SpADe: On Shape-based Pattern Detection in Streaming Time Series

Yueguo Chen[§] Mario A. Nascimento[†*] Beng Chin Ooi[§] Anthony K. H. Tung[§]
[§]National University of Singapore [†]University of Alberta
{chenyueg, ooibc, atung}@comp.nus.edu.sg mn@cs.ualberta.ca

Abstract

Monitoring predefined patterns in streaming time series is useful to applications such as trend-related analysis, sensor networks and video surveillance. Most current studies on such monitoring employ Euclidean distance to calculate the similarities between given query patterns and subsequences of streaming time series. Euclidean distance has been shown to be ineffective in measuring distances of time series in which shifting and scaling usually exist. Consequently, warping distances such as dynamic time warping (DTW), longest common subsequence (LCSS), have been proposed to handle warps in temporal dimension. However, they are inadequate in handling shifting and scaling in amplitude dimension. Moreover, they have been designed mainly for full sequence matching, whereas in online monitoring applications, we typically have no knowledge on the positions and lengths of possible matching subsequences. In this paper, we first discuss the weaknesses of existing warping distances on detecting patterns from streaming time series. We then propose a novel warping distance, which we name Spatial Assembling Distance (SpADe), that is able to handle shifting and scaling in both temporal and amplitude dimensions. We further propose an efficient approach for continuous pattern detection using SpADe, that is fundamental for subsequence matching on streaming data. Finally, our experimental results show that SpADe is effective and efficient for continuous pattern detection in streaming time series.

1 Introduction

A good way to understanding changes in streaming time series is to detect interesting patterns from time sequences. A pattern in time series is a set of sequential data items collected in discrete time points, describing a meaningful tendency of evolving data items during a period of time. Patterns (e.g., trajectories of objects, signals from devices) are very important in stream based applications as they imply important phenomenon of the monitored objects. The problem of pattern detection in streaming time series can thus be defined as follows. Given a set of query patterns $\{Q_i\}$, a distance measure on time series, and a distance threshold δ, we want to continuously monitor matching subsequences of streaming time series against any given query pattern in $\{Q_i\}$. A subsequence is said to be matching if the distance between the subsequence and a query pattern is no more than δ. Without loss of generality, we assume that the time interval between two consecutive data items in the sequences is fixed. Otherwise, a transformation into equal time interval sequences can be done by using interpolation.

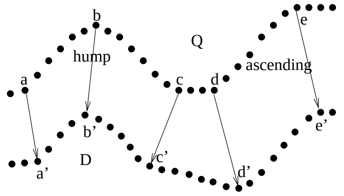


Figure 1. Illustration of shifting and scaling in temporal and amplitude dimensions.

The so-called warps in temporal and amplitude dimensions of time series impose difficulties in measuring distances between time series. Figure 1 shows cases of warps (shifting and scaling) existing between query pattern Q and data sequence D. Note that D is similar to Q at semantic level, as there is a hump followed by an ascending trend in both of them. The first warp is time shifting, i.e., the lag of ascending trend to the hump in Q (measured as $d - c$) is different from that (measured as $d' - c'$) in D. The second is amplitude shifting, e.g., the values of data items between d and e in Q are larger than those of the corresponding items between d' and e' in D. The third is scaling, the extensions of humps in Q and D are different in both temporal dimension (from c-a and c'-a') and amplitude dimension (from $Q[b]$-$Q[a]$ and $D[b']$-$D[a']$).

Euclidean distance is a simple measure for similarity between two time sequences. However, it is sensitive to the warps mentioned earlier and similar patterns can be separated by very large Euclidean distances if the data items are

[*] The work was done while the author was on his sabbatical leave at the Natl. Univ. of Singapore and partially supported by NSERC, Canada

not aligned. To reduce such effect, warping distances such as DTW [2], LCSS [16] and edit distance on real sequence (EDR [4]) have been proposed. Nevertheless, they remain sensitive to shifting and scaling in the amplitude dimension and their effectiveness degenerates quickly in such cases. Figure 2 shows two of such examples. Intuitively, sequence A and B are much more similar in shapes than A and C in the two cases. However, due to amplitude shifting (a), and scaling (b), the warping distances of A and B in the two cases will be much larger than those of A and C.

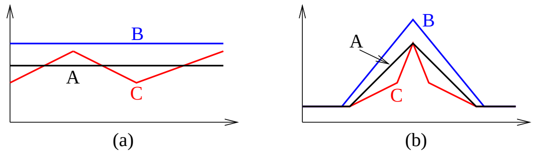


Figure 2. Impact of shifting and scaling in amplitude on existing warping distances.

All the above mentioned distances are used mainly for full sequence matching, in which distance is measured based on the full length of sequences. However, in pattern detection on streaming time series, we have no *priori* knowledge on the positions and lengths of the possible matches. We must first divide the potential subsequences from the streaming time series, and then compare them to query patterns based on full matching. An obvious solution is to compare the most recent subsequences of streaming time series to the query patterns whenever a new data item arrives. However, such an approach is computational intensive, and incurs redundant computational overhead.

Segmentation is a simple way to handle subsequence matching, in which potential matching subsequences are extracted from streaming time series and compared to query patterns. However, potential segments may be hard to extract as many time series patterns have no clear boundaries. Moreover, even though some boundaries can be detected, e.g., by local extreme points in streaming series, there is no effective error bound on distances of the extracted potential matching subsequences, especially when shifting or scaling exists in the streaming time series. Segmentation on pattern detection can introduce many false dismissals. Figure 3 shows an example where false dismissals may occur by using segmentation on streaming time series. When using warping distances such as DTW to measure the distance between Q and the segmented subsequence starting at x', the distance d may be larger than δ due to the unmatched region before y and y'. However, if the segmented subsequence starts after x', e.g, at a point between x' and y', the distance is expected to be less than d. Therefore, it is possible to miss a matching subsequence due to pruning by segmentation. Besides identifying the starting point, the choice of the length of the potential matching subsequences is also difficult with the existence of time shifting and scaling. This is especially true when query patterns are complex.

