FTW that improves the performance of DTW distance-based similarity search. It consists of three major concepts: *LBS* (Lower Bounding distance measure with Segmentation),

TABLE 1. Comparison of existing and proposed methods.

Method	Early abandon	Dynamic band	Remark
FastDTW	×	◯	Use dynamic bands to obtain the DTW distance with less computation.
FTW	◯	×	Use lower bounds to prune dissimilar sequences at an early stage.
HybridFTW	◯	◯	Use dynamic bands for fast DTW computation and at the same time use lower bounds for pruning many sequences at an early stage.

EarlyStopping, and *Refinement*. First, LBS is a lower bound of the DTW distance. FTW obtains LBS as follows: (1) it divides a high dimensional sequence into multiple segment sequences of the same size; (2) it gets maximum and minimum entries of each segment sequence; and (3) it computes a lower bound, called LBS, by constructing a 2-dimensional array of those maximum and minimum entries and by applying dynamic programming to the array. Second, EarlyStopping is the phase of performing early stopping, also known as early abandon [11]. That is, if the intermediate LBS is larger than the user-specified tolerance, FTW ignores the next computation for achieving the high search performance. Third, Refinement reduces the segment size if the early stopping does not occur and repeats LBS and EarlyStopping phases for the reduced segment sequences. In the proposed HybridFTW, we also use these three concepts in a similar way, and we will explain their working mechanisms in detail in Section IV.

Table 1 compares two existing methods and our HybridFTW. As shown in the table, FastDTW uses the dynamic band, and it focuses on computing the exact DTW distance efficiently by reducing the warping band dynamically. However, it does not use the early abandon, which prunes dissimilar sequences as soon as possible for performance improvement of similarity search. On the other hand, FTW exploits the early abandon by using the segment-based lower bounds and achieves the high performance of similarity search. However, it does not use the concept of dynamic band, and thus, it considers the whole array cells for each iteration of computing the lower bound. By taking advantages of FastDTW and FTW, our HybridFTW uses dynamic bands to reduce computation overhead of the DTW distance or its lower bounds, and at the same time it uses lower bounds to prune many sequences at an early stage of similarity search.

IV. HybridFTW: HYBRID COMPUTATION OF DYNAMIC TIME WARPING DISTANCES

In this section, we explain the concept of HybridFTW and propose the HybridFTW-based range and k-NN searches. First, we explain the basic concept of HybridFTW and its related work in Section IV-A. Next, we present efficient range and k-NN search algorithms that exploit HybridFTW in Sections IV-B and IV-C, respectively.

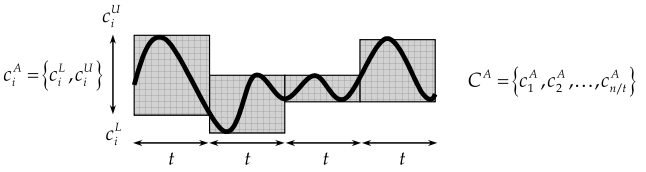

FIGURE 4. Construction of the segment sequence and its entries.

A. CONCEPT OF THE HYBRID APPROACH

The proposed HybridFTW consists of three major steps: *Hybrid-Stopping*, *Hybrid-Projection*, and *Hybrid-Refinement*. In the Hybrid-Stopping step, it constructs segment sequences by dividing a sequence into multiple segments, as in EarlyStopping of FTW. Definition 4 shows how to construct segment sequences in HybridFTW as well as in FTW [25].

Definition 4: For a sequence $C(= \{c_1, \ldots, c_n\})$ of length n, we divide the sequence C into the *segment sequence* $C^A(= \{c_1^A, \ldots, c_{n/t}^A\})$, where the i-th entry c_i^A of C^A is obtained from t entries of C, $c_{t\cdot(i-1)+1}, \ldots, c_{t\cdot i}$, by using Eq. (4).

$$c_i^A = \left(c_i^L : c_i^U\right), \quad \text{where } c_i^L = \min_{j=1}^{t} c_{(t\cdot(i-1)+j)}, \quad c_i^U = \max_{j=1}^{t} c_{(t\cdot(i-1)+j)}. \tag{4}$$

□

Figure 4 depicts how to construct the segment sequence C^A and its entries c_i^A. As shown in the figure, t is the segment size, and c_i^U (c_i^L) is the maximum (minimum) value among all entries of the i-th segment. We assume that t is a divisor of n without loss of generality.

Definition 5 describes LBS, a lower bound distance computed from two segment sequences that are constructed by Definition 4.

Definition 5: Suppose n-dimensional sequences Q, C are divided into t-sized segment sequences $Q^A(= \{Q_1^A, \ldots, Q_{n/t}^A\})$, $C^A(= \{C_1^A, \ldots, C_{n/t}^A\})$ of length $\frac{n}{t}$, respectively. We define the *segment matrix* by the array g of size $\frac{n}{t} \times \frac{n}{t}$ constructed from Q^A and C^A by Eq. (5) and the *segment path* by the minimum path computed from the *segment matrix* g.

$$g(i,j) = g_{cell}(i,j) + \min(g(i-1,j-1), g(i,j-1), g(i-1,j)), \tag{5}$$

$$g_{cell}(i,j) = \min(q_i^T, c_j^T) \times D_{seg}(q_i^R, c_j^R),$$
$$D_{seg}(q_i^R, c_j^R) = \begin{cases} ||q_i^L - c_j^U|| & (q_i^L > c_j^U) \\ ||c_i^L - q_j^U|| & (c_i^L > q_j^U) \\ 0 & (otherwise), \end{cases}$$
$$g(0,0) = 0, \quad g(i,0) = g(0,j) = \infty.$$

In Eq. (5), q_i^T (or c_i^T) is the i-th time interval and q_i^R (or c_i^R) is the i-th segment range, and these are the notations given in FTW [25]. We also define *LBS* of Q^A and C^A by the distance computed from the *segment path* of the *segment matrix*, and we denote it by $LBS(Q^A, C^A)$. □

Lemma 1 shows that LBS of Definition 5 is a lower bound of the actual DTW distance [25].

