

Segment-Based Approach for Subsequence Searches in Sequence Databases

Sanghyun Park
 Department of Computer Science
 University of California, Los Angeles
 shpark@cs.ucla.edu

Sang-Wook Kim
 Software Tools and Techniques Team
 IBM T.J. Watson Research Center
 swkim@us.ibm.com

Wesley W. Chu
 Department of Computer Science
 University of California, Los Angeles
 wwc@cs.ucla.edu

Abstract

This paper deals with the subsequence searching problem under time-warping in sequence databases. Our work is motivated by the observation that subsequence searches slow down quadratically as the average length of data sequences increases. To resolve this problem, the Segment-Based Approach for Subsequence Searches (SBASS) is proposed. The SBASS divides data and query sequences into a series of piece-wise segments, and retrieves all data subsequences that satisfy the two conditions: 1) the number of segments is the same as the number of segments in a query sequence, and 2) the distance of every segment pair is less than or equal to a tolerance. Our segmentation scheme allows segments to have different lengths; thus we employ the time warping distance as a similarity measure for each segment pair.

For efficient retrieval of similar subsequences, we extract feature vectors from all data segments exploiting their monotonically changing properties, and build a spatial index using feature vectors. Using this index, queries are processed with the four steps: 1) R-tree filtering, 2) feature filtering, 3) successor filtering, and 4) post-processing. The effectiveness of our approach is verified through experiments on synthetic data sets.

1 Introduction

There have been many research efforts [1, 6, 7] for efficient similarity searches in sequence databases using the Euclidean distance as a similarity measure. However, recent techniques [9, 10, 11, 14] tend to favor the time warping distance for its higher accuracy and wider applicability at the expense of high computation cost. Time warping is a generalization method for comparing discrete sequences to sequences of continuous values, and has been used extensively in matching of voice, audio, and medical data (e.g., electrocardiograms) [12]. To find the minimum difference between two sequences, time warping allows each element of a sequence to match one or more neighboring elements of another sequence.

The time warping distance can be applied to both whole sequence and subsequence searches. First, let us consider the computation cost for whole sequence searches. Given a data sequence \vec{X} and a query sequence \vec{Q} , the time warping distance has the complexity $O(|\vec{X}||\vec{Q}|)$ where $|\vec{X}|$ and $|\vec{Q}|$ are the lengths of \vec{X} and \vec{Q} , respectively. With M data sequences whose average length is \bar{L} , whole sequence searches require the complexity $O(M\bar{L}|\vec{Q}|)$. Thus, the cost increases linearly both in the total number of data sequences and in the average length of data sequences.

Now, let us consider the computation cost for subsequence searches. A data sequence \vec{X} with length L contains $L(L+1)/2$ subsequences. With M data sequences whose average length is \bar{L} , subsequence searches have the complexity $O(M\bar{L}^2|\vec{Q}|)$. Thus, the cost increases linearly in the total number of data sequences but *quadratically* in the average length of data sequences.

The analysis for subsequence searches is supported by our experimental results in Figure 1 and Figure 2. Figure 1 shows the elapsed times of sequential scan for subsequence searches with increasing numbers of data sequences and Figure 2 shows those with increasing average lengths of data sequences. We used random-walk data sequences. We observe that search times increase quadratically with increasing average lengths of data sequences while maintaining the linearity with increasing total numbers of data sequences.

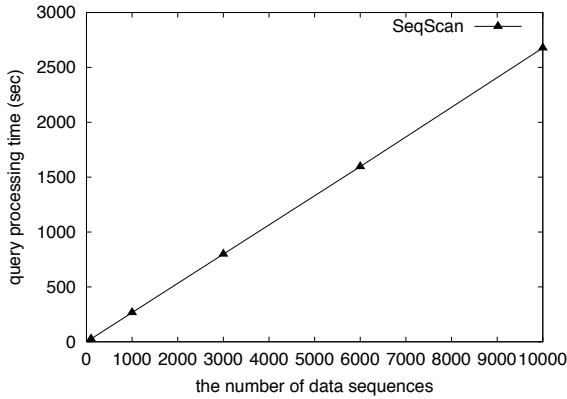


Figure 1: Subsequence searches with increasing numbers of data sequences. The average length of data sequences is 200.

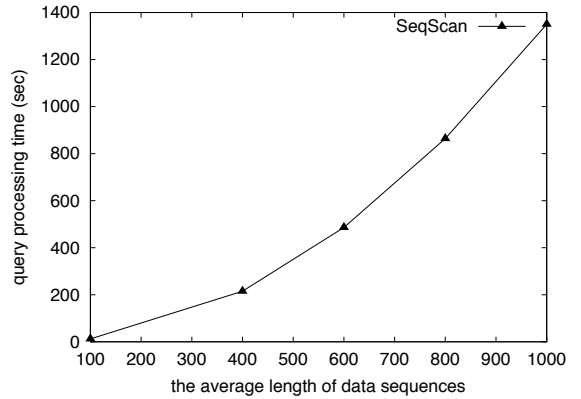


Figure 2: Subsequence searches with increasing average lengths of data sequences. The total number of data sequences is 200.

Typically, spatial index structures such as the R-tree have been widely used to speed up (sub)sequence searches. However, spatial access methods which are based on the triangular inequality cannot avoid *false dismissals* [1] under time-warping since the time warping distance does not satisfy the triangular inequality [14]. To overcome this problem, [11] proposed a subsequence searching technique based on a categorized suffix tree [3, 13] that does not assume the triangular inequality. Even though this technique

achieved a significant speedup, search times still increased quadratically with the average length of data sequences.

Thus, search performance would degrade seriously in typical database environment where a large number of long sequences are stored. In this paper, we tackle this problem and propose a novel scheme to resolve it. The primary goal of this paper is to make subsequence searches linear to the average length of data sequences. To achieve this goal, we employ the Segment-Based Approach for Subsequence Searches (SBASS) and propose an efficient indexing technique for the SBASS.

The SBASS divides data and query sequences into a series of piece-wise segments with monotonically changing values and retrieves all data subsequences that satisfy the two conditions: 1) the number of segments is the same as the number of segments in a query sequence, and 2) the distance of every segment pair is less than or equal to a tolerance. Since the lengths of segments could be different, the time warping distance is used as a similarity measure for segment pairs.

For efficient retrieval of similar subsequences without false dismissals, we extract a feature vector from each segment and build an index structure that consists of the three filters: R-tree filter, feature filter, and successor filter. Using boundary values of segments, the R-tree filter retrieves the set of candidate segments that are similar to a query segment. The feature filter further refines these candidate segments using the remaining features. Finally, the successor filter selects candidate subsequences exploiting the ordering relationship of candidate segments.

