This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

Non-overlapping Subsequence Matching of Stream Synopses

Su-Chen Lin, *Student Member, IEEE,* Mi-Yen Yeh, *Member, IEEE,* and Ming-Syan Chen, *Fellow, IEEE*

Abstract—In this paper, we propose SUbsequence Matching framework with cell MERgence (SUMMER) for online subsequence matching between histogram-based stream synopsis structures under the dynamic time warping distance. Given a query synopsis pattern, SUMMER continuously identifies all the matching subsequences for a stream as the bins are generated. To effectively reduce the computation time, we design a Weighted Dynamic Time Warping (WDTW) algorithm, which computes the warping distance directly between two histogram-based synopses. Furthermore, a Stack-based Overlapping Filter Algorithm (SOFA) is provided to remove the overlapping subsequences to avoid the redundant information. Finally, we design an optional refinement module to relax the subsequence range limit and improve the matching accuracy. Our experiments on real datasets show that the proposed method significantly speeds up the pattern matching without compromising the accuracy required when compared with other approaches.

Index Terms—Subsequence matching, Dynamic Time Warping, Synopsis stream

1 Introduction

Real-time subsequence matching for streaming data is an important operation in data mining systems. It is originally motivated by the need of continuously monitoring certain queries on an evolving data stream, such as network intrusion detection, sensor monitoring, and financial data analysis. This operation becomes more challenging in the Big data era when the data are generated and accumulated in a very fast speed. Instead of tackling the raw streaming data directly, researchers resort to their compact representation to meet the real-time requirement of stream applications at a reasonable cost of accuracy loss. This results in abundant studies on stream synopsis techniques [1].

The histogram-based synopses, which summarize a data stream with a series of bins, i.e., piecewise constants, are relatively elegant and efficient to be used in an online environment among all well-received synopsis structures. Given that the input data stream is transformed to histogram-based synopses, the subsequence matching algorithm should make use of it.

Unfortunately, a general subsequence matching algorithm on histogram-based synopsis is yet to be developed. The original DTW computation is designed to handle time series with regular time intervals while each bin of a histogram-based synopsis stream may have various time spans. Of course we may convert each histogram bin to consecutive data points of the same value at fixed intervals and use the original DTW computation to compute the warping distance. However, such a method is obviously not efficient in a streaming environment since it is contrary to the original intent of reducing the size of data to be processed by constructing synopsis. Although there exist algorithms for subsequence matching on synopsis streams of *equal-width* bins such as [2]–[4], applying them directly on synopsis streams of *arbitrary-width* bins causes either accuracy loss or efficiency degradation. This motivates us to design a general subsequence matching algorithm that can deal with histogram-based synopsis streams of arbitrary-width bins.

Our goal is to design an online algorithm that can identify all non-overlapped subsequences satisfying a given distance threshold with minimized delays. In an online streaming environment, some applications require to continuously monitor whether there are similar instances in the evolving stream. In such applications, we may prefer to make range queries for subsequence matching rather than to use top one search. The range query aims to find all similar but non-overlapping subsequences to the query pattern exceeding a given distance threshold. The non-overlapping requirement is a useful criteria for removing redundant subsequences because the overlapped subsequences usually indicate the same instance with little time shift. Note that the non-overlapping examination may introduce the extra delay in reporting qualified subsequences. In light of this concern, the proposed algorithm should report these subsequences within an acceptable delay.

Dynamic Time Warping (DTW) is a robust distance measurement for time series since it offers elastic scaling and shifting capabilities in time axis, making it possible to match two subsequences of different lengths. As a result, DTW has been widely used in many applications such as speech processing and posture recognition. Hui et al. [5] have provided comprehensive experiments to demonstrate the better effectiveness of DTW compared with other common distance measurements. More recent studies [6]–[8] also agree that DTW is an important and better measurement and apply it on different applications. However, the time complexity of DTW is quadratic for a single pair of sequence

- *S.-C. Lin is with the Department of Electrical Engineering, National Taiwan University, Taiwan. E-mail: sclin@arbor.ee.ntu.edu.tw.*
- *M.-Y. Yeh is the Institute of Information Science, Academia Sinica, Taiwan. E-mail: miyen@iis.sinica.edu.tw.*
- *M.-S. Chen is with the Department of Electrical Engineering, National Taiwan University, Taiwan. E-mail: mschen@cc.ee.ntu.edu.tw.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

matching. The most common way to speed up the DTW computation includes lower bounding filters and matching path constraints [9]–[11]. For example, cDTW [11] restricts the warping path within a Sakoe-Chiba band to reduce the search space. Recently, there are also methods for the DTW computation on the streaming data. For example, UCR-DTW, the DTW version in UCR_SUITE, utilizes both the filter and path constraints to efficiently compute the DTW distance for streaming time series [7]. However, the above methods cannot be applied to the synopsis steams we studied in this paper. More details about the related work please refer to Section 7

To fill up the gap, we design a novel algorithm to do subsequence matching given a query pattern on synopsis stream data using the DTW distance. Note the query pattern and the stream data are both in the format of histogram-based synopses with arbitrary bin widths. The challenge is how to compute the DTW distance between two sequences continuously in an online fashion while reporting similar and non-overlapping subsequences.

To address this challenge, we propose SUbsequence Matching framework with cell MERgence (SUMMER), which is to the best of our knowledge, the first online subsequence matching framework that is able to efficiently process the arbitrary-width synopsis stream. This framework is composed of two stages: a *generating* stage and a *filtering* stage. In the generating stage, we propose the Weighted Dynamic Time Warping (WDTW) algorithm that generates a candidate subsequence at each new bin arrival. Taking the synopsis advantage, WDTW computes the DTW distance between the query sequence and a candidate subsequence by matching their bins instead of matching their data points. To approximate the warping distance more accurately, we further design a novel bin-cell mergence technique, which calculates the distances by merging adjacent bin pairs rather than independently calculating them. In the filtering stage, we propose the Stack-based Overlapping Filter Algorithm (SOFA) to remove the redundant subsequences. A stack is used to keep overlapping subsequences on hold until the best representatives among them are determined. We prove that there is no miss for the detection of qualified subsequences in SOFA. For applications that the accuracy is the main concern, we propose an optional refinement process that executes DTW again with the original sequence length. Through the refinement process, we can obtain qualified subsequences more accurately.

To evaluate our WDTW and SOFA algorithms, we conduct experiments on real datasets of time series. We compare the proposed algorithms with the state-of-the-art methods. The results show that the proposed SUMMER framework with WDTW and SOFA outperforms other methods in terms of both throughput and accuracy.

Our contributions are as follows. 1) To our knowledge, we are the first to propose the online subsequence matching algorithm of stream synopses which are composed of arbitrary-width bins under the DTW distance. 2) We propose WDTW, an efficient subsequence matching algorithm for histogram-based synopsis stream in streaming environment. We speed up the subsequence matching problem by $1/b^2$ times, where b is the synopsis rate which is the ratio of the number of bins to the number of points in data stream and $b \ll 1$ in general. With the significant speed-up, WDTW still leads to high-quality approximated results with a novel bin-cell mergence technique. 3) We provide SOFA, a stack-based filtering algorithm to retrieve all non-overlapping subsequences with minimized delays. We theoretically prove that no qualified subsequences are missed.

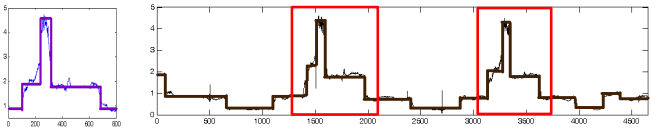


Fig. 1. The left side is the synopsis query sequence given and the right side is the online synopsis data sequence. The subsequences in the bold rectangles are the objectives of the synopsis subsequence matching.

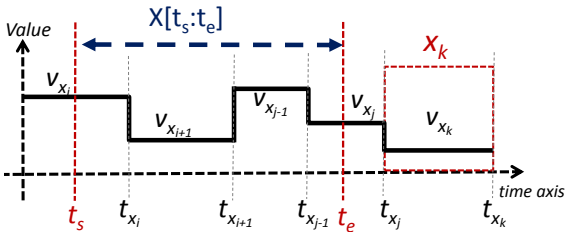


Fig. 2. The solid line represents a histogram-based synopsis stream composed of four bins of arbitrary widths. $X[t_s : t_e]$ represents a subsequence which starts from t_s and ends at t_e. In this example, $X[t_s : t_e]$ cross three bins.

The rest of the paper is organized as follows. Section 2 provides the problem definition and the framework outline. Sections 3 and 4 introduce WDTW and SOFA for the generating and filtering stages, respectively. Section 5 provides complexity analysis of SUMMER. Section 6 shows the performance. Related work is in Section 7. Finally, Section 8 concludes the paper.

2 Problem Formulation

2.1 Synopsis Subsequence Matching Problem

We deal with subsequence matching problem under the DTW distance for online synopsis streams. Our goal is to find all non-overlapping subsequences in minimized delays. An example is illustrated in Fig. 1. Given the synopsis query pattern in the left figure, our objective is to find the synopsis subsequences that are similar to the query, as those framed with bold rectangles.

We focus on the *histogram-based* synopsis structure that summarizes data streams to consecutive bins of arbitrary widths in the synopsis subsequence matching problem. Summarization techniques such as Haar wavelet reconstruction [12] and adaptive piecewise constraint approximation (APCA) [13] produce this format of synopses. Basically, the histogram-based summarization uses n piecewise constants to represent the raw stream of N data points, where $n \ll N$ in general. As a result, the synopses of a stream look like a histogram with several consecutive bins of arbitrary widths. An example of the arbitrary-width histogram-based synopsis is shown in Fig. 2, where the stream is summarized with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

four bins. Each bin represents several consecutive points with the same value. Since the histogram-based synopsis structure can be generated faster than other synopsis structures, it is more suitable for streaming environments.