Lemma 1: Suppose segment sequences Q^A and C^A are constructed from original sequences Q and C, respectively, then $LBS(Q^A, C^A)$ is a lower bound of $DTW(Q, C)$. That is, Eq. (6) holds.

$$LBS(Q^A, C^A) \leq DTW(Q, C). \tag{6}$$

Proof: The proof process of constructing segment sequences and computing their LBS is the same as that of [25]. Therefore, readers are referred to [25] for the detailed proof of its lower bound property. □

Hybrid-Stopping of HybridFTW uses the concept of LBS and EarlyStopping of FTW. That is, during the process of computing LBS in Earlystopping of FTW (i.e., during the process of constructing a segment matrix g), if the intermediate distance of a segment is larger than the given tolerance, FTW prunes the next computation. Hybrid-Stopping also uses the same pruning strategy. Unlike EarlyStopping, however, Hybrid-Stopping obtains the segment path of Definition 5 while computing LBS. Note that this segment path will be very used in the next step, Hybrid-Projection.

Example 2: Figure 5 shows an example of processing Hybrid-Stopping (i.e., EarlyStopping of FTW). As shown in the figure, Q^A and C^A are the segment sequences constructed from Q and C, respectively; g is the segment matrix constructed from Q^A and C^A by dynamic programming. We here set the tolerance to 30. During the process of computing LBS (i.e., constructing matrix g), we do not compute $g(1,4)$ since $g(1,3)$ is larger than the tolerance. This is because if $g(1,3)$ is larger than the tolerance, $g(1,4)$ is also obviously larger than the tolerance. Similarly, we do not need to compute $g(2,4)$ since $g(2,3)$ is larger than the tolerance. We also note that $g(2,1)$ is larger than $g(2,2)$, and thus, we can ignore $g(i,1)$ for the third $(i = 3)$ and fourth $(i = 4)$ columns and start to compute $g(i,2)$ for those two columns. Eventually, we can discard the sequence C since $LBS(Q^A, C^A)$ $(= 42)$ is larger than the tolerance $(= 30)$ without processing the next steps. □

As the second step, Hybrid-Projection determines a dynamic band of the next step from the low dimensional warping path. Like FastDTW, we then increase the dimension by *radius* of Eq. (3). Definition 6 explains how to construct current and next step bands dynamically.

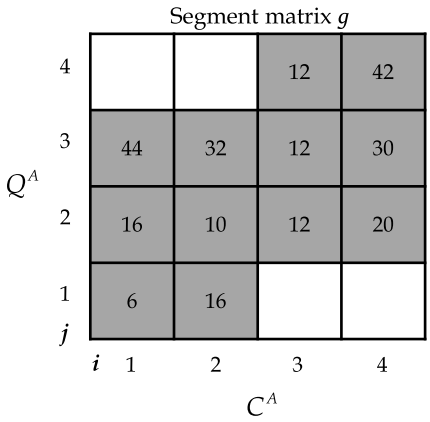

FIGURE 5. An example of processing the Hybrid-Stopping step.

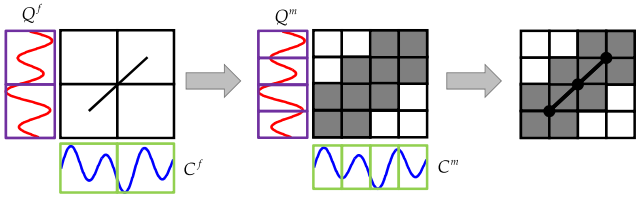

FIGURE 6. Process of constructing PL-Band in Hybrid-Projection.

Definition 6: Suppose two n-dimensional sequences Q, C are transformed into the *current* f-dimensional sequences Q^f, C^f, and into the *next* m-dimensional sequences Q^m, C^m, where $m = 2 \times f \times radius$ by Eq. (3). After obtaining the current segment matrix and segment path for Q^f and C^f, we construct a new band by projecting the segment path between Q^f and C^f into the next segment matrix between Q^m and C^m. We call this projected band *PL-Band* (*Projected LBS Band*). □

As shown in Definition 6, the proposed Hybrid-Projection differs from Projection of FastDTW in constructing the warping band. That is, Hybrid-Projection constructs the band based on the segment path between segment sequences of Hybrid-Stopping, while FastDTW constructs it based on PAA. Likewise, both HybridFTW and FastDTW construct the band on the fly, but HybridFTW produces much narrower bands than FastDTW since it exploits the segment concept of FTW rather than the static PAA.