The rest of this paper is organized as follows. A brief overview of (sub)sequence searching problems is described in Section 2. In Section 3, the SBASS and its similarity measure are defined. The index construction and the query processing methods are presented in Section 4 and Section 5, respectively. The effectiveness of the proposed approach is verified by the experimental results in Section 6. Finally, Section 7 summarizes the paper and suggests future research directions.

2 Related Work

Recently, several approaches for fast retrieval of similar sequences have been recently proposed. In [1], whole sequences are converted into the frequency domain by the Discrete Fourier Transform (DFT) and are subsequently mapped into low-dimensional points by selecting the first few DFT coefficients. Similar sequences are efficiently retrieved by utilizing the R^* -tree. This technique has been extended to locate similar subsequences in [6]. However, these approaches are not applicable to sequences of different lengths since both of them use the Euclidean distance as a similarity measure.

Some approaches in [4, 14, 11] permit the matching of sequences with different lengths. [4] employs the modified version of edit distance, and considers two sequences similar if a majority of elements match. In [14], the time warping distance is used as a similarity measure with the two-step filtering process: a

FastMap [5] index filter followed by a lower-bound distance filter. Since the modified editing distance and the time warping distance are very expensive, both [4] and [14] just focus on whole sequence searches. [11] presents a new access method for subsequence searches with the time warping distance. Using a categorized suffix tree as an index structure and two lower-bound distance functions as index filters, [11] retrieves similar subsequences without false dismissals. However, its computation complexity is quadratic to the average length of data sequences.

Recently, segment-based subsequence searching algorithms have been proposed in [9, 10]. [9] converts data sequences into ordered lists of piece-wise linear segments and compares subsequences starting and ending at segment boundaries with a query sequence using the time warping distance. Even though it reduces the search times significantly, its piece-wise linear representation loses much information on data segments and its query processing time is still quadratic to the total number of data segments. [10] also suggests the segment-based subsequence searching technique with the accumulated time warping distance as a similarity measure. For efficient query processing, [10] extracts feature vectors from data sequences and builds a suffix tree from categorized representation of feature vectors. Though [10] shows fairly good performance on subsequence searches, the optimal number of categories is hard to determine and a suffix tree is apt to become very large when data sequences are very long.

3 Segment-Based Approach for Subsequence Searches (SBASS)

We introduce the SBASS to reduce the number of subsequences to be compared with a query sequence. The SBASS first converts each sequence into an ordered list of piece-wise segments. Then, it locates the subsequences that start at the first position of a segment and end at the last position of the same segment or one of its following segments. These subsequences are called *aligned subsequences*. A data sequence with N segments has $N(N + 1)/2$ aligned subsequences.

When a query sequence \vec{q} is given, only those aligned subsequences with the same number of segments as the number of segments in \vec{q} are retrieved and then compared with \vec{q} . Consider an aligned subsequence \vec{x} and a query sequence \vec{q} both with K segments. Similarity of \vec{x} and \vec{q} is determined by similarities of corresponding segment pairs of \vec{x} and \vec{q} . If every i -th segment ($1 \leq i \leq K$) of \vec{x} is similar to the i -th segment of \vec{q} , the SBASS makes a decision that \vec{x} and \vec{q} are similar.

Let us analyze the computation cost for the SBASS. If the average number of elements in a segment is C , then there are $|\vec{X}|/C$ segments in a data sequence \vec{X} and $|\vec{q}|/C$ segments in a query sequence \vec{q} . Therefore, in a data sequence \vec{X} , the number of aligned subsequences consisting of $|\vec{q}|/C$ segments is $|\vec{X}|/C - |\vec{q}|/C + 1$. Therefore, the number of aligned subsequences to be compared with a query sequence is linear to the length of a data sequence. Now, we present detailed description of the SBASS. Table 1 shows a list of notations used in this paper.

Notation	Description
\vec{X}	sequence of numeric elements. $\vec{X} = \langle X_1, \dots, X_N \rangle$.
\vec{x}	subsequence or aligned subsequence. $\vec{x} = \langle x_1, \dots, x_N \rangle$.
\vec{X}^S	segmented representation of \vec{X} . $\vec{X}^S[i]$ is the i-th segment of \vec{X} .
$\vec{\alpha}$	segment. $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_N \rangle$.
$F(\vec{\alpha})$	feature vector obtained from $\vec{\alpha}$.
$IP(i)$	interpolation value of the i-th element.
$D(\vec{x}, \vec{y})$	distance function for two aligned subsequences \vec{x} and \vec{y} .
$D_{tw}(\vec{\alpha}, \vec{\beta})$	time warping distance function for two segments $\vec{\alpha}$ and $\vec{\beta}$.
$D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$	distance function for two feature vectors $F(\vec{\alpha})$ and $F(\vec{\beta})$.

Table 1: List of notations.

3.1 Segmentation

There can be many different methods to obtain an ordered list of piece-wise segments from a sequence. To extract useful feature vectors from segments, we take the method that makes every segment have a monotonically changing pattern. A segment $\vec{\alpha} = \langle \alpha_1, \alpha_2, \dots, \alpha_N \rangle$ has a monotonically changing pattern if $\alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_N$ (monotonically increasing pattern) or $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_N$ (monotonically decreasing pattern). Algorithm 1 describes the method to get the segmented representation \vec{X}^S from a sequence \vec{X} . Here, $\langle \rangle$ represents the empty sequence and ‘ \bullet ’ is the binary operator concatenating its operands. For example, $\langle 4, 5, 6 \rangle \bullet \langle 7, 8 \rangle$ makes $\langle 4, 5, 6, 7, 8 \rangle$.

Input : sequence \vec{X}
Output: segmented sequence \vec{X}^S

```

 $\vec{X}^S \leftarrow \langle \rangle;$ 
 $\vec{T} \leftarrow \langle \rangle;$ 
for  $i \leftarrow 1$  to  $|\vec{X}|$  do
    if  $\vec{T} \bullet \langle X_i \rangle$  maintains the monotonically changing property then  $\vec{T} \leftarrow \vec{T} \bullet \langle X_i \rangle;$ 
    else
         $\text{insert } \vec{T} \text{ into } \vec{X}^S;$ 
         $\vec{T} \leftarrow \langle X_i \rangle;$ 
    end if
 $\text{insert } \vec{T} \text{ into } \vec{X}^S;$ 
return  $\vec{X}^S;$ 

```

Algorithm 1: Segmentation

Given a segment $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_N \rangle$, we can define the *interpolation line* connecting the first and last elements, α_1 and α_N . If there is an element α_i that deviates more than the pre-defined threshold from the interpolation line, we may divide the segment into two sub-segments. This sub-division process may proceed recursively until all the elements are within the pre-defined threshold from their interpolation line. This sub-division helps bound each segment more tightly to their interpolation line. However, we

do not consider the sub-division in this paper due to the difficulty in determining the threshold value.