Definition 1 (Histogram-based synopsis stream and bin)**.** *A histogram-based synopsis stream is composed of consecutive bins. We denote an online histogram-based synopsis stream by* $X = \{\langle t_{x_1}, v_{x_1}\rangle, \langle t_{x_2}, v_{x_2}\rangle, ..., \langle t_{x_i}, v_{x_i}\rangle, ...\}$*, where* t_{x_i} *and* v_{x_i} *are the endpoint timestamp and the value of the* i^{th} *bin, respectively. We also denote the* i^{th} *bin by* x_i*. For each bin* x_i*, it starts from* $t_{x_{i-1}} + 1$ *and ends at* t_{x_i}*. Thus, its width (time span), denoted by* l_{x_i}*, equals* $t_{x_i} - t_{x_{i-1}}$*, where* $t_{x_0} = 0$*.* □

Definition 2 (Synopsis subsequence and its reconstructed subsequence)**.** *A synopsis subsequence that starts from timestamp* t_s *and ends at timestamp* t_e *of a synopsis stream is denoted by* $X[t_s : t_e]$*, where* t_s *and* t_e *may not always be an endpoint timestamp of bins. To generally represent the subsequence* $X[t_s : t_e]$ *as consecutive bins, we have* $X[t_s : t_e] = \{\langle t_{x_i}, v_{x_i}\rangle, \langle t_{x_{i+1}}, v_{x_{i+1}}\rangle, ..., \langle t_e, v_{x_j}\rangle\}$ *with starting timestamp* t_s*, where* t_s *and* t_e *are located in the bin* $\langle t_{x_i}, v_{x_i}\rangle$ *and* $\langle t_{x_j}, v_{x_j}\rangle$ *in the synopsis stream* X*, respectively. The first bin* $\langle t_{x_i}, v_{x_i}\rangle$ *starts from* t_s *and ends at* t_{x_i}*; the last bin* $\langle t_e, v_{x_j}\rangle$ *starts from* $t_{x_{j-1}} + 1$ *and ends at* t_e*. The other bins between* x_i *and* x_j *are the same as bins in* X*.*

In addition, we denote the time sequence reconstructed from $X[t_s : t_e]$ *by* $X'[t_s : t_e]$*. In the reconstructed subsequence, one data value exists at each timestamp. For each bins in* $X[t_s : t_e]$*, say* x_k*, it is reconstructed to* l_{x_k} *consecutive points with the same value* v_{x_k}*, of which the last point is at time* t_{x_k}*.* □

Example 1. *Fig. 2 is an example of a histogram-based synopsis sequence* X*. The bin* x_k *is framed with the dashed rectangle. We have synopsis subsequence* $X[t_s : t_e] = \{\langle t_{x_i}, v_{x_i}\rangle, \langle t_{x_{i+1}}, v_{x_{i+1}}\rangle, \langle t_{x_{j-1}}, v_{x_{j-1}}\rangle, \langle t_e, v_{x_j}\rangle\}$ *with starting timestamp* t_s *and the corresponding reconstructed sequence* $X'[t_s : t_e] = \{\underbrace{v_{x_i}, \cdots v_{x_i}}_{t_{x_i} - t_s + 1}, \underbrace{v_{x_{i+1}}, \cdots x_{i+1}}_{t_{x_{i+1}} - t_{x_i}}, \underbrace{x_{j-1}, \cdots x_{j-1} x_j, \cdots x_j}_{t_{x_{j-1}} - t_{x_{i+1}}\ t_e - t_{x_{j-1}}}\}$*.* □

Following the above definition, a histogram-based synopsis query sequence summarized with m bins is denoted by $Q = \{\langle t_{q_1}, v_{q_1}\rangle, \langle t_{q_2}, v_{q_2}\rangle, ...\langle t_{q_m}, v_{q_m}\rangle\}$. For convenience, we simply refer to the histogram-based synopsis stream/sequence as synopsis stream/sequence in the rest of the paper.

Our final goal of the synopsis subsequence matching problem is to retrieve all non-overlapping subsequences in minimized delays. Thus, we first present the definitions of overlap and disjoint. Then we formally define our target, the maximal non-overlapping subsequences set.

Definition 3 (Overlap and Disjoint)**.** *Given two subsequences* $p = X[t_s : t_e]$ *and* $q = X[t'_s : t'_e]$*,* p *and* q *are disjoint if* $t_e < t'_s$ *or* $t'_e < t_s$*. Otherwise,* p *overlaps* q*. Note that overlap is a symmetric operator.* □

Definition 4 (Maximal Non-Overlapping Subsequence Set)**.** *Given a synopsis subsequence set* S*, where all subsequences are from a synopsis stream* X *and no two subsequences have both the same starting and end points, a subset* $R \subseteq S$ *is a maximal non-overlapping subsequences set if 1)* $\forall r \in R$*,* $\nexists r' \in R$ *and* $r \neq r'$ *that* r' *overlaps* r *in time axis, and 2)* $\forall h \in S - R$*,* $\exists r \in R$ *that* r *overlaps* h *and* $Dtw(h, Q) \geq Dtw(r, Q)$*, where* $Dtw(r, Q)$ *is the DTW distance between the reconstructed sequences* r' *and* Q'*. We break the tie that the endpoint timestamp of* r *is smaller than that of* h*. In other words,* r *ends earlier than* h *does.* □

Condition 2) says that for any filtered subsequence h, there exists one subsequence r whose DTW distance to Q is not larger than the distance of the filtered one h. Thus, the remaining subsequences ($r \in R$) are non-overlapping and more similar to the query than the filtered ones.

The formal problem definition is as follows.

Definition 5 (Synopsis Subsequence Matching)**.** *For an online synopsis stream* X*, given a fixed length of query pattern synopsis* Q *and a distance threshold* ϵ*, the goal is to locate all synopsis subsequences* $X[t_s : t_e]$ *that satisfy the following constraints:*

1) *Similarity constraint:* $Dtw(X[t_s : t_e], Q) < \epsilon$*,*
2) *Non-overlapping constraint: when a group of subsequences satisfying similarity constraint has overlaps, only the ones in the maximal non-overlapping subsequences set will be reported.*

$Dtw(X[t_s : t_e], Q)$ *is the DTW distance between two reconstructed sequences* $X'[t_s : t_e]$ *and* Q'*. Note that* t_s *and* t_e *are not necessary the endpoints of bins. The subsequences satisfying these constraints are called* qualified subsequences*.* □

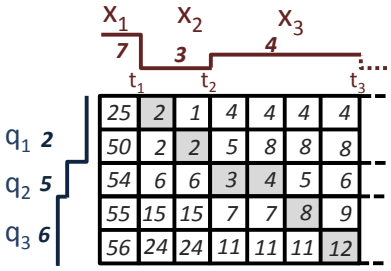
It is note that SPRING has been proposed to solve the online subsequence matching problem on a raw data stream [14]. It can identify all subsequences without sacrificing any accuracy in linear time to the stream length. Although SPRING can be adapted to solve the synopsis subsequence by transforming stream synopses back to the original representation, i.e., one data value per timestamp, SPRING still suffers from its time complexity. Its time complexity is $O(MN)$, where M is a constant length of the query pattern and N is the length of stream X. If the data arrival rate is high, that is, the time interval between two data points of stream X is very small, this execution time may not meet the real-time requirement in streaming environments in some cases. We use Example 2. to illustrate how the adapted SPRING performs.

Example 2. *We want to find the optimal subsequence of the synopsis stream* $X = \{\langle 1, 7\rangle, \langle 3, 3\rangle, \langle 7, 4\rangle, ...\}$ *at time* 7 *given a synopsis query pattern* $Q = \{\langle 2, 2\rangle, \langle 3, 5\rangle, \langle 5, 6\rangle\}$ *as shown in Fig. 3(a). By reconstructing* X *and* Q *to the raw data streams form as* $X' = \{7, 3, 3, 4, 4, 4, 4\}$ *and* $Q' = \{2, 2, 5, 6, 6\}$*, SPRING fills up all the original-cells in the accumulated distance table* $\mathcal{D}$ *and finds the optimal subsequence* $X[2 : 7]$ *with a distance* 12 *to the query* Q*.* □

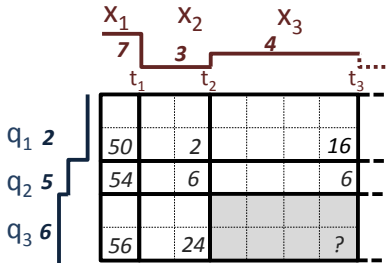
2.2 SUMMER: Subsequence Matching Framework

To meet the real-time requirement, we propose SUbsequence Matching framework with cell-MERge (SUMMER) to resolve the synopsis subsequence matching problem in Definition 5. In the proposed framework, we design the *Generating* stage and the *Filtering* stage to deal with both similarity and non-overlapping constraints, and an optional

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering



(a) SPRING uses an original-cell as a computing unit



(b) WDTW uses a bin-cell as a computing unit.

Fig. 3. An example to compute the distance between $X = \{\langle 1,7\rangle, \langle 3,3\rangle, \langle 7,4\rangle, ...\}$ and $Q = \{\langle 2,2\rangle, \langle 3,5\rangle, \langle 5,6\rangle\}$ (a) SPRING obtains the optimal subsequence $X[2:7]$ with distance 12. (b) Bin cell $C(3,1)$ is composed of $4 \times 2 = 8$ original-cells. WDTW computes nine bin-cells in this example.

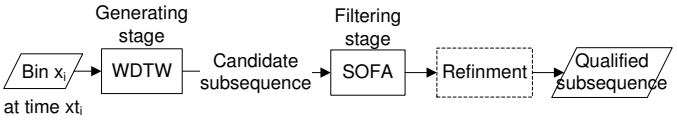


Fig. 4. The subsequence matching framework *SUMMER* preforms when each bin x_i arrives at time t_{x_i}.

refinement module to refine the answers. The framework structure is illustrated in Fig. 4.

In an online streaming environment, the bins of the data stream are generated continuously. When an incoming bin x_i arrives at time t_{x_i}, the bin is processed in the generating stage first. In this stage, SUMMER produces a candidate subsequence that is the optimal subsequence ended at the endpoint of current bin, i.e., t_{x_i}. We propose Weighted Dynamic Time Warping Distance (WDTW) to efficiently find the candidate subsequences and their distances to the query.

Next, the candidate subsequence is processed in the filtering stage, which applies a Stack-based Overlap Filtering Algorithm (SOFA) to prune the candidate subsequences violating the similarity or non-overlapping constraints. SOFA guarantees to find all qualified subsequences with no dismissals. Finally, we provide an optional refinement module to further improve the accuracy of qualified subsequences. The refinement recomputes the DTW distance between the reconstructed sequences from the synopsis query pattern and the qualified subsequences reported by SOFA. The distance can be computed more accurate and the starting and ending points of qualified subsequences can be relaxed to any timestamps.

3 Weighted Dynamic Time Warping Algorithm

In this section, we propose Weighted Dynamic Time Warping (WDTW) algorithm in the generating stage. When a new bin $\langle t_{x_i}, v_{x_i}\rangle$ arrives, WDTW generates a subsequence with the minimum distance to the query among all subsequences ending at timestamp t_{x_i}. This subsequence is called the optimal subsequence at the certain timestamp. The goal of WDTW is to generate a series of the optimal subsequences ending at different endpoints of bins as candidate subsequences. The formal description is as follows.