Example 3: Figure 6 shows the procedure of processing Hybrid-Projection. HybridFTW first computes the segment matrix and segment path based on segment sequences Q^f and C^f, and it then computes LBS by the distance along with the segment path. If the early abandon does not occur by the current LBS, HybridFTW proceeds the projection to the next step. In Figure 6, *radius* is 1; m becomes $4 (= 2 \times 2 \times 1)$ by Eq. (3); HybridFTW computes the segment matrix and segment path from 4-dimensional segment sequences Q^f and C^f. As shown in the figure, we project the 2-dimensional segment path into the 4-dimensional matrix, and we include neighborhood cells of the projected path into a

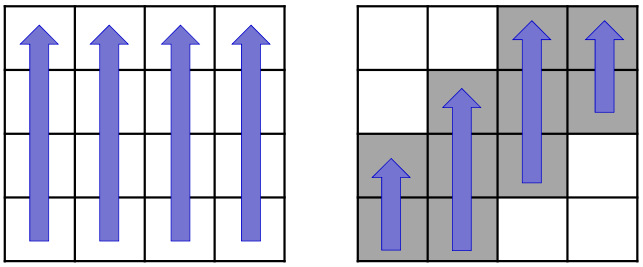

FIGURE 7. Comparison of the refinement procedures of FTW and HybridFTW. (a) Refinement of FTW. (b) Hybrid-Refinement.

new PL-Band. Therefore, in the next Hybrid-Refinement step, we can compute the next LBS from the new PL-Band and try the early abandon again. □

As the third step, Hybrid-Refinement computes LBS within PL-Band, which is constructed in the Hybrid-Projection step. The refinement step of HybridFTW differs from that of FTW in choosing the first row of computations. More precisely, the refinement of FTW computes LBS starting from the first row of each column, while Hybrid-Refinement computes LBS starting from the first row of PL-Band. The concept of limiting cells in the band is borrowed from the Refinement step of FastDTW, and it can significantly reduce the number of investigated cells compared with FTW. HybridFTW tries to compute tight LBSs and prune unnecessary cells early by repeating the Hybrid-Stopping, Hybrid-Projection, and Hybrid-Refinement steps. However, these steps might be reached to the original sequences without any early abandon since LBS is a lower bound rather than the actual DTW distance. Therefore, for guaranteeing no false dismissal, we need to check the similarity of original sequences through computing the actual DTW distance in the final step. We can also use the early abandon [4], [16], that is, if the intermediate distance exceeds the tolerance, we immediately stop the computation of the DTW distance [23].

Figure 7 depicts the refinement procedure of FTW and HybridFTW. First, as shown in Figure 7(a), Refinement of FTW tries to perform the early abandon by computing LBS starting from the first row of each column. On the other hand, our Hybrid-Refinement computes LBS only within PL-Band, which is constructed in Hybrid-Projection. Second, as shown in Figure 7(b), we note that the first and second columns are started from the first row, but the third and fourth columns are started from the second and third rows, respectively. Also, we never consider outside cells of PL-Band. Likewise, computing LBS of HybridFTW is efficiently processed only within the band, and Lemma 1 guarantees that LBS by PL-Band is a lower bound of the DTW distance.

Finally, Figure 8 shows the overall procedure of HybridFTW. The procedure looks like FastDTW, but it quite differs from FastDTW in a few detailed steps. From the viewpoint of FastDTW, it considers light gray cells as well as dark gray cells. In contrast, HybridFTW uses different approaches to light and dark gray cells as follows: light gray cells are allowable ranges (bands) determined by the previously constructed segment path while dark gray cells are the region of computing LBS through Hybrid-Stopping. Likewise, HybridFTW first investigates LBS from dark gray cells only, and it then performs the early abandon for the cells of which LBSs exceed the tolerance on the fly. Therefore, even for the large tolerance, the number of investigated cells in HybridFTW becomes smaller than that of FastDTW, and it eventually outperforms FastDTW as well as FTW.

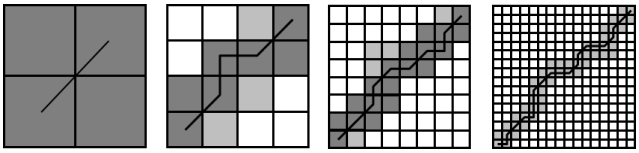

FIGURE 8. Procedure of computing LBS and DTW distance in HybridFTW.

B. HybridFTW-BASED RANGE SEARCH ALGORITHM

The eventual purpose of computing DTW distances is to measure the similarity between sequences for similar sequence matching. Representative queries in similar sequence matching are range and k-NN search queries, and we deal with the range search first in this section. Algorithm 1 shows the proposed HybridFTW-based range search algorithm. The inputs to the algorithm are query sequence Q, dimension increasing parameter r, and tolerance ϵ; the output is a set of data sequences whose DTW distances from Q are within the tolerance ϵ. In Line 1 of the algorithm, we initialize the result set $\mathbb{R}$. In Line 2, we divide the query sequence Q into a set Q^A of multiple segment sequences $Q_s^A, \ldots, Q_1^A$. Next, in Lines 4-9, we perform Hybrid-Stopping by using the corresponding segment sequences of query and data sequences. In Hybrid-Stopping of Line 5, we compute LBS and at the same time construct the segment path. If the computed LBS is larger than the tolerance, we perform the early abandon (Line 6); otherwise (i.e., if LBS is smaller than or equal to the tolerance), we construct the band of the next step by using the segment path (Line 7). That is, the function Expand-Windows() constructs the band through the segment path. In summary, Line 7 is relevant to Hybrid-Projection, and the process of performing Hybrid-Stopping based on this Hybrid-Projection is the very Hybrid-Refinement. If LBS is smaller than the tolerance, we repeat Hybrid-Projection and Hybrid-Refinement (Lines 5-8) until the cell size reaches to 1.