While the number of derived segments depends on the distribution of element values in a sequence, in most cases, it is much smaller than the number of elements in an original sequence. The *compaction ratio* (CR) is used to represent the average number of elements in a segment, $CR = |\vec{X}|/|\vec{X}^S|$.

Example 1: Let us consider a data sequence $\vec{X} = \langle 4, 5, 8, 8, 8, 8, 9, 11, 8, 4, 3, 7, 10 \rangle$. As shown in Figure 3, \vec{X} is segmented to $\vec{X}^S = \langle \langle 4, 5, 8, 8, 8, 8, 9, 11 \rangle, \langle 8, 4, 3 \rangle, \langle 7, 10 \rangle \rangle$ by Algorithm 1. Then, $\vec{X}^S[1] = \langle 4, 5, 8, 8, 8, 8, 9, 11 \rangle$, $\vec{X}^S[2] = \langle 8, 4, 3 \rangle$, and $\vec{X}^S[3] = \langle 7, 10 \rangle$. Since $|\vec{X}| = 13$ and $|\vec{X}^S| = 3$, $CR = 13/3 = 4.3$. ■

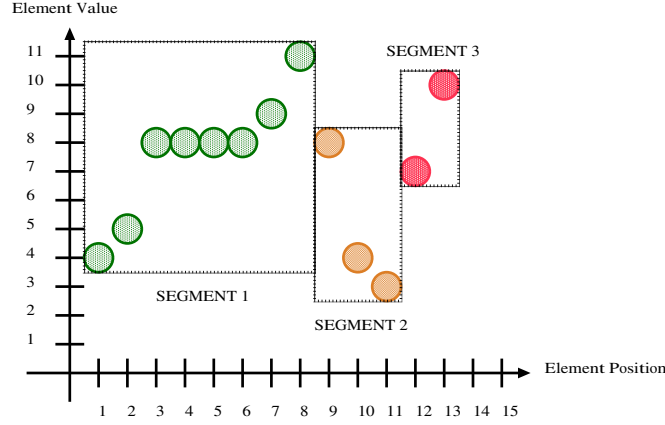


Figure 3: Segmentation Example.

3.2 Similarity Measure

Given two (sub)sequences, we want to make a decision on whether they are similar or not. However, it is not easy to find an appropriate similarity measure because sequences that are qualitatively identical may be quantitatively different. Here, we propose a similarity measure of the SBASS that is intuitive to users and is easily applicable to sequences with different lengths.

Definition 1: Given two aligned subsequences \vec{x} and \vec{y} that have K segments, the distance function $D(\vec{x}, \vec{y})$ is defined as follows:

$$D(\vec{x}, \vec{y}) = \max (D_{tw}(\vec{x}^S[1], \vec{y}^S[1]), D_{tw}(\vec{x}^S[2], \vec{y}^S[2]), \dots, D_{tw}(\vec{x}^S[K], \vec{y}^S[K])),$$

where $D_{tw}(\vec{x}^S[i], \vec{y}^S[i])$ ($i = 1, 2, \dots, K$) is the time warping distance function for two segments $\vec{x}^S[i]$ and $\vec{y}^S[i]$. This implies that if $D(\vec{x}, \vec{y}) = \epsilon$, every segment pair is within the time warping distance ϵ . ■

The time warping distance function [2, 12] allows each element of a segment to match one more neighboring elements of another segment to minimize the distance between two segments. Its formal

definition [12] is given below.

Definition 2: Given two non-empty segments $\vec{\alpha}$ and $\vec{\beta}$, the time warping distance function $D_{tw}(\vec{\alpha}, \vec{\beta})$ is defined as follows:

$$\begin{aligned} D_{tw}(\langle \rangle, \langle \rangle) &= 0 \\ D_{tw}(\vec{\alpha}, \langle \rangle) &= D_{tw}(\langle \rangle, \vec{\beta}) = \infty \\ D_{tw}(\vec{\alpha}, \vec{\beta}) &= |\alpha_1 - \beta_1| + \min \begin{cases} D_{tw}(\vec{\alpha}, \vec{\beta}[2 : -]) \\ D_{tw}(\vec{\alpha}[2 : -], \vec{\beta}) \\ D_{tw}(\vec{\alpha}[2 : -], \vec{\beta}[2 : -]) \end{cases} \end{aligned}$$

where $\langle \rangle$ represents the empty segment and $\vec{\alpha}[2 : -]$ is the subsegment of $\vec{\alpha}$ including all the elements of $\vec{\alpha}$ except for the first. ■

4 Index Construction

For efficient subsequence searches, we employ a modified R-tree consisting of feature vectors extracted from data segments. Before proceeding, let us define the several notations used in the following sections. The maximum and the minimum element values of a segment $\vec{\alpha} = \langle \alpha_1, \alpha_2, \dots, \alpha_N \rangle$ are denoted by $\min(\vec{\alpha})$ and $\max(\vec{\alpha})$, respectively. In a monotonically increasing pattern, $\min(\vec{\alpha}) = \alpha_1$ and $\max(\vec{\alpha}) = \alpha_N$. In a monotonically decreasing pattern, $\min(\vec{\alpha}) = \alpha_N$ and $\max(\vec{\alpha}) = \alpha_1$. Let $\alpha_i - \min(\vec{\alpha})$ be the height h_i of the i -th element. Then, $\vec{\alpha}$ can be rewritten as $\vec{\alpha} = \langle \alpha_1, \min(\vec{\alpha}) + h_2, \dots, \min(\vec{\alpha}) + h_{N-1}, \alpha_N \rangle$.

4.1 Feature Extraction

Given a segment $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_N \rangle$, a 6-tuple feature vector $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$ is extracted exploiting a monotonically changing property.

1. B is the first or the beginning element value ($=\alpha_1$).
2. L is the last element value ($=\alpha_N$).
3. N is the number of elements.
4. H is the sum of heights from the second element to $(N-1)^{th}$ element. That is, $H = \sum_{i=2}^{N-1} h_i = \sum_{i=2}^{N-1} (\alpha_i - \min(\vec{\alpha}))$.
5. Eu is the maximum non-negative deviation value from the interpolation line of $\vec{\alpha}$.
6. Ed is the minimum non-positive deviation value from the interpolation line of $\vec{\alpha}$.