Definition 6 (Optimal subsequence and Candidate subsequence). *An optimal subsequence at timestamp t_{x_k} is defined as the subsequence with the minimum distance to the query among all subsequences ending at timestamp t_{x_k}. Formally speaking, $X[t^*_{x_i} : t_{x_k}]$ is an optimal subsequence at timestamp t_{x_k}* **iff** *$Dtw(X[t^*_{x_i} : t_{x_k}], Q) \leq Dtw(X[t_{x_i} : t_{x_k}], Q)$, where t_{x_i} is an endpoint of a bin which comes before bin x_k, that is, $t_{x_i} < t_{x_k}$. Then, the optimal subsequence $X[t^*_{x_i} : t_{x_k}]$ will be identified as a candidate subsequence at time t_{x_k} for the online subsequence matching.* □

The proposed WDTW is motivated by SPRING [14]. We first follow the streaming idea of SPRING for online environments. Then, we extend the point-matching of SPRING to the bin-matching in order to take the advantage of the data size reduction of the stream synopses. Specifically, we know that synopsis streams X and Q are composed of many constant pieces in a synopsis subsequence matching. When SPRING matches the data points within two sequences of bins x_i and q_j, the same value $\| v_{x_i} - v_{q_j} \|$ will be computed many times since data points within one bin have the same value. This motivates us to compute the DTW distance between two synopsis streams by matching their bins weighted by their time span rather than matching their values at each point. Instead of filling up each cell in the accumulated distance table by SPRING, WDTW compute one accumulated distance for a set of cells indexed by two bins, one from X and one from Q. In other words, WDTW replaces cells and accumulated distance tables in SPRING with *bin-cells* and *accumulated distance bin tables*, respectively. The formal definitions are as follows.

Definition 7 (Bin-cell). *The $C_{i,j}$ denotes a bin-cell which is a matching bin pair determined by the i^{th} bin of the synopsis stream X and the j^{th} bin of the query pattern Q. A bin-cell $C_{i,j}$ contains $l_{x_i} \times l_{q_j}$ original-cells, where l_{x_i} and l_{q_j} are the widths of bins x_i and q_j, respectively. Each original-cell within one bin-cell has the same distance $\| v_{x_i} - v_{q_j} \|$, which is denoted by $d_{i,j}$.* □

Definition 8 (Accumulated distance bin table). *An accumulated distance bin table D is used to obtain the distance between the query pattern and the optimal subsequence at timestamp t. Specifically, let $X[t_{x_s} : t_{x_i}]$ be the optimal subsequence at timestamp t_{x_i}, then $D(i,j)$ represents the approximated DTW distance between the reconstructed subsequences $X'[t_{x_s} : t_{x_i}]$ and $Q'[1 : t_{q_j}]$. The $D(i,j)$ value equals the distance accumulated from $C_{s,1}$ to $C_{i,j}$.* □

In Section 3.1, we first introduce the basic idea of WDTW using bin-cells to efficiently approximate the DTW distance between the reconstructed sequences from the query and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

synopsis stream. We further propose the bin-cell mergence technique to improve the accuracy of DTW distance in WDTW, as illustrated in Section 3.2.

3.1 Basic Ideas of Weighted Dynamic Time Warping

The essence of WDTW is to compute the DTW distance between the query synopsis and the data synopsis streams by matching their bins directly and its answers approximate to the ones by matching their points. Since the objective of SUMMER is to deal arbitrary-width synopsis streams, the bin-cells should be assigned different weights when we approximate the DTW distance.

We first consider a simple case to estimate the weight of a bin-cell. When we try to match x_i and q_j, the distance of this matching should be $d_{i,j} \times w_{i,j}$, where $w_{i,j}$ is the weight of $C_{i,j}$ and $d_{i,j}$ is the distance of original-cell within $C_{i,j}$. A matching can be represented by a warping path defined as follows.

Definition 9 (Warping Path and Optimal Warping Path)**.** *A warping path is a series of matching pairs (original-cells) between two sequences using point-matchings. A legal warping path should follow the boundary and continuous conditions in DTW [11]. An optimal warping path is the path with the minimum total distance, that is, the summation of distances of original-cells the warping path pass.*

In order to approximate the distance computed using point-matchings, the weight of a bin-cell should be the number of original-cells passed by the optimal warping path when computing the DTW distance between two reconstructed subsequences. We show how to compute the weight of each bin-cell in the following Lemma.

Lemma 1. *The weight of bin-cell $C_{i,j}$, which is the number of original-cells passed by the optimal warping path, is $\max\{l_{x_i}, l_{q_j}\}$, where l_{x_i} and l_{q_j} are the widths of bin x_i and bin q_j, respectively.*

Proof. Suppose P is a warping path formed by matching reconstructed x'_i and q'_j. We assume P passes w original-cells in $C_{i,j}$, where $w \in \mathbb{N}$. A legal warping path must start at the first original-cell $(1,1)$ and end at the last original-cell (l_{x_i}, l_{q_j}) in $C_{i,j}$ according to the DTW boundary condition [11]. P^* denotes a shortest path with the minimum length among all P. Under the boundary and shortest path conditions, the number of original-cells the path P^* will pass is $w^* = \max\{l_{x_i}, l_{q_j}\}$. Since P^* is the shortest path, we have $w^* \leq w$. Therefore, $d_{i,j} \times w^* \leq d_{i,j} \times w$ and we conclude that P^* is also an optimal warping path in $C_{i,j}$ and the weight of $C_{i,j}$ is $w^* = \max\{l_{x_i}, l_{q_j}\}$ according to P^*. □

To compute the DTW distance between a candidate subsequence and Q, we accumulate all distances of the bin-cells passed by the optimal warping path P using a bin-cell as the matching unit, that is, $\sum_{C_{i,j} \in P} d_{i,j} \times \max\{l_{x_i}, l_{q_j}\}$. The optimal warping path can be retrieved using dynamic programming as follows.

Under the bin-matching scenario, the recursive form of the accumulated distance in WDTW is as follows.

$$D(i,j) = \begin{cases} 0, & \text{if } j = 0, \\ \infty, & \text{if } i = 0, j \neq 0, \\ d_{i,j} \times \max\{l_{x_i}, l_{q_j}\} + \min\{D(i,j-1), & \\ \quad D(i-1,j), D(i-1,j-1)\}, & \text{otherwise,} \end{cases} \quad (1)$$

(a) independent (b) merged

Fig. 5. (a) The total weight of the two bin-cells is eight by counting the shaded original-cells if the two bin-cells are treated independently. (b) The total weight becomes four if we do the bin-cell mergence.

where $i = 0, 1, 2, ...$ and $j = 0, 1, ..., m$.

Meanwhile, each starting position $P(i,j)$ can be obtained from the starting position of the cell with the minimum distance among the three possible choices, $P(i, j-1)$, $P(i,j)$ and $P(i,j-1)$, where $P(t_{x_i}, 0) = t_{x_{i-1}} + 1$, which is the starting timestamp of the bin x_i. After WDTW gets $D(i,j)$ and $P(i,j)$ for all j at time t_{x_i}, $X[P(i,m) : t_{x_i}]$ will be reported as a candidate subsequence with the corresponding accumulated distance $D(i,m)$ to the query. The details of the starting position can be fond in SPRING [14].

Example 3. *In Fig. 3(b), we assume that all $D(i,j)$ have been calculated by Eq. (1) except for $D(3,3)$. Then, $D(3,3)$ can be calculated as:*

$$\begin{aligned} D(3,3) &= min\{D(3,2), D(2,3), D(2,2)\} \\ &\quad + d_{3,3} \times \max\{l_{x_3}, l_{q_3}\} \\ &= min\{6, 24, 6\} + (4-6)^2 \times 4 = 22. \end{aligned}$$

$P(3,3) = P(2,2) = P(1,2) = P(0,2) = t_{x_1} + 1 = 2$. Thus, the optimal subsequence at time t_{x_3} is $X[P(3,3) : t_{x_3}] = X[2 : 7]$ with the distance 22 to the query. □

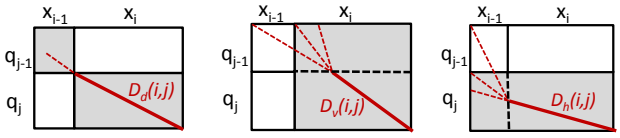
3.2 Weighted Dynamic Time Warping with Bin-Cell Mergence

Nevertheless, the DTW distance may be overestimated if we naively accumulate the distance of each bin-cell independently. An example is given in Fig. 5(a), the weight of each bin-cell is four and the corresponding accumulated distance is $D(1,2) = d_{1,1} \times 4 + d_{1,2} \times 4$. However, there are redundant horizontal movements in this warping path, which passes through the bin-cells in the same column twice. If the movement through $C_{1,1}$ and $C_{1,2}$ can be considered together, a better path with a smaller accumulated distance can be found as shown in Fig. 5(b). The two bin-cells $C_{1,1}$ and $C_{1,2}$ can share the four horizontal movements. Thus, the new path passes through only four original-cells in total and its corresponding accumulated distance is $d_{1,1} \times 2 + d_{1,2} \times 2$.

To improve the accuracy of bin-cell weight estimations, we consider to merge the distance computation and allow crossed bin matchings. Meanwhile, the time complexity should be kept the same as the basic version in Section 3.1, that is, $O(1)$ in one bin-cell distance computation. To do that, we design a bin-cell mergence technique, which is to eliminate the redundant horizontal/vertical movements by merging adjacent bin-cells as follows.

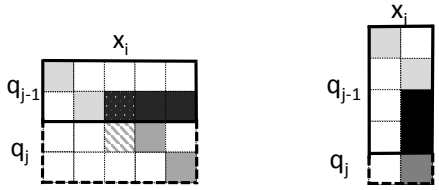
When computing $D(i,j)$, we consider three possible directions: vertical, horizontal, and diagonal, as illustrated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering



(a) From diagonal cell $C_{i-1,j-1}$ (b) From vertical cell $C_{i,j-1}$ (c) From horizontal cell $C_{i-1,j}$

Fig. 6. The estimated accumulated distances of $C_{i,j}$ from the cells of three directions



(a) can be redistributed (b) cannot be redistributed

Fig. 7. There are two cases merging vertically. (a) The extra horizontal movements (black original-cells) can be redistributed to $C_{i,j}$. (b) The extra vertical movements cannot be redistributed to $C_{i,j}$.

in Fig. 6 and denote the three corresponding estimated accumulated distances as $D_d(i,j)$, $D_v(i,j)$, and $D_h(i,j)$. After computing the three ones, WDTW will choose the smallest one as $D(i,j)$.

$$D(i,j) = \min\{D_d(i,j), D_v(i,j), D_h(i,j)\}. \quad (2)$$

$D_d(i,j)$ is the accumulated distance of the path coming from the diagonal bin-cell $C_{i-1,j-1}$. We cannot merge $C_{i-1,j-1}$ and $C_{i,j}$ since they are not adjacent. Thus, its computation is similar to Eq. (1) as follows.