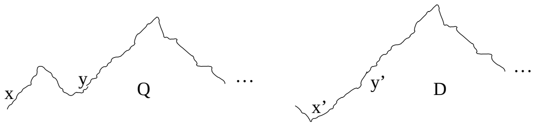


Figure 3. An example of segmentation on subsequence matching. Q and D are well matched from y and y'. However, the starting point of a potential matching subsequence is obtained by segmentation at position x' of D.

As a subsequence matching problem, pattern detection on streaming time series is naturally expensive. Existing warping distances have so far not been extended for online pattern detection in streaming time series while taking both shifting and scaling into account.

In this paper, we introduce a novel definition of warping distance, called Spatial Assembling Distance (SpADe, which is used in our system called PIPA, a database engine for interactive media, for supporting effective matches of shapes extracted from multimedia objects), which supports shifting and scaling in both temporal and amplitude dimensions. SpADe can be used to efficiently perform continuous detection of patterns on streaming time sequences without the need to perform sequence segmentation. Our contributions are as follow:

- We identify the weaknesses of existing warping distances, and explain why they are not suitable for pattern detection on streaming time series in cases where both shifting and scaling along amplitude and temporal dimension exist.

- We propose a novel warping distance SpADe, which can be applied to both full sequence and subsequence matching. SpADe is a robust measure of distances between shape-based time series as it is not sensitive to shifting and scaling in temporal and amplitude dimensions of streaming time series.

- We propose a continuous SpADe calculation approach which can naturally be used on pattern detection in steaming time series. We improve the efficiency of pattern detection by using a pruning approach.

- Experimental study was conducted. Results show that SpADe is both efficient and effective as a distance metric for subsequence matching on streaming time series.

The rest of the paper is organized as follows. Section 2 gives an overview of warping distances and existing solutions on subsequence matching. Section 3 defines SpADe, and Section 4 introduces the approach of continuous pattern detection. Section 5 shows the experimental study of SpADe. Section 6 gives the conclusions.

2 Related Work

Existing works on pattern matching of time series are mainly focused on full sequence matching. Those similarity metrics of full sequence matching can be classified into three categories. The first is Euclidean based metrics in which Euclidean distance is used in measuring distance between either two original time sequences or features got from the original time sequences. It has been observed that Euclidean metric is very sensitive to distortion and noise [4, 9]. It only handles global time scaling by shrinking or stretching time sequences compulsively. As a solution, DTW [2] was employed by finding the optimal alignment between two time sequences. It handles local time shifting and scaling [3], but is still sensitive to amplitude shifting and scaling as the amplitude differences of data items will be accumulated. In addition to DTW, common subsequence based distances such as LCSS [16] and EDR [4] were proposed to further handle noise. However, they are more sensitive to amplitude shifting and scaling than DTW as common subsequences are found by comparing amplitude values. Our proposed distance SpADe tries to handle both shifting and scaling in temporal and amplitude dimensions of time series. We compare SpADe to existing distances on time series in Table 1.

	Time shifting	**Time scaling**	**Amplitude shifting**	**Amplitude scaling**	**Noise**
Euclidean		partially			
DTW	✓	✓			
CommonSub	✓	✓			✓
SpADe	✓	✓	✓	✓	✓

Table 1. The abilities of distance functions.

To our knowledge, little study on subsequence matching has been conducted. ST-index [7], Dual Match [12] and General Match [11] use fixed size sliding windows on sequences. They map each window of data items into a multidimensional point and use indexing techniques to efficiently match the subsequences in feature space. The difference between them is whether a sliding window or disjoint window is used in query sequences and streaming sequence. The limitation of these studies is the use of Euclidean distance on measuring similarities in feature space. Park et al. [14] proposed an approach for subsequence matching by applying DTW. The suffix tree is used to index possible subsequences of the data sequences. However, all these studies on subsequence matching try to search the matches of short query patterns to long sequences in a database, where index can be built on long data sequences.

Pattern detection on streaming time series is different from finding matching subsequences in database. It tries to detect matching subsequences within long streaming sequences to any given query pattern. Wu et. al [19] proposed an online segmentation and pruning algorithm to simplify the data sequence as zigzag shapes. They measured similarities of subsequences based on the permutation of the end points of piecewise segments. However, the piecewise linear representation limits its application in shape based pattern matching on time series. Wei et. al [18] used a wedge to merge multiple candidate sequences, and compared the set of wedges against the subsequences in the coming data stream. The limitation of their work is that it does not consider how to partition the steaming time series and how to efficiently find matching subsequences.

Euclidean distance or its variation (e.g., correlations) was used in matching patterns in some recent works on streaming time series such as BRAID [15], SPIRIT [13]. Gao et. al [8] also studied continuous pattern queries on streaming time series. They attempted to detect the nearest neighbor pattern when new data value arrives. Fast Fourier Transform was used to efficiently find the cross correlations of time series, which yields a batch processing model. As mentioned earlier, the use of simple Euclidean distance or correlation in these studies affects the effectiveness of pattern matching where shifting and scaling exist.

3 Spatial Assembling Distance

We have a hypothesis that in human intuition, the match between two shape-based sequences always comes from multiple sequential matches of a set of subsequences. Based on this hypothesis, we propose a novel warping distance, Spatial Assembling Distance (SpADe), which can be applied to measure distances between shape-based time series under both full and partial sequence matching. We first introduce SpADe for full sequence matching.

3.1. Local Pattern Match

In full sequence matching, the distance between two time sequences $Q[0:m]$ and $D[0:n]$ are measured based on the full length of two sequences. We borrow the idea from General Match [11], and extract a set of small patterns from time series by using a fixed size of sliding window. These small patterns of the same length are called local patterns. By using the fixed size sliding window on two time sequences, we get two sets of local patterns.

A local pattern lp from a time sequence Q can be described as $lp = (\theta_{pos}, \theta_{amp}, \theta_{shp}, \theta_{tscl}, \theta_{ascl})$, which are the position of lp in Q, the mean amplitude of data items in lp, the shape signature of lp, and the temporal and amplitude scales (equal to 1 if Q is not scaled) of lp respectively. They should be easily computed from Q. The distance of two local patterns lp in Q and lp' in D, can be measured as

$D_1(lp', lp) = f(|\theta'_{amp} - \theta_{amp}|, |\theta'_{shp} - \theta_{shp}|)$, which is a weighted sum of the differences in amplitude and shape features of two local patterns. The weights in f is application-specific, depending on the tolerance of the amplitude difference and that of the shape difference.