The interpolation line for a segment $\vec{\alpha}$ is obtained by connecting the first and last elements, and thus is expressed as $IP(i) = (\frac{\alpha_N - \alpha_1}{N-1})i + (\alpha_1 - \frac{\alpha_N - \alpha_1}{N-1})$. Since α_1 and α_N are represented by B and L respectively in the feature vector, the interpolation line can also be expressed as $IP(i) = (\frac{L-B}{N-1})i + (B - \frac{L-B}{N-1})$. For the i -th element of $\vec{\alpha}$, its deviation value is defined as $\alpha_i - IP(i)$. From the deviation values of all elements, the maximum non-negative one is assigned to Eu and the minimum non-positive one is assigned to Ed .

Example 2: Let us extract a feature vector from a segment $\vec{\alpha} = \langle 4, 5, 8, 8, 8, 8, 9, 11 \rangle$. $B = 4, L = 11$, and $N = 8$ can be easily obtained. The computation of H is also straightforward. $H = \sum_{i=2}^{N-1} (\alpha_i - \min(\vec{\alpha})) = (5-4) + (8-4) + (8-4) + (8-4) + (8-4) + (9-4) = 22$. Eu and Ed are calculated from the interpolation line $IP(i) = i + 3$. From the set of eight deviation values $\{4 - IP(1), 5 - IP(2), 8 - IP(3), 8 - IP(4), 8 - IP(5), 8 - IP(6), 9 - IP(7), 11 - IP(8)\} = \{0, 0, 2, 1, 0, -1, -1, 0\}$, Eu is assigned 2 and Ed is assigned -1. Therefore, $F(\vec{\alpha}) = \langle 4, 11, 8, 22, 2, -1 \rangle$. ■

4.2 R-tree construction

The R-tree [8] is a height-balanced spatial index structure that efficiently supports both range queries and point queries. Leaf nodes in an R-tree contain entries of the form (MBR, ID) , where ID refers to the identifier of the spatial object indexed and MBR is the minimum bounding rectangle for the spatial object. Non-leaf nodes contain entries of the form $(MBR, ChildPointer)$, where $ChildPointer$ is the address of the child node and MBR is the minimum bounding rectangle that covers all rectangles in the child node.

To filter out segments that are not similar to a query segment, we build the R-tree using the set of feature vectors extracted from data segments. Each feature vector occupies a single entry in leaf nodes. For more effective filtering, the R-tree structure is slightly modified. The modified R-tree uses only the first and last elements (B and L) in a feature vector $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$ as organizing attributes. The remaining four features are kept only in leaf nodes for further filtering. Thus, entry structures for leaf nodes are changed to $(MBR, ID, OtherFeatures)$, where MBR is a 2-dimensional point (B, L) and ID is the identifier of a segment indexed, and $OtherFeatures$ stores the remaining features (N, H, Eu, Ed). Entry structures of non-leaf nodes are not changed.

To locate the actual data sequence from the database efficiently and to find the ordering relationship of segments easily, the identifier of a segment is expressed as $(sequence\#, segment\#)$. If a segment $\vec{\alpha}$ has the identifier (t, s) , its immediately preceding segment $prev(\vec{\alpha})$ has the identifier $(t, s - 1)$ and its immediately following segment $next(\vec{\alpha})$ has the identifier $(t, s + 1)$.

5 Query Processing

In this section, we present a query processing algorithm for efficient retrieval of aligned subsequences similar to a query sequence \vec{q} within a tolerance ϵ . Remember that the distance of \vec{x} and \vec{q} is within ϵ if every segment pair has a time warping distance less than or equal to ϵ . Our algorithm consists of three filters and a post-processing as shown in Figure 4.

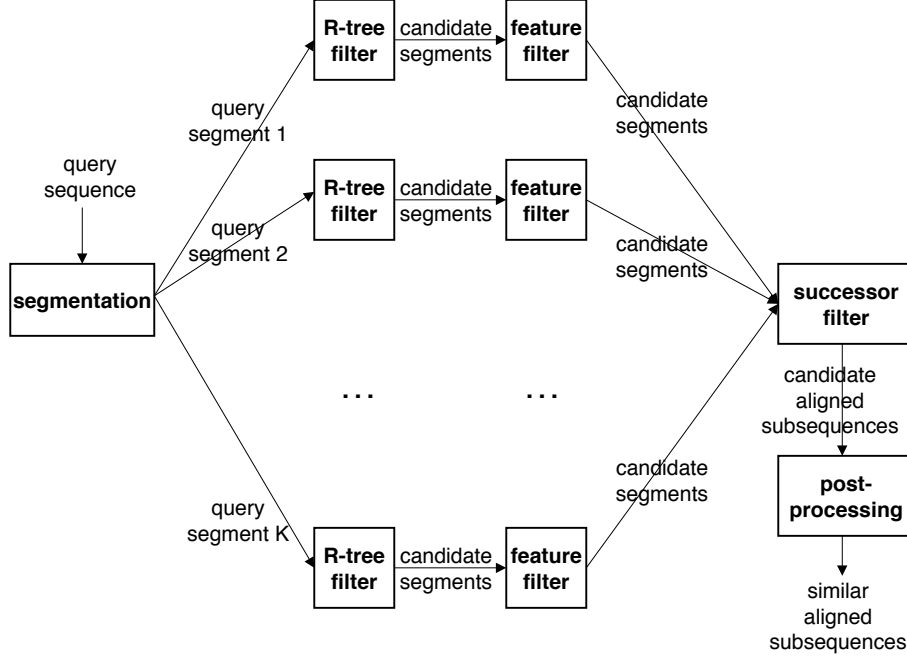


Figure 4: Query Processing.

The algorithm first converts a query sequence \vec{q} into its segmented representation \vec{q}^S . Then, each query segment is sent to the R-tree filter. Using the first and last values of segments, the R-tree filter retrieves the set of candidate segments that are similar to a passed query segment. The R-tree filters could run in the serial or parallel fashion depending on environments. The feature filter further refines the output of the R-tree filter exploiting all the features in leaf nodes of the R-tree. The successor filter receives outputs of all the feature filters and assembles the candidate subsequences using the ordering relationship of candidate segments. Final answers are obtained after post-processing where actual data sequences \vec{x} are retrieved from a database and our similarity measure $D(\vec{x}, \vec{q})$ is applied to discard *false alarms* [1].