Next, Lines 10-13 decide the similarity by computing the actual DTW distance. That is, if the pruning step fails in Lines 4-9, we need to decide the similarity by computing the DTW distance in Lines 10-13. Function DTW-with-EarlyAbandon() in Line 11 computes the DTW distance, where if the intermediate distance exceeds the tolerance, it stops the computation immediately by the early abandon [23]. Finally, if the final distance (d_{exact}) is within

Algorithm 1 HybridFTW-Range-Search (Query Sequence Q, *Radius* r, Tolerance ϵ)

1. $\mathbb{R} := \emptyset$; // initialize the result set
2. Compute Q^A from Q; // $Q^A = \{Q_s^A, \ldots, Q_1^A\}$
3. **for each** data sequence $C \in$ *database* **do**
4. **for** $i := s$ **to** 1 **do** // s = maximum segment size
5. d_{lbs} := Hybrid-Stopping(Q_i^A, C_i^A, ϵ); // d_{lbs} is LBS between Q_i^A and C_i^A.
6. **if** $d_{lbs} > \epsilon$ **then break**;
7. **else** Expand-Windows(Q_i^A, C_i^A, r);
8. **end-if**
9. **end-for**
10. **if** $d_{lbs} < \epsilon$ **then**
11. d_{exact} := DTW-with-EarlyAbandon(Q, C, ϵ); // d_{exact} is the actual DTW distance.
12. **if** $d_{exact} \leq \epsilon$ **then** $\mathbb{R} := \mathbb{R} \bigcup C$;
13. **end-if**
14. **end-for**
15. **return** $\mathbb{R}$;

the tolerance, we include the data sequence into the set $\mathbb{R}$ in Line 12. A brief comparison of the differences between Algorithm 1 and FTW/FastDTW is as follows. First, we use Hybrid-Stopping() and DTW-with-EarlyAbadnon() instead of EarlyStoping() of FTW. Next, Algorithm 1 adopts Expand-Windows() for the window extension function of FastDTW.

Theorem 1 proves the correctness of the proposed HybridFTW-Range-Search algorithm.

Theorem 1: The HybridFTW-Range-Search algorithm performs the range search correctly. That is, HybridFTW-Range-Search retrieves all data sequences whose DTW distances from the query sequence Q are within the tolerance ϵ without any false dismissal.

Proof: In Lines 4-9 of HybridFTW-Range-Search, we prune the data sequence whose LBS exceeds the tolerance. Since LBS is a lower bound of the DTW distance by Lemma 1, the data sequence pruned by LBS has obviously a larger DTW distance than the tolerance. Thus, the pruning step does not incur any false dismissal. Next, in Lines 10-13, we decide the similarity between query and data sequences by computing the actual DTW distance. Thus, this step does not also incur any false dismissal. By these two major steps, HybridFTW-Range-Search retrieves all similar sequences correctly without any false dismissal. □

By Theorem 1, the HybridFTW-based range search finds all similar data sequences correctly, and we compare the proposed range search algorithm with the existing ones in Section V-B.

C. HybridFTW-BASED k-NN SEARCH ALGORITHM

Algorithm 2 shows the HybridFTW-based k-NN search algorithm. The inputs are Q, *radius*, and a coefficient k of k-NN; the output is a result set $\mathbb{R}$ of k data sequences which are k-nearest neighbors of Q in the DTW distance. We explain the algorithm in detail. In Line 1, we initialize the set $\mathbb{R}$, which is implemented as a *priority queue*. In Line 2, we divide the query sequence Q into s segment sequences Q^A as in the range search algorithm. We then initialize the variable *maxdist* which will be used as the k-th distance (Line 3). In Lines 5-10, we apply HybridFTW to compare query and data sequences first. That is, we perform Hybrid-Stopping by using the segment sequences of query and data sequences. In particular, in Line 6, we compute LBS between query and data sequences of the lowest step (i.e., $i = s$). After then, if LBS is larger than *maxdist*, we do the early abandon (Line 7); otherwise, we construct the band of the next step by using the segment path (Line 8). Expand-Windows() is a function of constructing the band as in the range search algorithm. That is, Line 8 corresponds to Hybrid-Projection; Hybrid-Stopping based on Hybrid-Projection corresponds to Hybrid-Refinement. In contrast, if LBS is less than or equal to *maxdist*, we repeat Hybrid-Projection and Hybrid-Refinement of Lines 5-10.