A local pattern match (LPM) p is formed from lp and lp' if $D_1(lp', lp) < \varepsilon$, which means that there is a match between local pattern lp and lp'. We label the positions of lp in Q and lp' in D as $p.x$ and $p.y$ respectively. A matching matrix of $m \times n$ is shown in Figure 4 to describe the match of local patterns in Q and D. The relative positions of lp and lp' are obtained by projecting p horizontally and vertically. p can be described by the following features: $p = (p.x, p.y, \psi_1, \psi_2, \psi_3, \psi_4) = (\theta_{pos}, \theta'_{pos}, \theta'_{pos} - \theta_{pos}, \theta'_{amp} - \theta_{amp}, \theta'_{tscl} - \theta_{tscl}, \theta'_{ascl} - \theta_{ascl})$, where ψ_1, ψ_2, ψ_3 and ψ_4 represent the shifting and scaling of p in temporal and amplitude dimensions respectively.

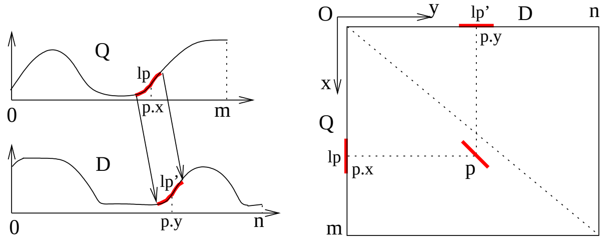


Figure 4. An example of a LPM and its corresponding local patterns in matching matrix.

Note that there may be a number of local patterns extracted from time sequences Q and D. A large number of LPMs will be formed if Q and D are similar in shapes. Their distribution can be visualized in the matching matrix formed from the two sequences.

3.2. Definition of SpADe

We measure the SpADe of two time sequences based on the distribution of LPMs in the matching matrix. For full sequence matching, we try to find the best combination of LPMs, such that they can maximize the matches of Q and D. The quality of pattern combination is determined by the following two criteria: 1), the projections (vertical and horizontal) of LPMs should cover large regions of Q and D. The larger the covered regions, the more data items in Q and D are matched; 2), the difference of features of two LPMs should be as small as possible, which means that two LPMs can be obtained by a similar transformation from local patterns in Q to local patterns in D.

3.2.1 Distance between two LPMs

We define the gap between two LPMs p_1 and p_2 on Q as $ED(p_2, p_1) = max(p_2.x - p_1.x - w, 0)$ if $p_2.x > p_1.x$; else $ED(p_2, p_1) = +\infty$. w is the length of the local patterns. Similarly, the gap between p_2 to p_1 on D is measured as $ED'(p_2, p_1) = max(p_2.y - p_1.y - w, 0)$ if $p_2.y > p_1.y$; else $ED'(p_2, p_1) = +\infty$. The gaps are used to handle the noise and local unmatched regions within time series.

Definition 1. *The distance of p_2 to p_1 is $D_2(p_2, p_1) = g(ED(p_2, p_1)) + g(ED'(p_2, p_1)) + h(|p_2.\psi - p_1.\psi|)$.*

Function $g(x)$ is a penalty on the gaps between two LPMs, which can be defined by users, but should satisfy the following properties: 1) $g(0) = 0$; 2) be monotonic. Function $h(x)$ is a weighted penalty of difference of features: time shifting, amplitude shifting, time scale and amplitude scale. Therefore, $h(|p_2.\psi - p_1.\psi|) = w_1 \cdot |p_2.\psi_1 - p_1.\psi_1| + w_2 \cdot |p_2.\psi_2 - p_1.\psi_2| + w_3 \cdot |p_2.\psi_3 - p_1.\psi_3| + w_4 \cdot |p_2.\psi_4 - p_1.\psi_4|$. In our study, $g(x) \geq w_1 \cdot x$ should be satisfied, which provides an elegant pruning approach shown later.

We also define the distance between a LPM p and a point at the top or bottom of the matching matrix according to the gaps between p and the point. For a point $P_s(0, y_s)$ satisfying $y_s \leq p.y - \frac{w}{2}$, the distance from P_s to p is $D_3(p, P_s) = g(p.x - \frac{w}{2}) + g(p.y - y_s - \frac{w}{2})$. Similarly, for a point $P_e(m, y_e)$ satisfying $y_e \geq p.y + \frac{w}{2}$, the distance from p to P_e is $D_4(P_e, p) = g(m - p.x - \frac{w}{2}) + g(y_e - p.y - \frac{w}{2})$.

3.2.2 SpADe in full sequence matching

Definition 2. *Given a path $r = P_s \rightarrow p_1 \rightarrow \ldots \rightarrow p_t \rightarrow P_e$ formed by $P_s(0, y_s)$, $P_e(m, y_e)$, and a number of LPMs $p_1, \ldots, p_t$, the length of r is defined as $Cost(r) = D_3(p_1, P_s) + \sum_{i=1}^{t-1} D_2(p_{i+1}, p_i) + D_4(P_e, p_t)$.*

Given two sequences $Q[0:m]$ and $D[0:n]$, a matching matrix can be built based on all the LPMs between Q and D. Given two corner points $P_s(0, 0)$ and $P_e(m, n)$ in the matching matrix, $\{r_i\}$ include all the paths derived from the LPMs, and linking P_s and P_e. The SpADe of Q to D under full sequence matching is defined as:

Definition 3. $SD(D, Q) = \min_t Cost(r_t), r_t \in \{r_i\}$.

In other words, the SpADe of two given time sequences is the length of shortest path from left-up corner to the right-bottom corner in the matching matrix of these two sequences. It is determined by the spatial distribution of the derived LPMs. We find the best combination of LPMs using the shortest path connecting two end points. That is why we call the distance as spatial assembling distance. Finding shortest paths has been well studied and the classic Dijkstra's algorithm [6] can be applied.

4 SpADe on Subsequence Matching

SpADe is useful not only for full sequence matching, but for subsequence matching as well. In this section, we show

how SpADe can be continuously calculated in subsequence matching. We achieve the efficiency of continuous SpADe calculation in two ways: efficient LPM detection and continuous shortest path calculation.