5.1 R-tree filter

For a given point (B, L) corresponding to the first and last elements of a query segment and a tolerance ϵ , the R-tree filter constructs a two-dimensional query rectangle $([B - \epsilon, B + \epsilon], [L - \epsilon, L + \epsilon])$ and finds data

points located within a query rectangle. The set of data points belonging to a query rectangle represents the set of data segments whose first and last elements are within $[B-\epsilon, B+\epsilon]$ and $[L-\epsilon, L+\epsilon]$, respectively. Through the following theorem, we claim that data segments outside a query rectangle always have the time warping distance larger than ϵ from the query segment.

Theorem 1: Given a segment $\vec{\alpha}$ whose data point is (B, L) and a segment $\vec{\beta}$ whose data point is (B', L') , if $|B - B'| > \epsilon$ or $|L - L'| > \epsilon$, then $D_{tw}(\vec{\alpha}, \vec{\beta}) > \epsilon$. ■

Proof: Let $m = \langle m_1, m_2, \dots, m_r \rangle$ be the best element mappings from which the time warping distance $D_{tw}(\vec{\alpha}, \vec{\beta})$ is computed. Each mapping m_k ($k = 1, \dots, r$) represents a pair of elements $(\alpha_{f(k)}, \beta_{g(k)})$ where $f(k)$ and $g(k)$ are warping functions whose ranges are $\{1, \dots, |\vec{\alpha}|\}$ and $\{1, \dots, |\vec{\beta}|\}$, respectively. The distance of the mapping m_k is expressed as $|m_k| = |\alpha_{f(k)} - \beta_{g(k)}|$ and the time warping distance between $\vec{\alpha}$ and $\vec{\beta}$ is computed as $D_{tw}(\vec{\alpha}, \vec{\beta}) = \sum_{k=1}^r |m_k|$. By the boundary condition [2] of the time warping distance function, $m_1 = (B, B')$ and $m_r = (L, L')$. Because $\sum_{k=2}^{r-1} |m_k| \geq 0$, if $|B - B'| > \epsilon$ or $|L - L'| > \epsilon$, then $D_{tw}(\vec{\alpha}, \vec{\beta}) > \epsilon$. □

Since the entry of leaf nodes in the R-tree has been changed to include all six features of segments, the R-tree filter produces a set of records $(ID(\vec{\alpha}), F(\vec{\alpha}))$ where $ID(\vec{\alpha})$ and $F(\vec{\alpha})$ are the identifier and the feature vector of $\vec{\alpha}$.

The filtering rate of the R-tree filter depends on a tolerance ϵ given at querying time. As ϵ increases and thus the number of query answers increases, the filtering rate decreases. The filtering rate also depends on the data distribution of segments. If a query rectangle falls in sparse area, the filtering rate increases. On the contrary, it decreases when a query rectangle falls in dense area.

5.2 Feature filter

The feature filter performs the second-round filtering on the output of the R-tree filter by estimating the distance of two segments more accurately utilizing all the extracted features. Before describing the distance function used in the feature filter, let us present the basic notations and concepts.

5.2.1 Value ranges of elements

Given a segment, its feature vector is easily computed. However, going the other way around, exact element values of a segment cannot be obtained from its feature vector. Instead, we can estimate the possible value range of each element. Using the interpolation line $IP(i)$, the maximum deviation value Eu , and the minimum deviation value Ed , it is apparent that the value of the i -th element lies between $IP(i) + Ed$ and $IP(i) + Eu$. The range of the i -th element can be bounded narrower using the obvious fact that the element cannot have a value smaller than the minimum or larger than the maximum of a segment. That is, the value of the i -th element is located between $\max(IP(i) + Ed, \min(\vec{\alpha}))$ and

$\min (IP(i) + Eu, \max(\vec{\alpha}))$. The notations LB_i and UB_i are used to represent the range of the i -th element.

Definition 3: Given a feature vector $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$, the lower-bound and the upper-bound values of the i -th element are defined as follows:

$$\begin{aligned} LB_i &= \max (IP(i) + Ed, \min(\vec{\alpha})). \\ UB_i &= \min (IP(i) + Eu, \max(\vec{\alpha})). \end{aligned}$$

■

Example 3: Let us derive the range of the second element of a segment $\vec{\alpha}$ from its feature vector $F(\vec{\alpha}) = \langle 4, 11, 8, 22, 2, -1 \rangle$. Since the interpolation line connecting (1,4) and (8, 11) is $IP(i) = i + 3$, we obtain that

$$\begin{aligned} LB_2 &= \max (IP(2) + Ed, \min(\vec{\alpha})) = \max (5 - 1, 4) = 4. \\ UB_2 &= \min (IP(2) + Eu, \max(\vec{\alpha})) = \min (5 + 2, 11) = 7. \end{aligned}$$

■

5.2.2 $UBset$ and $LBset$

Given a feature vector $F(\vec{\alpha})$ and a value v that is between $\min(\vec{\alpha})$ and $\max(\vec{\alpha})$, we introduce the two concepts that are useful to characterize a segment $\vec{\alpha}$.

- $UBset(F(\vec{\alpha}), v)$ is the set of UB_i that are smaller than v . $UBset(F(\vec{\alpha}), v) = \{UB_i \mid UB_i < v\}$.
- $LBset(F(\vec{\alpha}), v)$ is the set of LB_i that are larger than v . $LBset(F(\vec{\alpha}), v) = \{LB_i \mid LB_i > v\}$.

We can easily compute $UBset(F(\vec{\alpha}), v)$ and $LBset(F(\vec{\alpha}), v)$ by deriving UB_i and LB_i for the every i -th element from a feature vector. Since both UB_i and LB_i are derived with $O(1)$, $UBset(F(\vec{\alpha}), v)$ and $LBset(F(\vec{\alpha}), v)$ require $O(|\vec{\alpha}|)$ computation time.

Among the values in $UBset(F(\vec{\alpha}), v)$, the largest one $\max(UBset(F(\vec{\alpha}), v))$ is the closest to v . $\max(UBset(F(\vec{\alpha}), v))$ is equal to UB_p if either 1) $UB_p < v$ and $UB_{p+1} \geq v$ (when $\vec{\alpha}$ has an increasing pattern) or 2) $UB_p < v$ and $UB_{p-1} \geq v$ (when $\vec{\alpha}$ has a decreasing pattern) is satisfied. Such element position p is obtained by the following expression.

$$p = \begin{cases} ceil & ((\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1}) - 1) & \text{when } \vec{\alpha} \text{ has an increasing pattern.} \\ floor & ((\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1}) + 1) & \text{when } \vec{\alpha} \text{ has a decreasing pattern.} \end{cases}$$

Here, $\text{ceil}(arg)$ is the function that returns the smallest integer value not less than arg and $\text{floor}(arg)$ is the function that returns the largest integer value not greater than arg .