Algorithm 2 HybridFTW-k-NN-Search (Query Sequence Q, *Radius* r, Number k)

1. $\mathbb{R}$:= a priority queue having k entries; // an entry = <sequence, distance>
2. Compute Q^A from Q; // $Q^A = \{Q_s^A, \ldots, Q_1^A\}$
3. $maxdist := \infty$;
4. **for each** data sequence $C \in$ *database* **do**
5. **for** $i := s$ **to** 1 **do** // s = maximum segment size
6. d_{lbs} := Hybrid-Stopping(Q_i^A, C_i^A, *maxdist*); // $d_{lbs} = LBS(Q_i^A, C_i^A)$.
7. **if** $d_{lbs} > maxdist$ **then break**;
8. **else** Expand-Windows(Q_i^A, C_i^A, r);
9. **end-if**
10. **end-for**
11. **if** $d_{lbs} < maxdist$ **then**
12. d_{exact} := DTW-with-EarlyAbandon(Q, C, *maxdist*); // $d_{exact} = D(Q, C)$.
13. **if** $d_{exact} \leq maxdist$ **then**
14. **if** $\mathbb{R}$ is full **then** $\mathbb{R}$.pop();
15. $\mathbb{R}$.push($< C, d_{exact} >$);
16. $maxdist := \mathbb{R}.d_{max}$;
17. **end-if**
18. **end-if**
19. **end-for**
20. **return** $\mathbb{R}$;

Next, in Lines 11-18, we determine the similarity of original sequences by computing their actual DTW distance. That is, if the early abandon does not occur in Lines 5-10, we execute Lines 12-17 to know the similarity of two sequences. Function DTW-with-EarlyAbandon() in Line 12 is the same as that of Algorithm 1. If the final distance (d_{exact}) is within *maxdist*, we include the current data sequence into the priority queue $\mathbb{R}$. After including the current sequence into $\mathbb{R}$, we need to update *maxdist* as the maximum distance of $\mathbb{R}$, $\mathbb{R}.dmax$, in Line 16. We repeat Lines 5-18 for every data

sequence. Finally, we return the result set $\mathbb{R}$ which contains k-nearest neighbor data sequences from Q in the DTW distance. Similar to the range query of Algorithm 1, Algorithm 2 uses Hybrid-Stopping() and DTW-with-EarlyAbadnon() instead of EarlyStopping() of FTW and adopts Expand-Windows() for the window extension function of FastDTW.

Theorem 2 shows the correctness of the HybridFTW-k-NN-Search algorithm.

Theorem 2: The HybridFTW-k-NN-Search algorithm performs the k-NN search correctly. That is, HybridFTW-k-NN-Search retrieves k data sequences which are k-most similar with the query sequence Q in the DTW distance without any false dismissal.

Proof: In Lines 5-10 of the algorithm, we prune the data sequence whose LBS exceeds the k-th distance *maxdist*. Since LBS is a lower bound of the DTW distance by Lemma 1, this pruning step does not incur any false dismissal. Next, in Lines 11-18, we decide the similarity by computing the actual DTW distance. Thus, this post-processing step also incurs no false dismissal. Therefore, HybridFTW-k-NN-Search retrieves the k-most similar data sequences correctly without any false dismissal. □

By Theorem 2, the k-NN search algorithm finds k-NN data sequences correctly, and we compare it with the existing ones in Section V-C.

V. PERFORMANCE EVALUATION

A. EXPERIMENTAL DATA AND ENVIRONMENT

In the experiment, we use two different data sets. The first data set, a real ECG (electrocardiogram) data from PhysioNet [22], contains 20,000 time-series, each of which consists of 256 entries. We call this data set *ECG-DATA*. The second data set contains 50,000 random walk data, each of which consists of 256 entries generated by the random walk model [28]: the first entry is set to a random value in the range of (0, 10), and subsequent entries are obtained by adding a random value in the range of $(-0.5, 0.5)$ to the previous one. We call this data set *WALK-DATA*. Two data sets, ECG-DATA and WALK-DATA, have different characteristics, and the overall trend of their experimental results to be presented in Sections V-B and V-C is very similar. Even though we use other data sets, the experimental results would be similar to those of ECG-DATA and WALK-DATA. We will discuss additional experiments using various data sets in future works.

We evaluate the performance of HybridFTW against FastDTW and FTW by using both range and k-NN search algorithms. Through Theorems 1 and 2, we already formally prove the correctness of the proposed algorithms, and thus, here we focus on the performance improvement of HybridFTW. The hardware platform is an HP workstation equipped with Intel(R) Xeon(TM) 3.1GHz CPU, 4.0GB RAM, and 1TB HDD; its software platform is CentOS 5.9 Linux operating system. We use C/C++ language for implementing all distance computation and search algorithms.

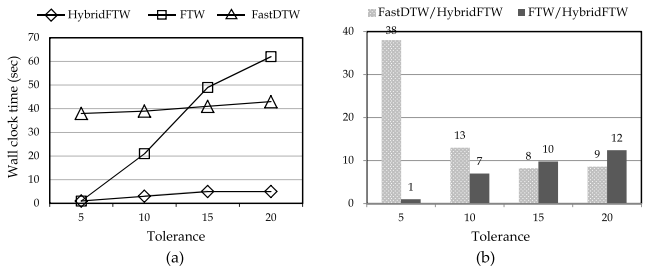

FIGURE 9. Experimental result of the range search for ECG-DATA.

We set *radius* to 2 and the segment sizes to $t_1 = 4$, $t_2 = 16$, $t_3 = 64$ as in [25]. The *radius* is used in HybridFTW and FastDTW; the segment sizes are used in HybridFTW and FTW. Sakurai *et al.* [25] have already showed that FTW is superior to LB_PAA, which is an advanced lower bound of LB_Keogh and LB_Improved. Thus, in this paper we compare HybridFTW with FTW since FTW outperforms LB_PAA, the tightest lower bound. As the performance metric, we measure the actual execution time of range and k-NN search algorithms. We take ten query time-series for each experiment and use their average execution time as the performance metric.