4.1. Efficient detection of LPMs

Detecting LPMs in streaming time series against a set of query patterns is the foundation of SpADe calculation. To improve the accuracy and efficiency of pattern matching, some preprocessing such as sampling and normalization can be conducted on time sequences.

Short local patterns are preferred to describe the fine grained shapes of time series. This is because long local patterns generate more false positive LPMs, as large ε is needed when patterns are long to guarantee no false dismissal of LPMs. Haar wavelet [5] is a good candidate for extracting θ_{amp} and θ_{shp} features from local patterns, as low band wavelet coefficients elegantly describe the mean amplitude and the general shape of local patterns. Moreover, it is computationally efficient.

Sliding window of width w is used to extract local patterns at every sliding step of the time series. To handle the possible scale difference between time series, query patterns are transformed into various scales in temporal and amplitude dimensions. Local patterns are extracted from those scaled query patterns. Considering θ_{amp} and θ_{shp} features are in low dimensions (e.g., 4), an R-tree can be used to index these local patterns so that the LPMs of lp can be efficiently detected.

To handle the variations in shifting and scaling, given a local pattern lp extracted from the streaming series, a large number of local patterns from queries will match lp. Therefore, many branches in R-tree are involved during the query, which incur much computational overhead. Inspired by VA-File [17], we partition the feature space into cells, and approximate the distance between local patterns according to the cells they fall in. As the number of dimensions is small and adequate variation should be allowed, the total number of cells will be much less than the number of local patterns. Therefore, efficiency on detecting LPMs can be achieved.

4.2. Continuous SpADe calculation

We have mentioned that there is a challenge on efficient measure of distances between subsequences in streaming data and queries. Our proposed SpADe is a good candidate to continuously monitor matching subsequences. To illustrate this, we first give some definitions.

4.2.1 Variance of SpADe in subsequence matching

Given a query $Q[0:m]$ and a window size of most recent data items $D[t_s:t_e]$ in the streaming time series, the local SpADe of D at time point t ($t_s \leq t < t_e$) is defined as:

Definition 4. $SD_l(D_t, Q) = \min_{i<t_e} SD(D[t:i], Q)$.

$SD_l(D_t, Q)$ measures the distance of the best matching subsequence (to Q) starting at time point t of D. As shown in Figure 5, $SD_l(D_t, Q)$ can be explained as the shortest path from point $P_s(0, t)$ to points $P_e(m, t')$ ($t < t' < t_e$). Let $t_r = argmin_{t'} SD(D[t:t'], Q)$. $SD_l(D_t, Q)$ is actually the full sequence matching SpADe of $D[t:t_r]$ to Q.

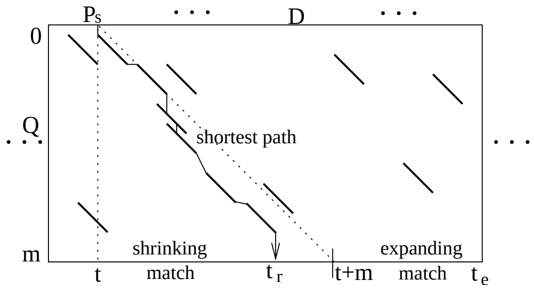


Figure 5. An example of local SpADe.

Pattern detection problem tries to find subsequences of D whose SpADe to query Q is less than some threshold δ. This can be achieved by continuously calculating local SpADe, i.e., finding matching subsequences satisfying $SD_l(D_t, Q) \leq \delta$ at every point of D. However, it is not efficient enough because each calculation of SpADe requires finding the shortest path of LPMs within some window size, which consumes much computation. To improve the efficiency of continuous SpADe calculation, we propose an incremental way of calculating SpADe.

In pattern detection applications, the probability of having matching subsequence grows as the number of LPMs increases. Much computation will be saved if SpADe distance is updated only when new LPMs are detected.

Definition 5. *The cumulating SpADe of a detected LPM p to query Q, noted as $SD_c(p)$, is the shortest path starting from points at the top edge of matching matrix to p.*

Definition 6. *The potential SpADe of a LPM p to query Q is defined as $SD_p(p) = SD_c(p) + g(m - p.x - \frac{w}{2})$.*

$SD_c(p)$ is a lower bound on the length of paths passing through p and linking the top and bottom edges of the matching matrix. Once $SD_c(p) > \delta$, p will not emerge in the path of any qualified matching subsequence for Q. On the other hand, if $SD_c(p) \leq \delta$, p is a promising LPM. Meanwhile, $SD_p(p)$ is an upper bound of local SpADe. Therefore, Once $SD_p(p) \leq \delta$, a qualified matching subsequence to the given region query on Q is found.

4.2.2 Incremental calculation of SpADe

On pattern detection in streaming time series, we actually detect LPMs by cutting the most recent local pattern from streaming data sequence, extracting feature from the chopped local pattern, and retrieving LPMs of the local pattern. On detecting a LPM p, it will be perfect if $SD_c(p)$ and $SD_p(p)$ can be calculated on the fly. The following lemma supports this incremental way of SpADe calculation.

Lemma 1. *The LPMs detected behind a LPM p on streaming time series will not change $SD_c(p)$.*

Proof: *Suppose p_1 is detected behind p. Therefore, $p_1.y \geq p.y$. If p_1 change $SD_c(p)$, it should be in the shortest path of $SD_c(p)$. Let $p_1 \rightarrow ... \rightarrow p_t \rightarrow p$ is a path from p_1 to p in shortest path. Then we must be able to find two consecutive LPMs p_{t_1} and p_{t_2} in the path, such that, p_{t_1} is detected behind p_{t_2}, i.e., $p_{t_1}.y \geq p_{t_2}.y$, and $ED(p_{t_2}, p_{t_1}) = +\infty$. According to Definition 1, $D_2(p_{t_2}, p_{t_1}) = +\infty$. Therefore, $SD_c(p) = +\infty$, which is impossible because we can at least find a path from $P_s(0, p.y - \frac{w}{2})$ to p whose cost is only $g(p.x - \frac{w}{2})$. Consequently, p_1 cannot change the value of $SD_c(p)$.* □

Lemma 1 guarantees that $SD_c(p)$ can be immediately calculated when p is detected from the streaming time series. The calculation of $SD_c(p)$ is to find the previous LPM of p, noted as p', from which the shortest path from top edge of the matching matrix to p is found, i.e., $p' = argmin_{p_1}(SD_c(p_1) + D_2(p, p_1))$. According to Definition 1, p' should be in the top-left corner of p. Figure 6 shows the searching region $ABOC$ of p'. This is because for those LPMs whose reference point is beyond $ABOC$, one of the gaps of p to them will be $+\infty$.