$$D_d(i,j) = D(i-1,j-1) + d_{i,j} \times max\{l_{x_i}, l_{q_j}\}. \quad (3)$$

For the computations of D_v and D_h, they may involve the bin-cell mergence in the vertical and horizontal directions, respectively. We first show the computation of D_v, and that of D_h can be done in the same manner. When we compute D_v, the mergence of two vertical adjacent bin-cells has two cases. Let us see the examples shown in Fig. 7. In the first case in Fig. 7(a), the movements passing through the three black original-cells in $C_{i,j-1}$ may be redundant if $C_{i,j}$ is considered together with $C_{i,j-1}$. In other words, these movements are extra and can be adjusted to the next vertical bin-cell $C_{i,j}$ to be merged. When we compute the accumulated distance, the three extra horizontal movements distributed to $C_{i,j}$ can be absorbed by the height of bin-cell $C_{i,j}$, which is $l_{q_j} = 2$. Thus, one more horizontal movement is required. This extra one horizontal movement will be considered to be redistributed to one of the bin-cells which are merged in the same column, i.e., either $C_{i,j}$ or $C_{i,j-1}$, depending on whose $d_{i,j}$ is the smallest. We assume $d_{i,j} < d_{i,j-1}$ in this example. Thus, the extra one horizontal movement is distributed to $C_{i,j}$, which is shown as the original-cell with slash lines in $C_{i,j}$. In the second case in Fig. 7(b), there is no extra horizontal movements in $C_{i,j-1}$ can be distributed to $C_{i,j}$ since the number of horizontal movements is smaller than that of vertical movements shown in $C_{i,j-1}$. Thus, there is no redundant extra movement can be reduced when merging two vertical bin-cells $C_{i,j-1}$ and $C_{i,j}$. By the way, the extra vertical movements in $C_{i,j-1}$, that is, the two black original-cells, can be distributed to the next horizontal bin-cell $C_{i+1,j}$ in the later computation, which is symmetric to handling the extra horizontal movements.

With the concept of bin-cell mergence, the computation of D_v is as follows.

$$D_v(i,j) = D(i,j-1) - A_v(i,j-1) + d_{i,j} \times l_{q_j} + A_v(i,j), \quad (4)$$

where $A_v(i,j-1)$ denotes the *vertically adjustable distance* contributed by the extra horizontal movements in $C_{i,j-1}$ and can be redistributed to the next vertical bin-cell $C_{i,j}$. Similarly, $A_v(i,j)$ is the vertically adjustable distance of $C_{i,j}$ that can be redistributed to $C_{i,j+1}$. For the example shown in Fig. 7(a), $A_v(i,j)$ is represented as the three black original-cells in $C_{i,j-1}$ and $A_v(i,j)$ is the one with slash lines in $C_{i,j}$. However, for the example in Fig. 7(b), $A_v(i,j-1)$ and $A_v(i,j)$ are both zero because there is no extra horizontal movements in $C_{i,j-1}$ and $C_{i,j}$. In both cases, the vertical movements are the same and counted as $d_{i,j} \times l_{q_j}$ in Eq. (4).

To compute $A_v(i,j)$, we first compute the number of extra horizontal movements that can be distributed to the next vertical bin-cell $C_{i,j+1}$, denoted by $N_h(i,j)$, and the corresponding distance of these movements $dm_v(i,j)$. Then, $A_v(i,j) = dm_v(i,j) \times N_h(i,j)$. The $N_h(i,j)$ value equals the number of the horizontal movements more than that of the vertical movements in $C_{i,j}$. When we merge $C_{i,j-1}$ vertically with $C_{i,j}$, only $N_h(i,j-1)$ horizontal movements are remained in the horizontal direction of $C_{i,j}$. On the other hand, the vertical movements is still the whole height of $C_{i,j}$, that is l_{q_j}. As a result,

$$N_h(i,j) = \max\{N_h(i,j-1) - l_{q_j},\ 0\}. \quad (5)$$

For example, in Fig. 7(a), suppose $N_h(i,j-1) = 3$. Then $N_h(i,j) = \max\{3-2,\ 0\} = 1$. Thus, $C_{i,j}$ has one extra horizontal movements that can be distributed to $C_{i,j+1}$. In contrast, $N_h(i,j) = \max\{0-1,\ 0\} = 0$ in Fig. 7(b), which means $C_{i,j}$ has no extra horizontal movements can be distributed to $C_{i,j+1}$. To find $dm_v(i,j)$, we need to find the bin-cell having the smallest distance $d_{m,n}$ among all bin-cells vertically merged in the same column until $C_{i,j}$. Thus, we have this recursive form

$$dm_v(i,j) = \min\{dm_v(i,j-1),\ d_{i,j}\}. \quad (6)$$

In Fig. 7(a), the one extra horizontal movement, i.e, $N_h(i,j) = 1$, can be counted in either $C_{i,j-1}$ or $C_{i,j}$, depending on the value of $d_{i,j-1}$ and $d_{i,j}$ as we mentioned earlier. If $d_{i,j} < d_{i,j-1}$, $dm_v(i,j) = d_{i,j}$ and the $A_v(i,j)$ is represented as the original-cell with slash lines. Otherwise, we have $dm_v(i,j) = d_{i,j-1}$ such that $A_v(i,j)$ is represented as the white-dotted original-cell in $C_{i,j-1}$. From Eq. (5) and (6), $A_v(i,j)$ in Eq. (4) is computed as follows.

$$A_v(i,j) = \min\{dm_v(i,j-1),\ d_{i,j}\} \times \max\{N_h(i,j-1) - l_{q_j},\ 0\}. \quad (7)$$

Symmetrically, to compute $D_h(i,j)$, we need to consider if the *horizontal adjustable distance* of $C_{i-1,j}$ can be distributed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

to the next horizontal bin-cell $C_{i,j}$, which is denoted by $A_h(i,j)$. We have

$$D_h(i,j) = D(i-1,j) - A_h(i-1,j) + d_{i,j} \times l_{x_i} + A_h(i,j), \quad (8)$$

where

$$A_h(i,j) = \min\{dm_h(i-1,j),\ d_{i,j}\} \times \max\{N_v(i-1,j) - l_{x_i},\ 0\}. \quad (9)$$

The $D(i,j)$ in Eq. (2) can be derived from $D_d(i,j)$, $D_v(i,j)$, and $D_h(i,j)$ based on Eq. (3), (4), and (8).

After $D(i,j)$ is determined, we know what direction the optimal path comes from. We shall then update $N_h(i,j)$, $N_v(i,j)$, $dm_v(i,j)$, $dm_h(i,j)$, $A_v(i,j)$, and $A_h(i,j)$ for later bin-cell mergences since they are originally computed under different assumptions of where the warping path comes from. The update of the number of extra horizontal and vertical movements $N_h(i,j)$ and $N_v(i,j)$ is as follows.

$$N_h(i,j) = \begin{cases} \max\{l_{x_i} - l_{q_j},\ 0\}, & \text{if PD,} \\ \max\{N_h(i,j-1) - l_{q_j},\ 0\}, & \text{if PV,} \\ \max\{l_{x_i} - N_v(i-1,j),\ 0\}, & \text{if PH, and} \end{cases} \quad (10)$$

$$N_v(i,j) = \begin{cases} \max\{l_{q_j} - l_{x_i},\ 0\}, & \text{if PD,} \\ \max\{l_{q_j} - N_h(i,j-1),\ 0\}, & \text{if PV,} \\ \max\{N_v(i-1,j) - l_{x_i},\ 0\}, & \text{if PH,} \end{cases} \quad (11)$$

where PD, PV, and PH represent the three conditions: the path coming diagonally from $C_{i-1,j-1}$, coming vertically from $C_{i,j-1}$, and coming horizontally from $C_{i-1,j}$, respectively. In general, $N_h(i,j)$ is the extra number of movements in the horizontal direction compared to the vertical movements in $C_{i,j}$. In contrast, $N_v(i,j)$ is the extra number of movements in the vertical direction compared with the horizontal movements. Thus, if the horizontal movements are more than the vertical movements in the bin-cell $C_{i,j}$, $N_h(i,j) > 0$ and $N_v(i,j) = 0$. If there are more vertical movements, then $N_v(i,j) > 0$ and $N_h(i,j) = 0$. When the warping path is from the diagonal bin-cell, which means no bin-cell mergence is allowed, the $N_h(i,j)$ and $N_v(i,j)$ values are reset to the difference between the lengths of bin x_i and bin q_j of $C_{i,j}$. This is because the extra movements accumulated until $C_{i-1,j-1}$ cannot be propagated to $C_{i,j}$ either horizontally or vertically. When the warping path is from the vertical or horizontal bin-cell, only the number of movements that are distributed to $C_{i,j}$ should be counted in the merging direction. Specifically, when merging vertically, the warping path only needs $N_h(i,j-1)$ horizontal and l_{q_j} vertical movements to pass through $C_{i,j}$. Thus, we have $N_h(i,j) = \max\{N_h(i,j-1) - l_{q_j},\ 0\}$ and $N_v(i,j) = \max\{l_{q_j} - N_h(i,j-1),\ 0\}$. The case of merging vertically is in the same manner.

The $dm_v(i,j)$ and $dm_h(i,j)$ are updated as follows.

$$dm_v(i,j) = \begin{cases} d_{i,j}, & \text{if PD,} \\ \min\{dm_v(i,j-1),\ d_{i,j}\}, & \text{if PV,} \\ d_{i,j}, & \text{if PH,} \end{cases} \quad (12)$$

$$dm_h(i,j) = \begin{cases} d_{i,j}, & \text{if PD,} \\ d_{i,j}, & \text{if PV,} \\ \min\{dm_h(i-1,j),\ d_{i,j}\}, & \text{if PH,} \end{cases} \quad (13)$$

where PD, PV, and PH have the same meaning as in Eq. (10) and (11). Along the merging direction, either vertical or horizontal, the smallest distance should be kept. In contrast, for the non-merging direction including the diagonal direction, the current bin-cell $C_{i,j}$ becomes the first cell for the later mergence, so $dm_v(i,j)$ and $dm_h(i,j)$ both equal $d_{i,j}$.

We recompute $A_v(i,j)$ and $A_h(i,j)$ by new $N_h(i,j)$, $N_v(i,j)$, $dm_v(i,j)$, and $dm_h(i,j)$. No matter where the path comes from, we have

$$A_v(i,j) = dm_v(i,j) \times N_h(i,j), \text{ and} \quad (14)$$

$$A_h(i,j) = dm_h(i,j) \times N_v(i,j). \quad (15)$$

At this point, we finish all the computation of $C_{i,j}$ and continue to process the next bin-cell. The computation is terminated when we reach $C_{i,m}$, where m is the length of the query pattern. Compared to the method in Section 3.1, only the computation of $D(i,j)$ is changed. The candidate subsequence and the corresponding accumulated distance $D(i,m)$ are obtained in the same way.

To illustrate how to compute the accumulated distances in WDTW with bin-cell mergence, we follow Example 3 in Fig. 3(b) and compute the same accumulated distance $D(3,3)$.