B. EXPERIMENTAL RESULT OF THE RANGE SEARCH

We first explain the experimental result of ECG-DATA. Figure 9 shows the actual and relative execution time of range search for ECG-DATA. In this experiment, we measure the wall clock time by increasing the tolerance ϵ from 5 to 20 by 5. In Figure 9(a), x axis represents the tolerance ϵ, and y axis represents the wall clock time of range search. We note that HybridFTW significantly reduces the wall clock time compared with FastDTW and FTW. In case of FastDTW, there is no or very little performance change as the tolerance increases. This is because FastDTW performs the range search after completing the computation of actual DTW distances. Due to this complete distance computation, FastDTW shows the worst performance among three methods. Next, when the tolerance is small, FTW, which performs EarlyStopping using LBS, shows the relatively good performance by exploiting the early abandon effect through segment sequences. However, the performance of FTW rapidly decreases (i.e., the execution time rapidly increases) as the tolerance increases. This is because FTW cannot exploit the early abandon effect any more for large segment sequences. On the other hand, the proposed HybridFTW shows the characteristics of integrating advantages of both methods. First, there is no or very little performance change according to the tolerance change like FastDTW; second, it significantly improves the performance by using the early abandon effect through LBS like FTW. In Figure 9(b), HybridFTW improves the performance by up to 38 times over FastDTW and by up to 12 times over FTW.

Next, Figure 10 shows the experimental result of the range search for WALK-DATA. In this experiment, the tolerances

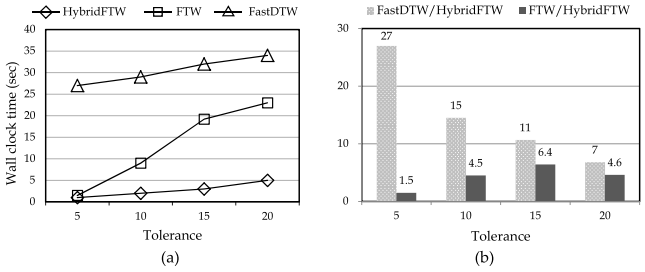

FIGURE 10. Experimental result of the range search for WALK-DATA.

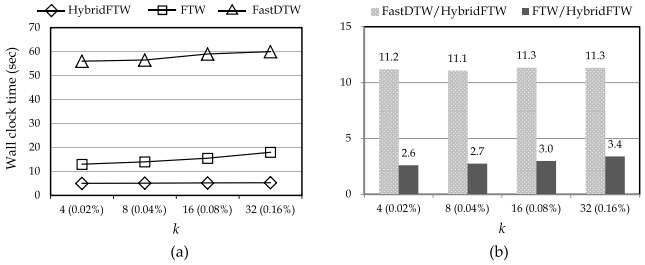

FIGURE 11. Experimental result of the k-NN search for ECG-DATA.

are the same as those of Figure 9. We note that the overall trend of Figure 10 is similar to that of Figure 9. That is, HybridFTW shows the best performance, FTW shows the next, and FastDTW shows the worst. A notable point is that the performance difference between HybridFTW and FTW is much smaller in Figure 10 than in Figure 9, especially for large tolerances. This is because, in case of WALK-DATA, the change of adjacent entries is very small, and thus, the LBS computed by segment sequences is not enough to prune many dissimilar data sequences. And accordingly, the performance of FTW and HybridFTW, which is largely influenced by LBS, is relatively worse in WALK-DATA than in ECG-DATA. In summary of Figure 10, HybridFTW improves the performance by up to 27 and 6.4 times over FastDTW and FTW, respectively.

C. EXPERIMENTAL RESULT OF THE k-NN SEARCH

Figure 11 shows the experimental result of the k-NN search for ECG-DATA. In the figure, x axis represents the coefficient k of k-NN, and y axis represents the actual wall clock time in Figure 11(a) and the relative execution time in Figure 11(b). In this experiment, we increase k from 4 to 32 by 2 times, which correspond to 0.02% to 0.16% of the total number of sequences. As shown in Figure 11, HybridFTW significantly reduces the k-NN search time compared with FastDTW and FTW. First, FastDTW shows no or little increase of execution times even for increase of k, but it shows the worst performance due to heavy overhead of computing the DTW distance completely. Second, FTW shows the relatively better performance than FastDTW by exploiting the early abandon effect through segment sequences. However, as k increases, the k-th distance *maxdist* also increases, and accordingly, the performance of FTW also decreases slightly since the early

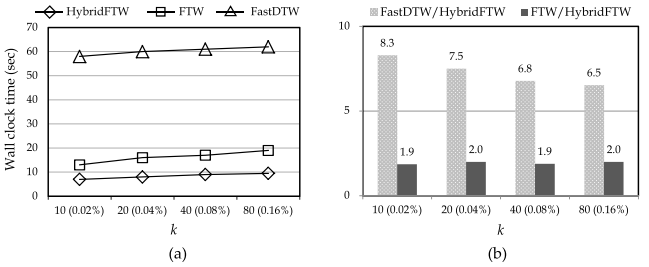

FIGURE 12. Experimental result of the k-NN search for WALK-DATA.

abandon effect is less exploited for large k's. By combining advantages of both FastDTW and FTW, the performance of HybridFTW shows no or little change even for different k's like FastDTW, and the overall performance improvement is much superior to FTW. In summary of Figure 11, HybridFTW improves the performance by up to 11 times over FastDTW and by up to 3.4 times over FTW.