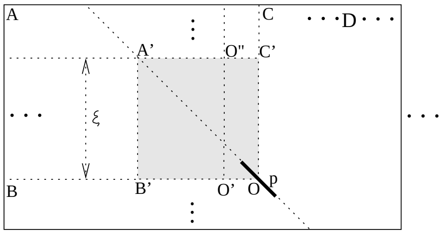


Figure 6. Searching region of previous LPM.

However, it is not necessary to search p' in the large region of $ABOC$, as large gaps are usually not allowed in practice. Therefore, the searching region of p' can be reduced by constraining the gaps between two consecutive LPMs. Figure 6 shows the constraint searching region $A'B'OC'$ with gap bound of ξ. The efficiency of calculating $SD_c(p)$ will be improved significantly, as small ξ is used in practice. The cumulating SpADe obtained from the constraint searching region is noted as $SD_{c,\xi}(p)$. On detecting p', we get $SD_{c,\xi}(p) = SD_{c,\xi}(p') + D(p, p')$. For range query, if $SD_{c,\xi}(p) > \delta$, we simply drop p as it will not appear as a LPM in a qualified matching subsequence.

To find p' of p, we need maintain those LPMs in the searching region of p', and test all the LPMs within this region row by row. To reduce the number of detected LPMs, we actually use a sliding step which equals to the length of local patterns. That means, we chop the streaming time series into disjoint local patterns, and detect the LPMs for every chopped local pattern. This technique is also used in ST-index [7]. It helps to reduce the number of LPMs could appear in the searching region of p'. As shown in Figure 6, for each row in $A'B'O'O$", the maximum of LPMs is $\lfloor \frac{\xi}{w} \rfloor$. Therefore, the memory cost of continuous SpADe calculation will be bounded as the maximal number of LPMs need to maintained, $O(\frac{N\bar{m}\xi}{w})$, where N is the number of query patterns and $\bar{m}$ is the average length of query patterns. The complexity of whole pattern detection will be $O(\frac{tNn\xi^2}{w})$, where t is the average number of $LPMs$ detected from one chopped local pattern of streaming time series. In the regions where no matching subsequences appear, the number of LPMs will be very small, close to zero. Therefore, the calculation of $SD_{c,\xi}(p)$ will be very efficient.

Along with the calculation of $SD_{c,\xi}(p)$, we record the starting point of the shortest path to p. $SD_{p,\xi}(p)$ is calculated following the calculation of $SD_{c,\xi}(p)$. As we have declared, once $SD_{p,\xi}(p)$ is found to be less than δ, a qualified matching subsequence is detected. The position of matching subsequence is actually the vertical projections from the starting point of the shortest path of p to the end point of p. Considering that the potential SpADe of some LPMs around p may also satisfy the range query, the LPM who has the smallest $SD_{p,\xi}(p)$ within a local region is returned as the matching subsequence in this region.

4.2.3 Pruning approach in SpADe calculation

The major computation cost of range query on streaming time series comes from the calculation of cumulating SpADe of detected LPMs. Query processing will be efficient if some LPMs can be pruned without the calculation of cumulating SpADe. In the following, we introduce the concepts of post-bound and estimate-bound, and show that how such a pruning approach is achieved.

Definition 7. *The post-bound of a LPM p is the lowest position of the potential posteriors of p, which can be located in the next column of p.*

A LPM p_2 is a potential posterior of p_1 if $SD_{c,\xi}(p_1) + D_2(p_2, p_1) \leq \delta$. Suppose the post-bound of p_1 is b_1, according to the definition, for any p_3 satisfying $p_3.y = p_1.y + w$ and $p_3.x > b_1$, p_3 will not be a potential posterior of p_1. Based on this, we have the following lemma.

Lemma 2. *For any LPM p_4 satisfying that $p_4.y \geq p_1.y + w$ and $p_4.x > b_1$ which is the post-bound of p_1, p_4 will not be a potential posterior of p_1.*

Proof: *We simplify $p_i.x$ and $p_i.y$ as x_i and y_i. For a p_4 satisfying the conditions in Lemma 2, a virtual LPM p_3 can be found such that $y_3 = y_1 + w$, $x_3 = x_4 > b_1$, and*

$p_3.\psi_{2-4} = p_4.\psi_{2-4}$. *Therefore, p_3 is not a potential posterior of p_1. To show that p_4 is also not a potential posterior of p_1, we only need prove that $D_2(p_4, p_1) \geq D_2(p_3, p_1)$. The relationship of p_1, p_3 and p_4 is shown in Figure 7.*

$$\begin{aligned}
D_2(p_3, p_1) &= g(x_3 - x_1 - w) + w_1(x_3 - x_1 - w) + \Delta\psi \\
D_2(p_4, p_1) &= g(x_4 - x_1 - w) + g(y_4 - y_1 - w) \\
&\quad + w_1|(x_4 - x_1) - (y_4 - y_1)| + \Delta\psi \\
\Delta\psi &= \sum_{i=2}^{4} w_i |p_4.\psi_i - p_1.\psi_i| \\
\therefore \Delta D &= D_2(p_4, p_1) - D_2(p_3, p_1) \\
&= g(y_4 - y_1 - w) + w_1|(x_4 - x_1) - \\
&\quad (y_4 - y_1)| - w_1(x_3 - x_1 - w) \\
&\geq w_1(y_4 - y_1 - w) - w_1(x_4 - x_1 - w) \\
&\quad + w_1|(x_4 - x_1) - (y_4 - y_1)| \\
&\geq w_1|(x_4 - x_1) - (y_4 - y_1)| \\
&\quad - w_1((x_4 - x_1) - (y_4 - y_1)) \\
&\geq 0
\end{aligned}$$