Example 4. *Since the path comes from $C_{2,1}$ diagonally to $C_{3,2}$, we have $N_h(3,2) = \max\{l_{x_3} - l_{q_2},\ 0\} = 3$ and $dm_v(3,2) = d_{3,2} = 1$. Then, $A_v(3,2) = dm_v(3,2) \times N_h(3,2) = 3$. If the path comes from $C_{3,2}$ vertically to $C_{3,3}$, $A_v(3,3) = min\{dm_v(3,2),\ d_{3,3}\} \times \max\{N_h(3,2) - l_{q_3},\ 0\} = 1 \times 1 = 1$, and $D_v(3,3) = D(3,2) - A_v(3,2) + d_{3,3} \times l_{q_3} + A_v(3,3) = 6 - 3 + 4 \times 2 + 1 = 12$. Similarly, we have $D_h(3,3) = D(2,3) - A_h(2,3) + d_{3,3} \times l_{x_3} + A_h(3,3) = 22$ and $D_d(3,3) = D(2,2) + d_{3,3} \times \max\{l_{x_3}, l_{q_3}\} = 22$. Finally, $D(3,3) = \min\{D_d(3,3), D_v(3,3), D_h(3,3)\} = 12$, and $C_{3,3}$ are merged in the vertical direction.* □

The value of $D(3,3)$ computed by WDTW with the bin-cell mergence is 12. This value is smaller than the one computed by WDTW without the bin-cell mergence, which is 22 as shown in Example 3. With these two examples, we know that the bin-cell mergence can solve the distance overestimate problem effectively and derive a more accurate accumulated DTW distance to the query.

4 Candidate Subsequence Filtering and Refinement

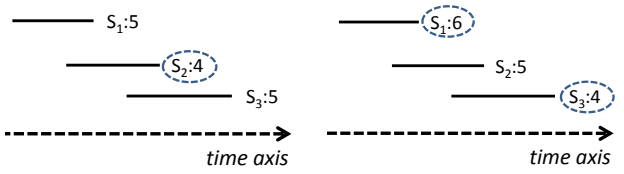
The candidate subsequences found in the generating stage may overlap. Usually, the overlapped subsequences are similar to each other. They may be with much redundant information since they may indicate the same time periods with little time shift. To remove redundant overlapping ones, we propose Stack-based Overlap Filtering Algorithm (SOFA) to online retrieve a set of non-overlapping qualified subsequences and keep the latency of each reported subsequence as short as possible.

4.1 The Stack-based Overlap Filtering Algorithm

SOFA aims to retrieve a set of non-overlapping subsequences according to the maximal non-overlapping subsequence set in Definition 4 from all the candidates obtained in the generating stage. Let us use an example to illustrate the idea of non-overlapping subsequences.

Example 5. *In Fig. 8(a), there are three overlapping candidate subsequences, S_1, S_2 and S_3, with distances to the query as 5,*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering



(a) S_2 is reported (b) S_1 and S_3 are reported

Fig. 8. Two examples of subsequences overlap with the distance threshold $\epsilon = 10$.

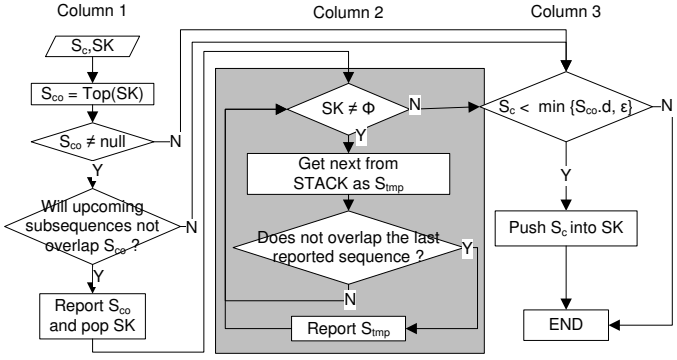


Fig. 9. The flow chart of SOFA

4, *and* 5, *respectively.* S_1 *overlaps* S_2, S_2 *overlaps* S_3, *but* S_1 *does not overlap* S_3. *Apparently,* S_2 *should the only one to be retained because it has the smallest distance and that is smaller than* $\epsilon = 10$. *Since both* S_1 *and* S_3 *overlap* S_2 *and have larger distances compared to* S_2, *they should be removed.*

Let us consider another situation in Fig. 8(b), where the overlaps among S_1 *to* S_3 *are the same as that in Fig. 8(a) but with difference distances, which are* 6, 5, *and* 4, *respectively, to the query. The* S_3 *must be reported since it has the minimum distance. Since* S_2 *overlaps* S_1 *and its distance is larger than that of* S_1, S_2 *should be removed. However,* S_1 *in fact should be reported since 1)* S_1 *overlaps only* S_2, *which is not reported, and 2) the distance of* S_1 *is smaller than the distance threshold. Thus,* S_1 *and* S_3 *are reported as qualified subsequences.* □

It is worth mentioning that SPRING [14] also aimed to report non-overlapping subsequences for the online subsequence matching. It applied a delaying and replacing method to do the filtering, that is, the optimal subsequence will not be reported until it is ensured that no more possible future candidates will overlap it. If the distance to the query of the current subsequence is smaller than that of the optimal one, the current subsequence will replace as the new optimal one. However, a qualified subsequence may be missed in this procedure in some cases. In Fig. 8(b), for example, S_1 is replaced by S_2 and then S_2 is replaced by S_3 using the delaying and replacing method of SPRING. Only S_3 will be reported, yet S_1 is also a qualified subsequence under the similarity and non-overlapping constraints of Definition 5.

Inspired by this observation, we propose SOFA, a Stack-based Overlap Filter Algorithm, which is used a stack to store possible qualified subsequences, to avoid the missing qualified subsequence problem. The flow and pseudo code of SOFA are shown in Fig. 9 and Algorithm SOFA, respectively. When WDTW produces the latest candidate subsequence S_c, SOFA is performed to check which subsequences should be reported. Suppose S_{co} is the current optimal subsequence, which has the minimum distance among the subsequences not decided to be reported yet at that time. A stack *K* is used to store the subsequences that were ever the optimal subsequences but have not been decided whether reported yet.

Algorithm SOFA

Input: S_c, D, and P /* *K* is a static variable */
Output: qualified subsequence if any
1: S_{co} = top(*K*);
2: **if** $(S_{co} \neq \text{null}) \wedge (\forall_j, P[j] > S_{co}.t_e \vee D[j] \geq S_{co}.d)$ **then**
3: /* The upcoming subsequences will not replace S_{co}*/
4: **if** $\exists_j, P[j] \leq S_{co}.t_e$ **then**
5: /* Remove the upcoming subsequences which overlap S_{co}*/
6: $D[j] = \infty$; S_c= null;
7: **end if**
8: Report S_{co}; $S_{last} = S_{co}$; $d_{min} = \epsilon$; Pop *K*.
9: **while** $|K| > 0$ **do**
10: S_{tmp} = pop *K* ;
11: **if** $S_{tmp}.t_e < S_{last}.t_s$ **then**
12: Report S_{tmp}; $S_{last} = S_{tmp}$;
13: **end if**
14: **end while**
15: **end if**
16: **if** $S_c.d < d_{min}$ **then**
17: Push S_c to *K*;
18: $d_{min} = S_c.d$;
19: **end if**

SOFA is composed of three parts, which are shown as three columns in Fig. 9. In the first part, the first column in Fig. 9, SOFA examines whether S_{co} should be reported. If the upcoming subsequences do not replace S_{co} in Line 2 of Algorithm SOFA, S_{co} will be reported. The correctness of this condition will be proved later in Lemma 2. In the second part, the second column in Fig. 9, the loop is designed for sequentially checking whether each subsequence in *K* overlaps the last reported subsequence in Lines $9-14$ of Algorithm SOFA. Once a qualified subsequence is determined in the first part, all the subsequences in *K* will be determined whether they are qualified. Each of them is popped out one by one as S_{tmp} to be examined whether it overlaps the last reported qualified subsequence S_{last}. If S_{tmp} does not overlap S_{last}, S_{tmp} will be reported. The loop continues until *K* is empty. In the last part, the third column in Fig. 9, SOFA decides whether the latest candidate subsequence S_c can replace S_{co}. If the distance of S_c is smaller than the current minimum distance ($\min\{S_{co}.d, \epsilon\}$), S_c will be push into *K* and will become a new S_{co}.

4.2 Correctness of SOFA

With the clear view of how SOFA works, we now prove its correctness. Lemma 2 proves that the reported subsequences satisfy the condition of the maximal non-overlapping subsequence set in Definition 4; Lemma 3 proves that the removed subsequences do not satisfy the condition of the maximal non-overlapping subsequence set. Combining these two lemmas, we can prove the correctness of SOFA in Theorem 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

Lemma 2. *The current optimal subsequence $S_{co} = X[t_s : t_e]$ with a distance d to the query Q can be reported as a qualified subsequence if $\forall j, P[j] > t_e$ or $D[j] \geq d$, where $P[j]$ and $D[j]$ refer to the starting position $P(i,j)$ and the accumulated distance $D(i,j)$ at time t_{x_i}, respectively.*

Proof. S_{co} is the subsequence with a smaller distance than those of the subsequences that overlap it and are generated before time t_{x_i} according to the condition of stack pushing in Line 16 of Algorithm SOFA. If the upcoming subsequences will not replace S_{co}, it can be guaranteed that S_{co} is the optimal subsequence, whose distance is smaller than those of all the subsequences overlapping it.

Assuming the warping path of an upcoming subsequence S_u passes through $C_{i,j}$, we know that the starting position of S_u is $P[j]$ and its distance to the query pattern Q must be larger or equal to $D[j]$ since its distance is accumulated from $D[j]$. S_u cannot replace S_{co} unless its starting position $P[j]$ is preceding or the same as the end point of S_{co} and also its distance is smaller than that of S_{co}. Thus, if $P[j] > t_e$ or $D[j] \geq d$, S_u will not replace S_{co}. All possible subsequences that overlap S_{co} must pass one of $C(i,j)$ for all j, but they will not replace S_{co} under the above condition. Therefore, S_{co} passes the non-overlapping constraint. Also, S_{co} must pass the similarity constraint in Line 16 of Algorithm SOFA. Thus, S_{co} can be reported as a qualified subsequence. □

Note that SOFA should remove the upcoming subsequences that overlap S_{co} when S_{co} is reported.

In Lemma 3, we show that the subsequences are correctly filtered out.

Lemma 3. *The subsequences filtered out by SOFA do not pass the similarity and non-overlapping constraints of Definition 5.*

Proof. A candidate subsequence is filtered out by SOFA only if it is in one of the following three conditions of Algorithm SOFA.