Next, Figure 12 shows the experimental result of the k-NN search for WALK-DATA. In this experiment, we increase k from 10 to 80, which are the same percentages of Figure 11. Comparing Figure 12 with Figure 11, the result for WALK-DATA is very similar to that of ECG-DATA, but the trend is slightly different. First, the similar point is that HybridFTW still significantly outperforms FastDTW and FTW, and FastDTW shows the worst performance among three methods. Second, the different point is that the performance difference in Figure 12(b) is slightly smaller than that of Figure 11(b). This is because, as we explained in Section V-B, the entry changes in WALK-DATA are relatively smaller than those of ECG-DATA, and accordingly, the early abandon effect by LBS is less exploited in WALK-DATA than in ECG-DATA. In summary of Figure 12, HybridFTW outperforms FastDTW and FTW by up to 8.3 times and 2.0 times, respectively.

VI. CONCLUSIONS

In this paper, we proposed HybridFTW as a very fast approach of computing the DTW distance. The DTW distance offers the high accuracy in similarity search, but it has high computational complexity. Recently, FastDTW and FTW were proposed to improve the efficiency of computing the DTW distance. FastDTW transforms high dimensional sequences to low dimensional sequences and refines the DTW distance starting from low dimensional sequences step by step. FTW defines the LBS as a lower bound and uses it to exploit the early abandon effect for the performance improvement.

As a hybrid approach, HybridFTW takes advantages of both FastDTW and FTW for efficient computation of the DTW distance. That is, HybridFTW takes the advantage of FastDTW that provides fast DTW computation and at the same time takes the advantage of FTW that exploits the early abandon effect. To formally propose HybridFTW, in this paper we first reanalyzed the detailed working procedure

of FTW and FastDTW thoroughly. We then presented their pros and cons and discussed how to efficiently combine those advantages in detail. As a result, we proposed a concept of HybridFTW as a hybrid approach, presented three major steps of HybridFTW, and formally described its detailed working steps. After then, we proposed the HybridFTW-based range search and k-NN search algorithms, and we also formally proved their correctness in Theorems 1 and 2, respectively. Finally, through the experiments on real and synthetic data sets, we showcased that HybridFTW significantly outperformed FTW as well as FastDTW. As a future work, we will apply HybridFTW to various data mining applications including clustering and classification algorithms.

ACKNOWLEDGMENT

The majority of work by Minwoo Lee was done while he was a graduate student at Kangwon National University.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. 4th Int. Conf. Found. Data Org. Algorithms*, Chicago, IL, USA, Oct. 1993, pp. 69–84.

[2] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. KDD Workshop Knowl. Discovery Databases*, Seattle, WA, USA, Jul. 1994, pp. 359–370.

[3] F. K.-P. Chan, A. W.-C. Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: With and without time warping," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 686–705, May/Jun. 2003.

[4] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proc. Int. Conf. Manage. Data ACM SIGMOD*, Minneapolis, MN, USA, May 1994, pp. 419–429.

[5] E. J. de Fortuny, T. De Smedt, D. Martens, and W. Daelemans, "Evaluating and understanding text-based stock price prediction models," *Inf. Process. Manage.*, vol. 50, no. 2, pp. 426–441, Mar. 2014.

[6] T. Furon, H. Jegou, L. Amsaleg, and B. Mathon, "Fast and secure similarity search in high dimensional space," in *Proc. Int. Workshop Inf. Forensics Secur.*, Guangzhou, China, Nov. 2013, pp. 18–21.

[7] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: The DTW package," *J. Statist. Softw.*, vol. 31, no. 7, pp. 1–24, Aug. 2009.

[8] W.-S. Han, J. Lee, Y.-S. Moon, and H. Jiang, "Ranked subsequence matching in time-series databases," in *Proc. 33rd Int. Conf. Very Large Data Bases*, Vienna, Austria, Sep. 2007, pp. 423–434.

[9] W.-S. Han, J. Lee, Y.-S. Moon, S.-W. Hwang, and H. Yu, "A new approach for processing ranked subsequence matching based on ranked union," in *Proc. Int. Conf. Manage. Data ACM SIGMOD*, Athens, Greece, Jun. 2011, pp. 457–468.

[10] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-23, no. 1, pp. 67–72, Feb. 1975.

[11] L. Junkui and W. Yuanzhen, "Early abandon to accelerate exact dynamic time warping," *Int. Arab J. Inf. Technol.*, vol. 6, no. 2, pp. 144–152, Apr. 2009.

[12] M. S. Kim, K. Y. Whang, and Y. S. Moon, "Horizontal reduction: Instance-level dimensionality reduction for similarity search in large document databases," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 1061–1072.

[13] B.-S. Kim, Y.-S. Moon, M.-J. Choi, and J. Kim, "Interactive noise-controlled boundary image matching using the time-series moving average transform," *Multimedia Tools Appl.*, vol. 72, no. 3, pp. 2543–2571, Oct. 2014.

[14] A. Kotsifakos, I. Karlsson, P. Papapetrou, V. Athitsos, and D. Gunopulos, "Embedding-based subsequence matching with gaps–range–tolerances: A query-by-humming application," *VLDB J.*, vol. 24, no. 4, pp. 519–536, Aug. 2015.