$\therefore$ *$D_2(p_4, p_1) \geq D_2(p_3, p_1)$, and p_4 is not a a potential posterior of p_1.* □

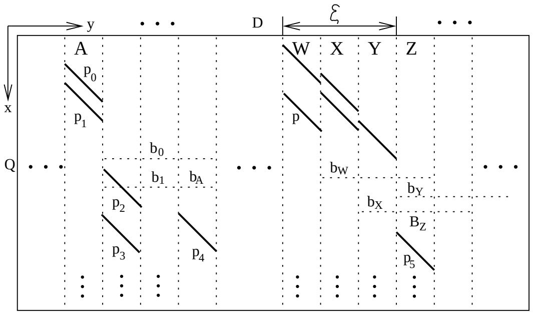


Figure 7. Pruning in SpADe calculation.

We have mentioned that disjoint sliding window is used to chop local patterns from streaming time series. Therefore, a column of LPMs will be got for every chopped local pattern. The post-bound of a LPM p_i in column A is b_i. The post-bound of column A can be defined as $b_A = \max_{p_i.y = A.y} b_i$, i.e., the lowest post-bound of p_i in column A. According to Lemma 2, any LPM below b_A and behind column A will not be a potential posterior of any LPM in column A.

Definition 8. *The estimate-bound of a column A can be defined as $B_A = \max_{A.y-\xi \leq X.y < A.y} b_X$, where X is a column before A.*

Figure 7 shows an example of estimate-bound B_Z of column Z. It is obvious that for a LPM p_5 below B_Z in column Z, it is not a potential posterior of any LPM in column W, X, Y. In other words, the previous LPM of p_5 will not be found in the searching region of p'_5. Therefore, p_5 can be pruned without the calculation of $SD_{c,\xi}(p_5)$.

The estimate-bound of a column can be continuously calculated based on the post-bound of previous columns. On getting a promising LPM in a new column, we update the post-bound of that column, so that it can be used to calculate the estimate-bound of following columns.

5 Performance Evaluation

5.1. Datasets

In our performance evaluation, we compare SpADe with Euclidean distance, DTW and EDR in terms of accuracy and efficiency. Our test platform is a PC with Pentium4 3.0G CPU and 1G RAM. To test the accuracy of SpADe for full sequence matching, we use UCR Time Series Classification/Clustering datasets [1]. Streaming time series datasets are needed to measure the performance of SpADe on continuous pattern detection. Unfortunately, we cannot find such a labelled streaming time series dataset. We observe that the sequential tandem of samples in the MotorCurrent(MC) dataset [1] is just a smooth streaming time series. The MC dataset contains 21 classes of signals with 20 samples of each signal class for a total of 420 signals. We do a fixed step sampling over the MC dataset, and achieve a length of 300 for each sample sequence. Then, a synthetic labelled streaming time series is generated by simply connecting these samples. To ensure that there are shifting and scaling along both time and amplitude dimensions of the streaming MC dataset, we randomly perform some shifting and scaling in random positions of the streaming dataset. On average, one shifting (or scaling) is introduced for half the length of each query pattern. We test the performance of distance metrics under various degree of shifting and scaling along temporal and amplitude dimensions in the synthetic streaming time series.

5.2. Setting of parameters

Abstract functions ($f(x)$ in $D_1(x)$, and $g(x)$, $h(x)$ in Definition 1) are used in SpADe to detect LPMs and penalize differences between LPMs. Users can specify these functions according to the variation factors they want to handle. To simplify the calculation, L_1 norm function is used in $f(x)$. The parameter ε determines the number of LPMs detected during query processing. Large ε generates more false LPMs and reduces the efficiency. While small ε may cause the dismissal of some positive LPMs. The choice of ε can be based on the tolerance of the differences between local patterns of LPMs learned from training data.

We set $g(x) = x$ as which satisfies the requirements on $g(x)$. One parameter related to $g(x)$ is the largest allowed gap ξ, which is dependant on the applications. In MC dataset, we set $\xi = 5w$, which should be large enough in most applications. Coefficients in $h(x)$ are set based on the percentage of penalty users want to put on various features of LPMs. They can be approximately set as long as they are comparable to $g(x)$. Note that all the above parameters on SpADe are not optimized. We simply use reasonable values, and we will show that such a way of setting parameters is already sufficient for SpADe to have better performance than the other distance metrics in most cases.

5.3. Accuracy of SpADe

5.3.1 Accuracy in full sequence matching

Like in many other studies [10, 4], one nearest neighbor classification (1NN) is used to test the accuracy of distances under full sequence matching. In 1NN classification, for each sequence in the testing dataset, we predict its label from its nearest neighbor in the training dataset. If the derived label is the same as the original label of the testing sequence, we get a hit; Otherwise, we get a miss. As shown in Table 2, error rates of three distance metrics are measured on all datasets. We compare the accuracy of the distance metrics based on these error rates. For each distance metric, we allow the adjustment of important parameters (e.g., warping width of DTW, matching threshold of EDR, pattern length of SpADe), and the best classification error rates are recorded. From Table 2, we see that in many datasets with smooth shapes, SpADe achieves lower error rates than the other distance metrics.

Dataset	Euclidean	DTW	EDR	SpADe	Performance
Syn. con.	0.12	0.017	0.04	0.08	normal
Gun point	0.087	0.087	0.02	0.007	good
CBF	0.148	0.004	0.0111	0.02	normal
FaceAll	0.286	0.192	0.194	0.214	normal
OSULeaf	0.483	0.384	0.215	0.132	good
Swed. leaf	0.213	0.157	0.096	0.125	normal
50words	0.369	0.242	0.198	0.215	normal
Trace	0.24	0.01	0.04	0	good
Two Pat.	0.09	0.002	0.002	0.005	normal
Wafer	0.005	0.005	0.007	0.012	bad
FaceFour	0.216	0.114	0.034	0.034	good
Lighting2	0.246	0.131	0.148	0.278	bad
Lighting7	0.425	0.288	0.301	0.315	normal
ECG200	0.12	0.12	0.1	0.13	bad
Adiac	0.389	0.391	0.384	0.319	good
Yoga	0.17	0.155	0.194	0.123	good
FISH	0.217	0.16	0.08	0.017	good

Table 2. Error rates of 1NN classification in full sequence matching.