Similarly, among the values in $LBset(F(\vec{\alpha}), v)$, the smallest one $\min(LBset(F(\vec{\alpha}), v))$ is the closest to v . $\min(LBset(F(\vec{\alpha}), v))$ is equal to LB_q if either 1) $LB_q > v$ and $LB_{q-1} \leq v$ (when $\vec{\alpha}$ has an increasing pattern) or 2) $LB_q > v$ and $LB_{q+1} \leq v$ (when $\vec{\alpha}$ has a decreasing pattern) is satisfied. Such element position q is obtained by the following expression.

$$q = \begin{cases} \text{floor} & ((\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) + 1) & \text{when } \vec{\alpha} \text{ has an increasing pattern.} \\ \text{ceil} & ((\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) - 1) & \text{when } \vec{\alpha} \text{ has a decreasing pattern.} \end{cases}$$

The expressions for the element position p satisfying $UB_p = \max(UBset(F(\vec{\alpha}), v))$ and the element position q satisfying $LB_q = \min(LBset(F(\vec{\alpha}), v))$ are derived in Appendix A. Since both element positions are computed with $O(1)$, the computation of $\max(UBset(F(\vec{\alpha}), v))$ and $\min(LBset(F(\vec{\alpha}), v))$ are also $O(1)$.

Once the element position p satisfying $UB_p = \max(UBset(F(\vec{\alpha}), v))$ is determined, the number of elements in $UBset(F(\vec{\alpha}), v)$ is easily obtained. When $\vec{\alpha}$ has an increasing pattern, $UB_i < v$ for every i from 1 through p . Thus, $|UBset(F(\vec{\alpha}), v)| = p$. When $\vec{\alpha}$ has a decreasing pattern, $UB_i < v$ for every i from p through $|\vec{\alpha}|$. Therefore, $|UBset(F(\vec{\alpha}), v)| = |\vec{\alpha}| - p + 1$. Likewise, the number of elements in $LBset(F(\vec{\alpha}), v)$ is obtained from the element position q satisfying $LB_q = \min(LBset(F(\vec{\alpha}), v))$. $|LBset(F(\vec{\alpha}), v)| = |\vec{\alpha}| - q + 1$ when $\vec{\alpha}$ has an increasing pattern, and $|LBset(F(\vec{\alpha}), v)| = q$ when $\vec{\alpha}$ has a decreasing pattern.

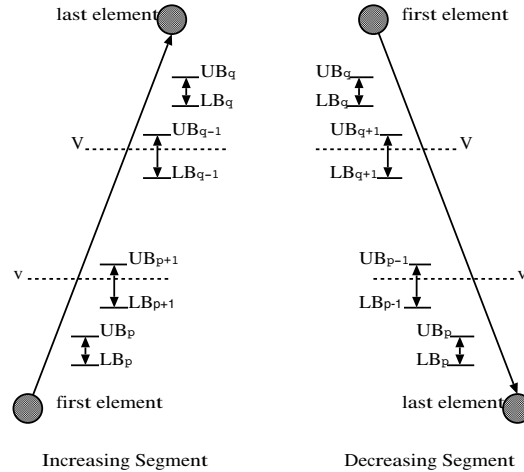


Figure 5: The element position p satisfying $UB_p = \max(UBset(F(\vec{\alpha}), v))$ and the element position q satisfying $LB_q = \min(LBset(F(\vec{\alpha}), v))$.

Example 4: Given a feature vector $F(\vec{\alpha}) = \langle 4, 11, 8, 22, 2, -1 \rangle$ and a value $v = 9$, let us derive $\max(UBset(F(\vec{\alpha}), v))$ and $|UBset(F(\vec{\alpha}), v)|$. Since $\vec{\alpha}$ has an increasing pattern,

$$\begin{aligned}
p &= \text{ceil} \left(\left(\frac{N-1}{L-B} \right) (v - Eu - B + \frac{L-B}{N-1}) - 1 \right) \\
&= \text{ceil} \left(\left(\frac{8-1}{11-4} \right) (9 - 2 - 4 + \frac{11-4}{8-1}) - 1 \right) \\
&= \text{ceil} (3) = 3. \\
UB_3 &= \min (IP(3) + Eu, \max(\vec{\alpha})) \\
&= \min (6 + 2, 11) \\
&= 8 = \max(UBset(F(\vec{\alpha}), v)) \\
|UBset(F(\vec{\alpha}), v)| &= p = 3.
\end{aligned}$$

■

Example 5: Let us consider a feature vector $F(\vec{\alpha}) = \langle 4, 11, 8, 22, 2, -1 \rangle$ again. Given a value $v = 5$, let us derive the $\min(LBset(F(\vec{\alpha}), v))$ and $|LBset(F(\vec{\alpha}), v)|$. Since $\vec{\alpha}$ has an increasing pattern,

$$\begin{aligned}
q &= \text{floor} \left(\left(\frac{N-1}{L-B} \right) (v - Ed - B + \frac{L-B}{N-1}) + 1 \right) \\
&= \text{floor} \left(\left(\frac{8-1}{11-4} \right) (5 - (-1) - 4 + \frac{11-4}{8-1}) + 1 \right) \\
&= \text{floor} (4) = 4. \\
LB_4 &= \max(IP(4) + Ed, \min(\vec{\alpha})) \\
&= \max(7 - 1, 4) \\
&= 6 = \min(LBset(F(\vec{\alpha}), v)). \\
|LBset(F(\vec{\alpha}), v)| &= |\vec{\alpha}| - q + 1 = 8 - 4 + 1 = 5.
\end{aligned}$$

■

5.2.3 Distance function for feature vectors

Now, we define the distance function for two feature vectors $F(\vec{\alpha})$ and $F(\vec{\beta})$ that lower-bounds $D_{tw}(\vec{\alpha}, \vec{\beta})$. Without loss of generality, we assume that $\max(\vec{\alpha}) \geq \max(\vec{\beta})$. If it does not hold, we change their roles. The distance function varies according to the ranges of two feature vectors compared.