[15] W.-K. Loh, S.-P. Kim, S.-K. Hong, and Y.-S. Moon, "Envelope-based boundary image matching for smart devices under arbitrary rotations," *Multimedia Syst.*, vol. 21, no. 1, pp. 29–47, Feb. 2015.

[16] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh, "Duality-based subsequence matching in time-series databases," in *Proc. 17th Int. Conf. Data Eng. (ICDE)*, Heidelberg, Germany, Apr. 2001, pp. 263–272.

[17] Y.-S. Moon, K.-Y. Whang, and W.-S. Han, "General match: A subsequence matching method in time-series databases based on generalized windows," in *Proc. Int. Conf. Manage. Data ACM SIGMOD*, Madison, WI, USA, Jun. 2002, pp. 382–393.

[18] Y.-S. Moon and J. Kim, "Efficient moving average transform-based subsequence matching algorithms in time-series databases," *Inf. Sci.*, vol. 177, no. 23, pp. 5415–5431, Dec. 2007.

[19] Y.-S. Moon, B.-S. Kim, M. S. Kim, and K.-Y. Whang, "Scaling-invariant boundary image matching using time-series matching techniques," *Data Knowl. Eng.*, vol. 69, no. 10, pp. 1022–1042, Oct. 2010.

[20] Y.-S. Moon and B. S. Lee, "Safe MBR-transformation in similar sequence matching," *Inf. Sci.*, vol. 270, pp. 28–40, Jun. 2014.

[21] A. Mueen, S. Nath, and J. Liu, "Fast approximate correlation for massive time-series data," in *Proc. Int. Conf. Manage. Data ACM SIGMOD*, Indianapolis, IN, USA, Jun. 2010, pp. 171–182.

[22] *Physiologic Signals*. Accessed: Nov. 1, 2017. [Online]. Available: http://www.physionet.org/

[23] D. Rafiei and A. O. Mendelzon, "Querying time series data based on similarity," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 5, pp. 675–693, Sep./Oct. 2000.

[24] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.

[25] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "FTW: Fast similarity search under the time warping distance," in *Proc. 24th ACM Symp. Principles Database Syst.*, Baltimore, MD, USA, Jun. 2005, pp. 326–337.

[26] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," *J. Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[27] Y. Yamada, E. Suzuki, H. Yokoi, and K. Takabayashi, "Decision-tree induction from time-series data based on a standard-example split test," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, USA, Aug. 2003, pp. 840–847.

[28] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," in *Proc. 26th Int.Conf. Very Large Data Bases*, Cairo, Egypt, pp. 385–394, Sep. 2000.

[29] J. Zhou and A. K. H. Tung, "SMiLer: A semi-lazy time series prediction system for sensors," in *Proc. Int.Conf. Manage. Data*, Melbourne, VIC, Australia, May 2015, pp. 1871–1886.

[30] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming," in *Proc. Int. Conf. Manage. Data*, San Diego, CA, USA, Jun. 2003, pp. 181–192.

MINWOO LEE received the B.S. and M.S. degrees in computer science from Kangwon National University in 2012 and 2014, respectively. He is currently with NHN Entertainment. His research interests include data mining, knowledge discovery, and privacy-preserving data mining.

SANGHUN LEE received the B.S. and M.S. degrees in computer science from Kangwon National University in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree in computer science. His research interests include data mining, knowledge discovery, data mining applications, and privacy-preserving data mining.

MI-JUNG CHOI (M'07) received the B.S. degree in CS from Ewha Womans University in 1998 and the M.S. and Ph.D. degrees from the Department of CSE, POSTECH, in 2000 and 2004, respectively. She was a Post-Doctoral Fellow with INRIA, France, from 2004 to 2005, and with the School of Computer Science, University of Waterloo, Canada, from 2005 to 2006. She was a Research Professor with POSTECH, South Korea, from 2006 to 2008. She is currently an Associate Professor with the Department of Computer Science, Kangwon National University, South Korea. Her research interests include malware analysis and SDN/NFV management.

YANG-SAE MOON (M'01) received the B.S., M.S., and Ph.D. degrees from the Korea Advanced Institute of Science and Technology in 1991, 1993, and 2001, respectively, all in computer science. From 1993 to 1997, he was a Research Engineer with Hyundai Syscomm, Inc., where he was involved in developing 2G and 3G mobile communication systems. From 2002 to 2005, he was a Technical Director with Infravalley, Inc., where he was involved in planning, designing, and developing CDMA and W-CDMA mobile network services and systems. He was a Visiting Scholar with Purdue University from 2008 to 2009. He is currently a Professor with Kangwon National University. His research interests include data mining, knowledge discovery, big data analysis and management, distributed processing systems, storage systems, access methods, multimedia information retrieval, mobile/wireless communication systems, and network communication systems. He is a member of the ACM.

HYO-SANG LIM (M'13) received the B.S. degree from Yonsei University, South Korea, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, all in computer science. He was a Post-Doctoral Research Fellow with Purdue University from 2007 to 2011. He is currently an Assistant Professor with the Computer and Telecommunications Engineering Division, Yonsei University Wonju Campus, South Korea. His research interests include database, database systems, data streams, sensor networks, database security, data trustworthiness, data quality, data integrity, and access control. He is a member of the ACM.