The length of local patterns w is an important parameter affecting the efficacy of SpADe. It should be chosen according to the sampling rates of time series. Figure 8 shows two examples of the impact of pattern length on the accuracy of 1NN classification in full sequence matching. Longer local patterns require larger ε, therefore, generate more false LPMs. On the other hand, shorter patterns are more ambiguous which will also generate more false LPMs when amplitude shifting and scaling need to be handled. Moreover, the computation cost of SpADe will be increased when the pattern is short.

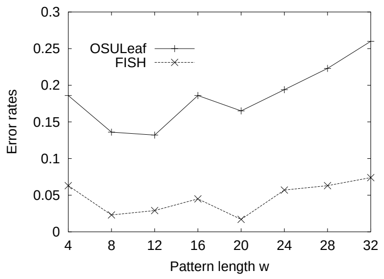


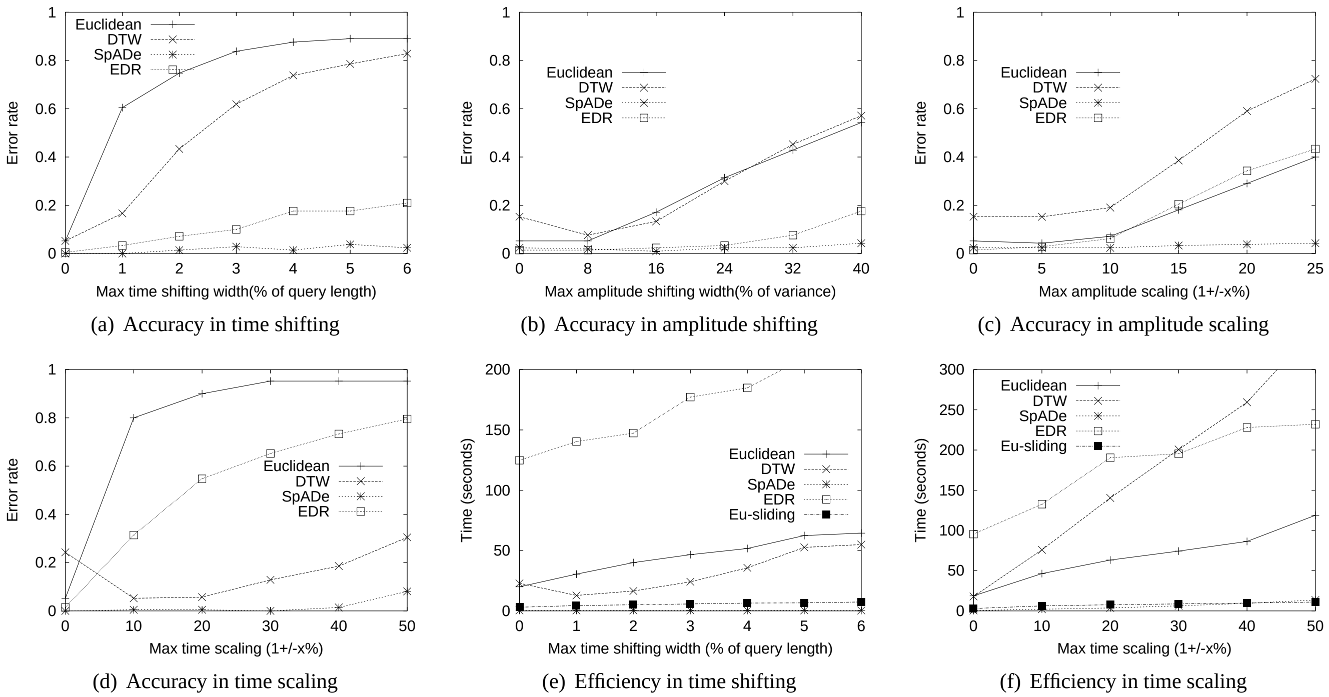
Figure 8. Impact of pattern length on accuracy in 1NN classification.

5.3.2 Accuracy in subsequence matching

We next test the accuracy of SpADe on streaming time series, which is based on subsequence matching. For each class of signals from the MC dataset, we generate a representative pattern as a query pattern. Therefore, there are a total of 21 query patterns. To test the accuracy of distance measures on pattern detection, we use KNN query to retrieve the 10 nearest neighbors of each query pattern (each class has 10-11 repetitive patterns in the dataset). Hits or misses of detected matching subsequences are determined by the positions of subsequences. Distances of subsequences are continuously measured and a small window size is used to retrieve the best matching subsequence in a local region. The accuracy is evaluated based on the average error rate of the KNN queries.

In the first test, we evaluate the accuracy of distance measures under various degree of time shifting. We introduce time shifting by moving some random partitions of time series forward or backward by a random step, so that items between two consecutive partitions is no more aligned. The warping window size of DTW and EDR is set to be the max shifting width in each test. As shown in Figure 9(a), the error rates of all distances are very low on the original time series (shifting width is 0). However, the error rates of DTW and Euclidean distance increase quickly when time shifting is enlarged. It is obvious that SpADe is much more accurate than the others.

In Figure 9(b) and 9(c), we compare the accuracy of the four distance metrics under various degree of amplitude shifting and amplitude scaling. No artificial time shifting is involved in these tests. To handle the intrinsic time shifting, the warping width of DTW and EDR is set as 2% of the query pattern length. The results clearly show that SpADe outperforms the other three in handling amplitude shifting and scaling. EDR performs better than DTW and Euclidean when the amplitude shifting and scaling are small because the matching threshold in EDR partially handle a small degree of amplitude variation.



(a) Accuracy in time shifting (b) Accuracy in amplitude shifting (c) Accuracy in amplitude scaling

(d) Accuracy in time scaling (e) Efficiency in time shifting (f) Efficiency in time scaling

Figure 9. Performance comparisons under various factors.