Definition 4: Given $\vec{\alpha}, \vec{\beta}$, $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$, and $F(\vec{\beta}) = (B', L', N', H', Eu', Ed')$, the distance function $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ is defined as follows:

- **Case 1:** When $\vec{\alpha}$ and $\vec{\beta}$ are disjoint ($\min(\vec{\alpha}) > \max(\vec{\beta})$),

$$D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) = |B - B'| + |L - L'| + \max \begin{cases} (N - 2)(\min(\vec{\alpha}) - \max(\vec{\beta})) + H \\ (N' - 2)(\min(\vec{\alpha}) - \min(\vec{\beta})) - H' \end{cases}$$

- **Case 2:** When $\vec{\alpha}$ and $\vec{\beta}$ overlap ($\min(\vec{\beta}) \leq \min(\vec{\alpha}) \leq \max(\vec{\beta})$),

$$D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) = |B - B'| + |L - L'| + \\ (|LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1) (\min(LBset(F(\vec{\alpha}), \max(\vec{\beta}))) - \max(\vec{\beta})) + \\ (|UBset(F(\vec{\beta}), \min(\vec{\alpha}))| - 1) (\min(\vec{\alpha}) - \max(UBset(F(\vec{\beta}), \min(\vec{\alpha}))))$$

- **Case 3:** When $\vec{\alpha}$ encloses $\vec{\beta}$ ($\min(\vec{\alpha}) < \min(\vec{\beta})$),

$$D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) = |B - B'| + |L - L'| + \\ (|LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1) (\min(LBset(F(\vec{\alpha}), \max(\vec{\beta}))) - \max(\vec{\beta})) + \\ (|UBset(F(\vec{\alpha}), \min(\vec{\beta}))| - 1) (\min(\vec{\beta}) - \max(UBset(F(\vec{\alpha}), \min(\vec{\beta}))))$$

■

Let $\vec{\beta}$ be the query segment. Then, for records $(ID(\vec{\alpha}), F(\vec{\alpha}))$ returned from the R-tree filter, the distance function $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ is applied. If $D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) > \epsilon$, the record $(ID(\vec{\alpha}), F(\vec{\alpha}))$ is safely filtered out based on the following theorem.

Theorem 2: Given $\vec{\alpha}$, $\vec{\beta}$, $F(\vec{\alpha})$, and $F(\vec{\beta})$, if $D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) > \epsilon$, then $D_{tw}(\vec{\alpha}, \vec{\beta}) > \epsilon$. ■

Proof: See Appendix B. □

Since $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ has the computation complexity $O(1)$, the feature filter requires the complexity $O(N)$ where N is the number of records returned from the R-tree filter. The feature filter increases the filtering rate because $D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) \geq |B - B'| + |L - L'|$. The result of the feature filter forms candidate segments.

5.3 Successor filter

The successor filter stitches candidate segments returned from feature filters to construct the set of candidate aligned subsequences exploiting the ordering relationship of candidate segments.

In this process, we stitch two candidate segments $\vec{\alpha}$ and $\vec{\beta}$ into a subsequence $\vec{\alpha} \bullet \vec{\beta}$ when they satisfy the following conditions: 1) $\vec{\beta} = next(\vec{\alpha})$, and 2) $\vec{\beta}$ belongs to the result of the $(i + 1)^{th}$ feature filter if $\vec{\alpha}$ belongs to the result of the i^{th} feature filter. This stitching process starts with the output of the first feature filter and ends with the output of the last feature filter. An ordered set of segments constructed in the successor filter is called a candidate aligned subsequence. Those candidate segments that fail to form candidate aligned subsequences are filtered out in this successor filter.

Finally, the post-processing step 1) receives candidate aligned subsequences \vec{x} from the successor filter, 2) accesses data sequences containing \vec{x} from a database, and 3) applies the similarity measure $D(\vec{x}, \vec{q})$

to discard remaining false alarms.

6 Performance Evaluation

The purpose of this performance evaluation through experiments is two folded: 1) to show that our SBASS successfully performs subsequence searches linearly both to total number of data sequences and to average length of data sequences, and 2) to compare the performance of our approach with that of the sequential scan.

6.1 Performance of the SBASS

In Section 3, we claimed that the SBASS has the linear computation cost both to the total number of data sequences and to the average length of data sequences. To verify this claim, we implemented the SBASS with the sequential scan and measured its performance with random-walk data sequences. The expression for generating data sequences is:

$$\begin{aligned}\vec{X}[0] &= \text{rand}([10, 100]). \\ \vec{X}[i] &= \vec{X}[i-1] + \text{rand}([-10, 10]).\end{aligned}$$

We first increased the number of data sequences from 1,000 to 5,000 while fixing the average length of data sequences at 500. Next, we increased the average length of data sequences from 1,000 to 5,000 while keeping the total number of data sequences 500. Query sequences were generated using the same way as data sequences. The average length of query sequences was one-tenth of data sequences and a tolerance ϵ was determined to get $10^{-2}\%$ answer-ratio. As shown in Figure 6 and Figure 7, the query processing times for the SBASS increased almost linearly both with increasing numbers of data sequences and with increasing lengths of data sequences.

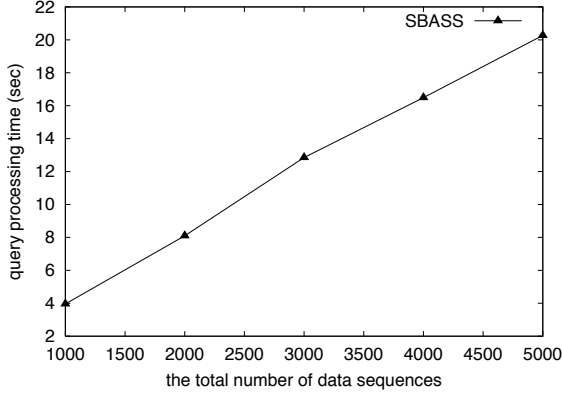


Figure 6: Query processing times for the SBASS with increasing numbers of data sequences. The average length of data sequences is 500.

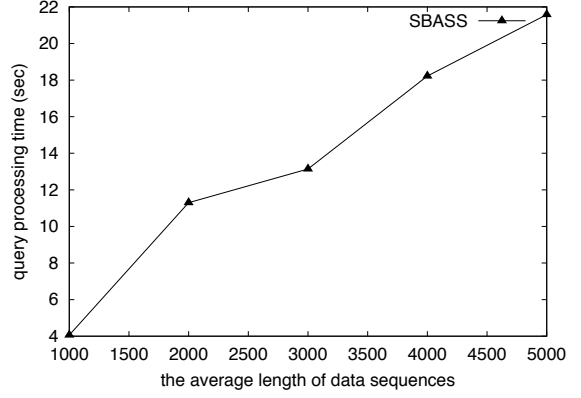


Figure 7: Query processing times for the SBASS with increasing lengths of data sequences. The total number of data sequences is 500.