First, in Lines $2-4$, if ($\forall_j, P[j] > S_{co}.t_e$ or $D[j] \geq S_{co}.d$) and ($\exists_j, P[j] \leq S_{co}.t_e$), then $d_i = \infty$ and $S_c = null$. When S_{co} is reported, that is, $S_{co} \in R$, the upcoming subsequences that overlap S_{co} are removed and so is S_c. They are removed since they are not in the maximal non-overlapping set in Definition 4.

Second, in Lines $9-14$, if $S_{tmp}.t_e \leq S_{last}.t_s$, then SOFA discards S_{tmp}. When the last subsequence S_{last} is reported, that is, $S_{last} \in R$, the subsequences S_{tmp} that are older than S_{last} in K and overlap S_{last} are removed since their distances must be larger than that of S_{last} according to the condition in Line 16. Thus, S_{tmp} is removed since it is not in the maximal non-overlapping subsequence set.

Third, in Line 16, if $S_c.d \geq d_{min}$, then SOFA discards S_c. If $d_{min} = \epsilon$, S_c cannot pass the similarity constraint. In another situation, if $d_{min} = S_{co}.d$, S_c must overlap S_{co}. Otherwise, S_{co} should have been reported and d_{min} is reset as ϵ in Line 8.

In this situation of S_c overlapping S_{co}, if S_{co} is reported, $S_{co} \in R$ and $S_c \geq S_{co}.d$. Thus, S_c is not in the non-overlapping subsequence set. On the other hand, if S_{co} is not reported, there must be a newer subsequence $S' \in R$ and S' replaces S_{co}. Since S' is newer than S_c and S_c is newer than S_{co}, S' also overlaps S_c. In addition, $S'.d < S_{co}.d < S_c.d$. Therefore, S_c is not in the maximal non-overlapping subsequence set. □

Algorithm SUMMER

Input: a new bin x_i at time t_{x_i}
1: **for** j = 1 to m **do**
2: Compute $D[j]$ and $P[j]$ by WDTW;
3: **end for**
4: $S_c = X[P[m] : t_{x_i}]$;
5: S_q = SOFA(S_c, D, P);
6: **if** $S_q \neq$ null **then**
7: S_q' = Refinement(S_q, Q);
8: **end if**
9: Substitute D' for D; Substitute P' for P;

To sum up, we prove the correctness of SOFA.

Theorem 1. *SOFA guarantees no false dismissals for finding the qualified subsequences of Definition 5 in an online process of receiving candidate subsequences from WDTW.*

Proof. With Lemma 2 and 3, SOFA can filter out the subsequences that do not satisfy the similarity and non-overlapping constraints and report all qualified subsequences defined in Definition 5. In addition, SOFA processes all candidate subsequences that are generated by WDTW. These candidate subsequences either are filtered out or are reported as qualified subsequences. Thus, SOFA guarantees no false dismissals for finding the qualified subsequences in an online process of receiving candidate subsequences generated by WDTW. □

4.3 Refinement of WDTW

According to the qualified subsequence $X[t_s : t_e]$ of Definition 5, t_s and t_e can be arbitrary timestamps only $t_s \leq t_e$. However, the candidate subsequences generated by WDTW always start and end at the endpoints of synopsis bins since WDTW computes the accumulate distances using a bin-cell as a unit. SUMMER may obtain the qualified subsequences more similar to the query pattern if we relax this restriction. Thus, we further offer an optional refinement process. The distance can be computed more accurate and the starting and ending points of qualified subsequences can be relaxed to any timestamps instead of being limited at the endpoints of bins.

The refinement module recomputes the DTW distance between the reconstructed subsequences $X'[t_s, t_e]$ and Q' using an original-cell as a unit like SPRING after a candidate subsequence $X[t_s : t_e]$ passes SOFA. The complexity of SUMMER will not increase if the number of refinement executions is not large. Since the length of the candidate subsequence is almost equal to M, which is the length of the query Q, the complexity of one refinement is only $O(M^2)$. In addition, the refinement and WDTW are computed in parallel. Specifically, when the refinement module computes the distance between S_i and Q, WDTW can produce the next candidate subsequence S_{i+1} at the same time. If the number of refinement executions is not large, the refinement process does not dominate the time cost of WDTW. In this situation, the complexity of SUMMER will not increase. The detail of complexity analysis is discussed in Section 5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

To sum up, the pseudo code of SUMMER, including WDTW, SOFA, and the refinement, is shown in Algorithm SUMMER. Because SUMMER is an online algorithm, it will be performed at a new bin x_i arriving. After WDTW computes the accumulated distance D and the starting position P, we obtain the candidate subsequence $S_c = X[P[m] : t_{x_i}]$ with corresponding accumulated distance $D[m]$, where m is the number of bins in the query Q. Then, SOFA determines that S_i and previous candidate subsequences should be reported or be removed. Finally, if there are qualified subsequences reported by SOFA, the refinement module will be performed on those qualified subsequences. Note that although D and P are originally designed for two-dimension arrays, only the data in the last column need to be preserved for the online environment. Thus, we record the data in one-dimension arrays, i.e., $D[j]$ and $P[j]$, which refer to the starting position $P(i, j)$ and the accumulated distance $D(i, j)$ at time t_{x_i}.

5 Complexity Analysis of SUMMER

Let X be an evolving synopsis stream composed of n bins to represent the original stream of length N and the synopsis rate b equals n/N. When $n \ll N$, which is the usual case, $b \ll 1$. The synopsis query pattern Q is constructed using the same synopsis rate in general. Thus, Q is composed about $m = bM$ bins for the reconstructed query pattern of length M.

The time and space complexity of WDTW with bin-cell mergence at each bin arrival are both $O(m) = O(bM)$. Since we keep only one-dimension array to store the accumulated distance D and the starting position P, and we also update each bin-cell in constant time, both the time and space complexity of WDTW are $O(bM)$. The time and space complexity of one refinement are both $O(M^2)$ as we mentioned in Section 4.3.

In summary, the time complexity of SUMMER to process subsequence matching in the synopsis stream X is $\max\{O(b^2MN), O(M^2R)\}$, where R is the number of refinement executions. Compared with SPRING, which computes DTW using an original-cell as a matching unit, the time complexity of processing X is $O(MN)$. We speed up the subsequence matching problem by at most $1/b^2$ times, where $b \ll 1$, especially when R is small.

6 Performance Evaluation

To demonstrate the performance of SUMMER, we conduct several experiments to evaluate the two parts of SUMMER: WDTW and SOFA. The goal of these following experiments is to test the following three hypotheses:

1) The throughput of WDTW is better than the compared algorithms within acceptable error.
2) The accuracy of WDTW and WDTW+R are the best two among all compared algorithms.
3) SOFA reports more number of qualified subsequences than SPRING-filter given the same candidate subsequences. In addition, SOFA reports these subsequences with minimum delays.

We compare WDTW and its effect with refinement, denoted by WDTW+R here, with four algorithms, Chan, PDTW, UCR-DTW, and SPRING-DTW, for online subsequence matchings. Chan [15] and PDTW [2] are the algorithms for similarity search of time series under the DTW distance based on histogram-based synopsis techniques. We adapt both of them to the online subsequence matching problem as SPRING. Since Chan and PDTW do subsequence matching bin by bin, their complexity is the same as that of WDTW. To our knowledge, Chan and our proposed WDTW are the only two algorithms that deal with the matching between subsequences of arbitrary-width bins under DTW. In contrast, PDTW is designed for equal-width bins. We compare with PDTW in order to show the differences in accuracy between arbitrary-width and equal-width synopses based on the same efficiency. To satisfy the input format of PDTW, we refine the data. Specifically, we adjust the data by equally distributing the total width of those arbitrary-width bins to each bin. We further modify PDTW using the same streaming technique as SPRING and WDTW. Through this modification, we speed up the efficiency of PDTW and the complexity of WDTW and PDTW becomes the same but the accuracy of PDTW is the same as original PDTW. We also compare WDTW with UCR-DTW because UCR-DTW is the state-of-the-art of subsequence matching for the raw stream. UCR-DTW is originally designed for the top one search [7]. We modify its pruning criteria to the similarity threshold ϵ, which is the same as the one used in other comparing algorithms. Finally, we compare WDTW with SPRING-DTW in order to show the efficiency improvement based on the fact that WDTW with little error. SPRING-DTW applies SPRING [14] on the stream reconstructed from the synopsis stream. Since SPRING-DTW is an exact algorithm for subsequence matching under the DTW distance, we treat the subsequences reported by SPRING-DTW as the ground truths for the accuracy measurement on finding candidate subsequences.

The comparison of SOFA is SPRING-filter, which is the filtering module in SPRING [14]. Both of their input candidate subsequences are produced by WDTW.

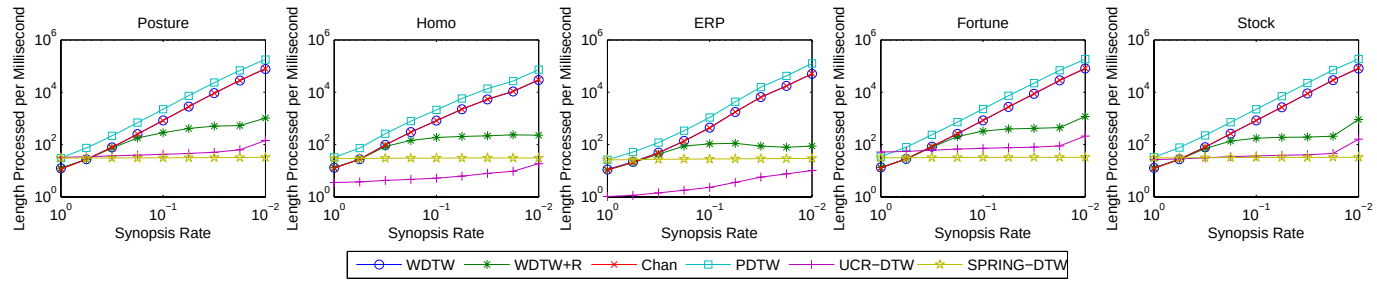
We conduct three experiments. The first two aim to demonstrate the throughput and accuracy of WDTW. In the last experiment, we examine the filtering quality and the latency of SOFA. All experiments are performed on five real stream datasets. The *Posture* dataset of length 33792 is downloaded from UCR Time Series Classification/Clustering Archive [16]. The *Homo*, *ERP*, *Fortune*, and *Stocks* datasets are obtained according to [17]. Their length are 10000, 6401, 14554, and 6481, respectively. All experiments are programmed in C++ and run on a PC with a 4-core 3.00GHz CPU and 8GB RAM.