Time scaling is the most difficult factor for distance metrics since various time scales need to be tested to guarantee no false dismissals. Note that DTW and EDR naturally handle some time scaling by the best warping paths in distance matrix. However, we do not enlarge the scope of warping paths of EDR to handle time scaling as it incurs huge computational cost due to the lack of pruning techniques such as Keogh lower bound [9] used in DTW. We compare the distance metrics with various time scales. As shown in Figure 9(d), SpADe is still the best. In this case, the Euclidean distance is not comparable to the others due to its inability to handle scaling without forced stretching and shrinking. The straight forward way to stretch or shrink of query patterns to various scales of subsequence incurs huge computation, and it is limited to handling global scaling.

5.4. Efficiency of SpADe

In our efficiency test, the Keogh lower bound [9] is used to accelerate the calculation of DTW. To ensure accuracy, we can measure the distances at every position when a new data item arrives. However, it is not efficient. We find that using a proper sliding step (i.e., computing distances for every batch of data items) only affects the accuracy slightly, but achieves efficiency significantly. For DTW and EDR, we choose the sliding step as the same width of warping window. This is because the warping window guarantees no dismissal of warping paths of matching subsequences. For SpADe, the best accuracy is achieved when sliding step equals to the length of local patterns. This is because small sliding step generates more LPMs and incurs some pathological paths. For Euclidean distance, we use two sliding steps to test the efficiency. One (Euclidean) is of length 1, which is used in testing accuracy, and the other (Eu-sliding) is sliding step with the same length as SpADe.

Figure 9(e) shows the efficiency of distance metrics in various scales of time shifting, corresponding to the accuracy measurements in Figure 9(a). We see that SpADe is much better than the others. EDR is expensive due to the lack of pruning supports. Figure 9(f) shows the efficiency of distance metrics in various degree of time scaling. The efficiency of SpADe is similar to that of Eu-sliding, while much better than that of the others. We also find that the efficiency of DTW deteriorates drastically due to the enlargement of warping region in distance matrix to handle large time scaling. The corresponding accuracy of these distances (except Eu-sliding) in this test group is shown in Figure 9(d). We do not compare the efficiency of distance measures under amplitude shifting and scaling, as DTW and Euclidean distance did nothing to cater to them, while SpADe pay some additional cost on handling these factors.

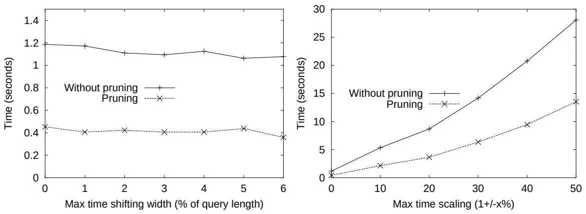
Note that the above tests treat the variation factors independently. We also conduct a comparison of accuracy and efficiency of distance measures in a case where 2% of max time shifting, 10% of max amplitude shifting and scaling, and 20% of max time scaling are introduced. The results are shown in Table 3. We see that SpADe is much more accurate than the other distances. The efficiency of SpADe is only worse than Eu-sliding slightly, while much better than the other three.

	Euclidean	DTW	Eu-sliding	EDR	SpADe
Error rates	0.867	0.376	0.871	0.567	0.019
Time cost	64.767	141.440	8.047	217.788	16.609

Table 3. Performance comparison on a case of compositive variations.

5.5. Pruning effect of SpADe

The pruning approach of SpADe reduces the number of cumulating SpADe need to be calculated in query processing. We test the effect of the proposed pruning approach under various scales of time shifting and scaling. As shown in Figure 10, 2-3 times of efficiency is achieved by the pruning approach in these tests.



(a) Time shifting

(b) Time scaling

Figure 10. Pruning effect of SpADe.

6 Conclusion

We argue that the existing warping distances do not work well in detecting shape based patterns. Our experiments show that Euclidean distance, DTW and EDR have poor accuracy on pattern detection in streaming time series, when shifting and scaling exist in temporal or amplitude dimensions.

We propose a novel distance, SpADe, which can be used to measure distance between shape based time series. The measure of SpADe is based on detection of the best combination of LPMs by calculating the shortest path in matching matrix. It is a good mapping of high level human intuition to low level distance. Therefore, good accuracy is achieved by SpADe in the conditions of shifting and scaling in both temporal and amplitude dimensions. We apply SpADe on pattern detection in streaming time series.

To speed up the calculation of SpADe, we propose to use wavelets to retrieve the important shape coefficients of local patterns. We use the R-tree/cells to index these multidimensional local patterns. To further speed up the continuous query processing on SpADe, we propose an incremental calculation of SpADe, which is very suitable for pattern detection on streaming time series. We also propose a pruning approach to limit the searching region of previous LPM and prune the calculation of cumulating SpADe. Extensive performance study was conducted and the results show that SpADe is both efficient and effective as a distance metric for subsequence matching on streaming time series.

References

[1] *UCR Time Series Data Mining Archive*. http://www.cs.ucr.edu/ eamonn/time_series_data/.

[2] D. J. Bemdt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 229–248, 1994.

[3] A. W. chee Fu, E. Keogh, L. Y. H. Lau, and C. A. Ratanamahatana. Scaling and time warping in time series querying. In *VLDB*, pages 649–660, 2005.

[4] L. Chen, M. T. Ozsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[5] C. K. Chui. *An Introduction to Wavelets*. Academic Press, San Diego, 1992.

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271, 1959.

[7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[8] L. Gao and X. S. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *SIGMOD*, pages 370–381, 2002.

[9] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[10] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *SIGKDD*, pages 102–111, 2002.

[11] 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 *SIGMOD*, pages 382–393, 2002.

[12] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh. Duality-based subsequence matching in time-series databases. In *ICDE*, pages 263–272, 2001.

[13] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[14] S. Park, W. W. Chu, J. Yoon, and C. Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *ICDE*, pages 23–32, 2000.

[15] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[16] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[17] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[18] L. Wei, E. Keogh, H. Van, and H. A. Mafra-Neto. Atomic wedgie: Efficient query filtering for streaming time series. In *ICDM*, pages 490–497, 2005.

[19] H. Wu, B. Salzberg, and D. Zhang. Online event-driven subsequence matching over finacial data stream. In *SIGMOD*, pages 23–34, 2004.