6.1 Throughput and Accuracy of WDTW

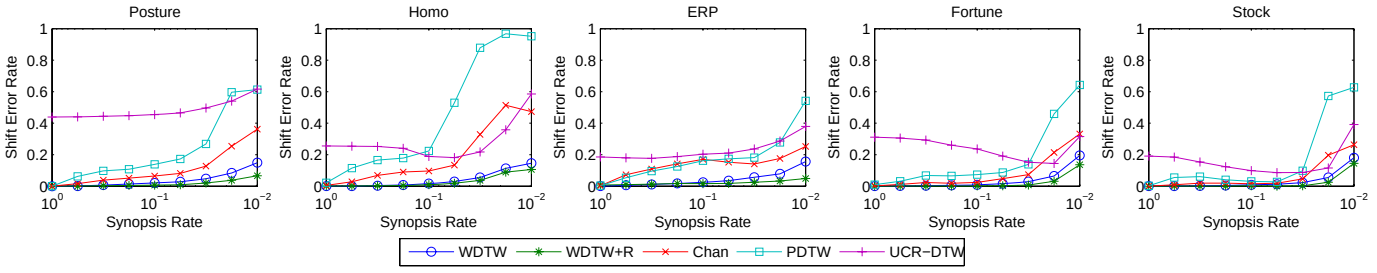
We first test Hypotheses 1 and 2 through the following experiments. We evaluate the performance of all methods on finding candidate subsequences using two metrics: throughput and accuracy. The *throughput* is measured by the amount of data processed per time unit. Specifically, it is a ratio of the raw stream length to the total execution time of matching a query over the entire stream.

$$\text{Throughput} = \frac{\text{raw stream length } N}{\text{execution time of matching}}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering



(a) Throughput



(b) Accuracy

Fig. 10. The throughput and accuracy of five algorithms at different synopsis rate b

On the other hand, the accuracy is measured by the *shift error rate*, which is the offset between the subsequences reported by the ground truth and various algorithms. Specifically, we define

$$\text{Shift Error Rate} \equiv 1 - \frac{\sum_{i,j} \text{overlap\_len}(G_i, S_j)}{(\sum_i \text{len}(G_i) + \sum_j \text{len}(S_j))/2},$$

where G_i and S_j are the subsequences reported by ground truth and each compared algorithm, respectively. The overlap_len(G_i, S_j) stands for the overlapping length of G_i and S_j in time axis[1].

We evaluate the influences of two parameters that are most significant to the throughput and accuracy of WDTW: the synopsis rate b and the distance threshold ratio λ. To consider different queries fairly, we use a distance threshold ratio λ to find the qualified subsequences in the similarity constraint of Definition 5 instead of an absolute value ϵ. We normalize a query sequence with respect to the largest difference of bin values in the query. That is, $Dtw(S,Q)/var < \epsilon / var = \lambda$, where *var* is the difference between the maximum bin value and the minimum bin value in the query sequence.

6.1.1 Synopsis Rate b

In the first experiment, we examine the throughput and accuracy of all methods at different *synopsis rates* b. The synopsis rate b is defined as a ratio of the number of the bins to the total length in time of the entire data stream, i.e., $b = n/N$. Note that both the data stream and the query stream are constructed using the same synopsis rate in the experiment. All values came from averaging the results of 1000 different query patterns, which are randomly chosen subsequences of length 1000 from the data stream, at a fixed distance threshold ratio for each dataset.

The throughput and accuracy of all five methods at different synopsis rates are shown in Fig. 10(a) and 10(b), respectively. The five subfigures show the results of the all algorithms on Posture, Homo, EPR, Fortune, and Stock datasets, respectively. Their distance threshold ratio for the five datasets are 1%, 1%, 5%, 20%, 20%, respectively.

We can see that the throughputs of the five methods on these five datasets had similar trends. The throughputs of WDTW, Chan, and PDTW increase when the synopsis rates decrease because they treat a bin-cell as a computing unit for the subsequence matching. When the synopsis rate is smaller than $10^{-0.25}$, where a bin represents about two original data points on average, WDTW, WDTW+R, Chan, and PDTW have higher throughputs compared to UCR-DTW and SPRING-DTW. For example, on the Posture dataset, when the synopsis rate is 10^{-2}, the throughputs of WDTW, WDTW+R, Chan, PDTW, and UCR-DTW are 2501, 28, 2617, 5747, and 5 times to that of SPRING-DTW. Chan and WDTW have almost the same throughputs at different synopsis rates. The throughput of PDTW is about three times to that of WDTW since PDTW only computes the current distance and need not to compute the estimated distances as WDTW. However, PDTW has unacceptable error since it forces arbitrary-width histogram-based synopses to be equal-width ones. The situation becomes worse when synopsis rate becomes small (Fig. 10(b)).

The throughput of WDTW+R shows a two-stage trend. Since the two processes, WDTW and the refinement, are executed in parallel in WDTW+R, the throughput of WDTW+R is dominates by the slower process. When the synopsis rate is larger than $10^{-0.75}$, WDTW takes more time compared with the refinement process. Thus, the throughput of WDTW+R is the same as that of WDTW. In contrast, when the synopsis rate is smaller than $10^{-0.75}$, the refine-

1. The ground truth G_i is based on the subsequence produced by SPRING-DTW on synopsis streams. Once the synopsis bins are determined, SPRING-DTW will not introduce any extra errors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

ment process takes more time than WDTW does. In this situation, WDTW+R has a lower throughput than WDTW. Even so, WDTW+R still outperforms UCR-DTW and SPRING-DTW on having a higher throughput. The throughputs of both SPRING-DTW and UCR-DTW are independent of the synopsis rate. This is because both methods match a query sequence to a data stream point-by-point. Thus, the number of data points of a stream to be matched is always equal to the original stream length no matter what synopsis rate is used.

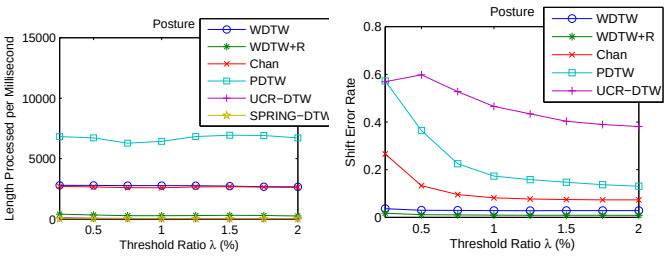
Furthermore, we observe that the relation between throughput and synopsis rate in Fig. 10(a) matches our time complexity analysis in Section 5 very well. Since the time complexity of WDTW is $O(b^2MN)$, where b is the synopsis rate, the throughput of WDTW should be proportional to b^{-2} in theory when M and N are fixed. On the Posture dataset of Fig. 10(a), the regression function of WDTW is $Throughput = 9.81b^{-1.96}$ with $R^2 = 0.998$. The R-squared value shows how well the regression line approximates the real data. $R^2 = 1$ means perfect match. This regression function ($b^{-1.96}$) is very close to our prediction of throughput (b^{-2}).

The accuracy of the five methods at different synopsis rates are shown in Fig. 10(b). WDTW+R and WDTW are always the best two algorithms on all five datasets. WDTW+R has the smallest shift error rates, which are never greater than 0.1 on all five datasets. The shift error rate of WDTW increased slightly with the synopsis rate decreasing. However, it is still much lower than those of Chan, PDTW, and UCR-DTW. This shows the bin-cell mergence technique of WDTW is able to efficiently improve its accuracy without reducing the throughput when compared with Chan. The shift error rate of PDTW significantly increases when the synopsis rate decreases since PDTW treats arbitrary-width histograms as equal-width ones. Lastly, the shift error rate of UCR-DTW (between 0.2 to 0.4 in all experiments) is much larger compared to the other algorithms. This is because UCR-DTW uses Sakoe-Chiba Band with width R to constrain the warping path. UCR-DTW can find the subsequences only within R tolerances in the length of query. Thus, the accuracy of UCR-DTW will be very low if we use the answer of SPRING-DTW as the ground truth.

We average the shift error rate over the five datasets. The shift error rate of WDTW is 0.016, which is 21%, 13%, and 7% of that of Chan, PDTW, and UCR-DTW at synopsis rate of 10^{-1}. The shift error rate of WDTW+R is 0.008, which is 10%, 6%, and 3% of that of Chan, PDTW, and UCR-DTW at synopsis rate of 10^{-1}. The shift error rate of WDTW is 0.165, which is 49%, 26%, and 34% of that of Chan, PDTW, and UCR-DTW at synopsis rate of 10^{-2}. The shift error rate of WDTW+R is 0.101, which is 30%, 15%, and 22% of that of Chan, PDTW, and UCR-DTW at synopsis rate of 10^{-2}.

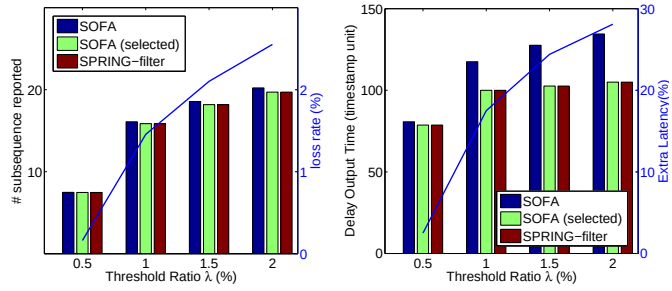
6.1.2 Distance Threshold Ratio λ

In the second experiment, we examine the throughput and accuracy of all methods at different distance threshold ratio λ. The synopsis rate is fixed at $10^{-1.25}$ and the query length is 1000. All values come from averaging of 1000 different queries. The results of throughput and shift error rate on *Posture* are shown in Fig. 11(a) and 11(b), respectively. The throughput of all algorithms remain constant at different



(a) Throughput (b) Accuracy

Fig. 11. The throughput and accuracy of different algorithms at different distance threshold ratio λ



(a) Quality of SOFA (b) Latency of SOFA

Fig. 12. (a) The histograms represent the average number of subsequences reported per query by SOFA, SOFA(selected), and SPRING-filter, respectively. The line is the corresponding loss rate of SPRING-filter compared with SOFA. (b) The histograms represent the average delay time per subsequence, and the line is the corresponding extra latency ratio.

distance threshold ratio. On the contrary, the shift error rates of all algorithms are sensitive to the chosen distance threshold ratio except for WDTW and WDTW+R. WDTW and WDTW+R have significantly lower shift error rates compared to other methods and their shift error rates never exceed 0.03 and 0.01, respectively. The shift error rates of PDTW and Chan are saturated at a certain level when the distance threshold ratio is larger than 0.75%. This is because the non-overlapping constraint becomes a dominate factor when the similarity constraint is relaxed to a certain level by increasing the distance threshold ratio. For UCR-DTW, the shift error rate is always high even if we adjusted the distance threshold ratio. This is due to the constraints of the reported subsequence length and the warping window size.

Concluded from the above three experiments, WDTW processes the online subsequence matching more efficiently compared with WDTW+R, Chan, UCR-DTW, and SPRING-DTW. Moreover, WDTW and WDTW+R outperform other algorithms in accuracy under all parameters and datasets we test. Although PDTW is faster than WDTW, it suffers from a significant high error rate. Thus, both Hypotheses 1 and 2 are accepted.

6.2 Filtering Quality and Report Latency of SOFA

We examine the filtering quality and report latency of SOFA at different distance threshold ratio to test Hypothesis 3. The filtering quality is measured by the number of qualified subsequences. We use the results of SOFA as the ground

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

truth and count the loss of the SPRING-filter since SOFA captures all qualified subsequences (in Theorem 1). The loss rate is measured by $1 - r/s$, where r and s are the numbers of qualified subsequences reported by SPRING-filter and SOFA, respectively.

The numbers of reported subsequences of SOFA and SPRING-filter and the corresponding loss rate of SPRING-filter (in a solid line) at different distance threshold ratio are shown in Fig. 12(a). In general, the numbers of subsequences reported by the two methods increase as the distance threshold ratio increases. Meanwhile, the loss rate of SPRING-filter increases as well. The reason is because the subsequences are more likely to overlap when there are more candidate subsequences passed the similarity constraint, which is the case when we enlarge the distance threshold ratio. In contrast to SOFA that keeps all the replaced subsequences in the stack, SPRING-filter only keeps the one with the smallest distance. When more subsequences are overlapped, SPRING-filter has higher chances to omitted qualified subsequences and thus has a higher loss rate.

We also show the quality of the common subsequences reported by both SOFA and SPRING-filter, which is denoted by SOFA(selected). The result shows the number of subsequence reported by SOFA(selected) is the same as that of SPRING-filter. Since SOFA guarantees no false dismissals for finding the qualified subsequences, all subsequences reported by SPRING-filter are also reported by SOFA. Thus, they report the same number of subsequences.

Next, the results of the report latency of SOFA and SPRING-filter are illustrated in Fig. 12(b). The latency of reporting a subsequence is measured by the difference between the timestamp of the subsequence generated by WDTW and the timestamp of the subsequence reported by SOFA or SPRING-filter. We can see that there exists report latency in both SOFA and SPRING-filter because they need to check whether each subsequence is overlapped by the following ones. SOFA has about $2.5\% \sim 28.0\%$ extra latency, which is plotted in a solid line in Fig. 12(b), compared to SPRING-filter. The extra latency of SOFA increases as the distance threshold ratio increases since more candidate subsequences are buffered in the stack when the distance threshold ratio becomes larger. Please note that, for the common subsequences reported by both SOFA and SPRING-filter, SOFA does not spend more time to discover and report qualified subsequences compared to SPRING-filter. The latency of SOFA(selected) is exactly the same as that of SPRING-filter. In summary, SOFA reports all the qualified subsequences while spending the same time cost to report those also reported by SPRING-filter.

Concluded from this experiment, SOFA reports more subsequences than SPRING-filter and the latency of subsequences that are also reported by SPRING-filter is the same as SPRING-filter. Thus, Hypothesis 3 is accepted.

7 Related Work

Many studies proposed to perform subsequence matching under the DTW distance [18]–[20]. Although DTW has many nice properties, it suffers from high computation cost. To solve this problem, many researchers have been working on various types of DTW algorithms to speed up the matching process. For example, Han et al. [21] applied multi-level lower bound filters to prune unqualified subsequences and solve a ranked subsequence matching problem. Athitsos et al. [6] proposed EBSM, an embedding-based subsequence matching algorithm. EBSM mapped query sequences and database time series into vectors through matching with reference sequences. The subsequences matching problem was then reduced to a low-dimensional vector matching problem. All the above methods leveraged the indexing techniques to speed up the matching process. However, building an index in advance is impractical in stream applications. Hence, developing an online method to compute the warping distance of data streams is necessary.

Thanawin et al. [7] proposed UCR-DTW, an online algorithm to do subsequence matching and searching on trillions of time series. They used different level filters and an early abandon trick to prune the candidate subsequences in a very fast way. However, UCR-DTW only searches those subsequences with the same length as the query length. It is not applicable for applications in which the target subsequence may differ to the query in length.

Another research direction for speeding up sequence matching under the DTW distance is to leverage the data synopsis, especially the histogram-based synopsis [1], which is popularly used in stream applications. PDTW used piecewise aggregate approximation (PAA) structure, which is composed of equal-width segments, to reduce the time complexity of DTW [2]. FastDTW [3] is an approximation of DTW. It used a multilevel approach that recursively projected a warping path from a lower resolution to a higher resolution and refine it. FTW also used the multilevel concept to estimate the DTW distance and achieved an exact similarity search with lower bound distance measurements [4]. However, these studies focus on histogram-based synopsis stream of equal-width bins.

Synopsis structures with arbitrary-width bins are designed for better approximation accuracy such as the adaptive piecewise constraint approximation (APCA) [13] and the Haar wavelet reconstruction [12]. For synopsis structures with arbitrary-width bins, Chan et al. [15] proposed a Haar wavelet-based approximation method under the time warping distance. The algorithm treated each arbitrary-width bin as the same width one and found a sequence of bin matching pairs under DTW. Then the accumulated distance was recomputed by the real width of each bin. However, the results of Chan algorithm have great overestimation of distances and Chan is not designed for online streams.

To the best of our knowledge, we are the first to propose an online subsequence matching algorithm on stream synopses composed of arbitrary-width bins under the DTW distance. We aim to design an efficient method to meet the real time demand in a stream environment while keeping the quality of subsequence matching.

8 Conclusions

We present the online subsequence matching framework *SUMMER* over histogram-based synopses under the DTW distance in a streaming environment. SUMMER is mainly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2725833, IEEE Transactions on Knowledge and Data Engineering

composed of WDTW and SOFA, an efficient subsequence matching algorithm and a stack-based filtering algorithm. Once a bin of an online stream arrives, WDTW computes the accumulated distance of the candidate subsequence using bin-matchings, which are weighted by their time spans, and efficiently reduces the distance overestimation using the bin-cell mergences technique. SOFA filters out the redundant overlapping ones and has been proven that there is no miss for the detection of qualified subsequences. Finally, we obtain subsequences more accurately through the refinement module. The experiment results shows that (1) when compared with Chan, PDTW, UCR-DTW, and SPRING-DTW, WDTW has a low computation cost to meet the time and space constraints of streams, while compromising little accuracy, and (2) when compared with SPRING-filter, SOFA improves significantly in the loss rate.

ACKNOWLEDGEMENTS

This study was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST103-2221-E-002-286-MY2, MOST104-2221-E-002-214-MY3, MOST105-2628-E-001-002-MY2 and MOST103-2221-E-001-006-MY2. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] C. C. Aggarwal and P. S. Yu, "A survey of synopsis construction in data streams," *Advances in Database Systems*, pp. 169–207, 2009.
[2] E. Keogh and M. Pazzani, "Scaling up dynamic time warping for datamining applications," pp. 285–289, 2000.
[3] S. Salvadora and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Journal of Intelligent Data Analysis*, vol. 11, pp. 561–580, 2007.
[4] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "Ftw: Fast similarity search under the time warping distance," *ACM SIGACT-SIGMOD-SIGART*, 2005.
[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *VLDB Endowment*, 2008.
[6] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos, "Approximate embedding-based subsequence matching of time series," *ACM SIGMOD*, pp. 365–378, 2008.
[7] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," *ACM SIGKDD*, 2012.
[8] A. Kotsifakos, I. Karlsson, P. Papapetrou, V. Athitsos, and D. Gunopulos, "Embedding-based subsequence matching with gaps-range-tolerances: a query-by-humming application," *The VLDB Journal*, vol. 24, pp. 519–536, 2015.
[9] D. Lemire, "Faster retrieval with a two-pass dynamic-time-warping lower bound," *Pattern Recognition*, 2009.
[10] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, 2013.
[11] H. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.
[12] C. S. Burrus, R. A. Gopinath, and H. Guo, "Introduction to wavelets and wavelet transforms: A primer," *Prentice Hall*, 1997.
[13] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM SIGMOD*, pp. 151–162, 2001.
[14] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance," *ICDE*, pp. 1046–1055, 2007.
[15] F. K.-P. Chan, A. W. chee Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: with and without time warping," *IEEE Transactions on Knowledge and Data Engineering*, pp. 686–705, 2003.
[16] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "Ucr time series classification/clustering archive," *http://www.cs.ucr.edu/∼eamonn/time_series_data/*.
[17] Y. Cai and R. T. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials," *ACM SIGMOD*, 2004.
[18] H.-K. Lee and J. H. Kim, "An hmm-based threshold model approach for gesture recognition," *IEEE Transation on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 961–973, 1999.
[19] M. Zhou and M. H. Wong, "Efficient online subsequence searching in data streams under dynamic time warping distance," *ICDE*, 2008.
[20] T. S. F. Wong and M. H. Wong, "Efcient subsequence matching for sequences databases under time warping," *IDEAS*, 2003.
[21] W.-S. Han, J. Lee, Y.-S. Moon, and H. Jiang, "Ranked subsequence matching in time-series databases," *VLDB*, pp. 423–434, 2007.

Su-Chen Lin received the Ph.D. degrees in Electrical Engineering from National Taiwan University, Taiwan, in 2016 and received the B.S. degrees in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2006. Her main research area is on data mining and machine learning, with a specific focus on social network analysis and time series mining.

Mi-Yen Yeh is currently an Associate Research Fellow of Institute of Information Science (and Research Center for IT Innovation under joint appointment) at Academia Sinica, Taiwan. She received her B.S. and Ph.D. degrees in Electrical Engineering from National Taiwan University, Taiwan, in 2002 and 2009, respectively. Her main research area is on data mining, with a specific focus on time series mining, social network analysis, and data management on non-volatile memory. She received the best paper award (in system software and security) of the 28th annual ACM Symposium on Applied Computing (SAC 2013), Distinguished Postdoctoral Fellowship in Academia Sinica, and Research Exploration Award in Pan Wen Yuan Foundation. She is a member of IEEE and ACM.

Ming-Syan Chen received the Ph.D. degrees in Computer, Information and Control Engineering from The University of Michigan, Ann Arbor, MI, USA. He is now the Dean of the College of Electrical Engineering and Computer Science and also a Distinguished Professor at National Taiwan University. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, the President/CEO of Institute for Information Industry (III), which is one of the largest organizations for information technology in Taiwan, and also a Distinguished Research Fellow and the Director of Research Center of Information Technology Innovation (CITI) in the Academia Sinica. His research interests include databases, data mining, social networks, and multimedia networking, and he has published more than 350 papers in his research areas. He is a recipient of the National Chair Professorship and Academic Award of the Ministry of Education in Taiwan, the NSC (National Science Council) Distinguished Research Award, Y. Z. Hsu Science Chair Professor Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and K.-T. Li Research Breakthrough Award for his research work, and also the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He received numerous awards for his research, teaching, inventions and patent applications. Dr. Chen is a Fellow of ACM and a Fellow of IEEE.