6.2 Performance comparison

To evaluate the performance of the proposed approach, we compared its query processing time with that of the sequential scan using a data set from UC Irvine KDD Archive (<http://kdd.ics.uci.edu>). The data set, called “Pseudo Periodic Synthetic Time Series”, is specifically designed for testing indexing schemes in time series databases. The data sequences are generated by the following function:

$$\vec{X} = \sum_{i=3}^7 \frac{1}{2^i} \sin(2\pi(2^{2+i} + rand(2^i))\vec{t})$$

where $0 \leq \vec{t} \leq 1$. We generated 100 data sequences whose lengths were all 10,000. Query sequences with average length 1,000 were generated using the same function. Table 2 and Figure 8 show the experimental results. Our approach consistently outperformed the sequential scan and achieved up to 4.98 speed-up.

ϵ	answer-ratio (%)	query processing time (sec.)	
		our method	seqScan
1	0.05	6.01	29.94
5	0.75	17.87	89.16
10	2.72	32.25	145.47
15	5.09	49.91	192.96
20	8.00	66.17	232.45
25	11.17	82.24	266.57
30	14.31	97.75	297.26

Table 2: Query processing times of our method and sequential scan with increasing tolerance values.

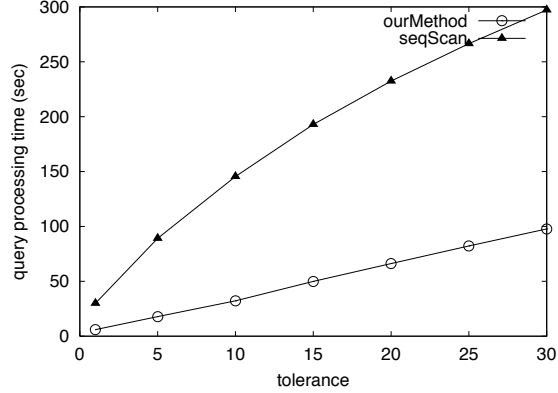


Figure 8: Query processing times of our method and sequential scan with increasing tolerance values.

7 Conclusion

Even though the time warping distance is one of good similarity measures for data sequences, it requires high computation cost. Especially, the performance of subsequence searches degrades seriously when data sequences are very long because the search cost increases quadratically to the length of data sequences.

To alleviate this problem, we proposed the segment-based approach for subsequence searches (SBASS) that compares only those subsequences starting and ending at segment boundaries with a query sequence. For fast retrieval of similar subsequences without false dismissals, we also suggested a novel indexing technique equipped with R-tree filter, feature filter, and successor filter. Performance evaluation through experimental results showed the effectiveness of our proposed approach.

The contributions of our work are: 1) proposing the SBASS and its similarity measure, 2) extracting feature vectors that precisely characterize segments, 3) deriving constant-time lower-bound distance functions exploiting monotonically changing patterns of segments, and 4) suggesting three filtering schemes that accelerate the query processing.

Data sequences may have some noises, thus making segments short. In this situation, we may apply noise elimination methods or categorize element values before building an index structure. The simplest noise elimination method detects the element α_i whose changing ratio from its preceding element α_{i-1} is smaller than the pre-defined threshold. There are several methods to categorize element values. The set of categories can be made to have the same interval or to include the same number of elements.

There still remain several research issues on our indexing technique. Even though our lower-bound distance functions have the constant computation complexity, they need to be tighter or closer to the actual distance functions to increase the filtering rates. The query processing algorithm can also be

improved by rearranging the query segments according to the criteria of selectivity. That is, the query segment with the highest selectivity is applied to the R-tree first and the query segment that has the second selectivity is applied next, and so on. This procedure stops when the number of candidates becomes smaller than the pre-defined threshold. By adding element counter in each R-tree node entry, selectivity of query segments can be easily determined by inspecting several top nodes of the R-tree.

References

- [1] R. Agrawal, C. Faloutsos, A. Swami, "Efficient Similarity Search in Sequence Databases", *Proc. FODO*, pp. 69-84, 1993.
- [2] D. J. Berndt, J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach", *Advances in Knowledge Discovery and Data Mining, AAAI/MIT*, pp. 229-248, 1996.
- [3] P. Bieganski, J. Riedl, J. V. Carlis, "Generalized Suffix Trees for Biological Sequence Data: Applications and Implementation", *Proc. Hawaii Int'l Conf. on System Sciences*, 1994.
- [4] T. Bozkaya, N. Yazdani, M. Özsoyoğlu, "Matching and Indexing Sequences of Different Lengths", *Proc. ACM CIKM*, pp. 128-135, 1997.
- [5] C. Faloutsos, K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets", *Proc. ACM SIGMOD*, pp. 163-174, 1995.
- [6] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases", *Proc. ACM SIGMOD*, pp. 419-429, 1994.
- [7] D. Q. Goldin, P. C. Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation", *Proc. Constraint Programming*, pp. 137-153, 1995.
- [8] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD*, pp. 47-57, 1984.
- [9] E. J. Keogh, M. J. Pazzani, "Scaling up Dynamic Time Warping to Massive Datasets", *Proc. Principles and Practice of Knowledge Discovery in Databases*, 1999.
- [10] S. Park, D. Lee, W. W. Chu, "Fast Retrieval of Similar Subsequences in Long Sequence Databases", *Proc. 3rd IEEE Knowledge and Data Engineering Exchange Workshop (KDEX)*, pp. 60-67, 1999.
- [11] S. Park, W. W. Chu, J. Yoon, C. Hsu, "Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases", *Proc. IEEE ICDE*, pp. 23-32, 2000.
- [12] L. Rabiner, B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.
- [13] G. A. Stephen, *String Searching Algorithms*, World Scientific Publishing, 1994.
- [14] B.-K. Yi, H. V. Jagadish, C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping", *Proc. IEEE ICDE*, pp. 201-208, 1998.

A Appendix: Element positions p and q

Given $\vec{\alpha}$, $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$, and a value v between $\min(\vec{\alpha})$ and $\max(\vec{\alpha})$, the derivation of expressions for the element position p satisfying $UB_p = \max(UBset(F(\vec{\alpha}), v))$ and the element position q satisfying $LB_q = \min(LBset(F(\vec{\alpha}), v))$ is given below.

- **Case 1: expression for the element position p**

Since UB_p is $\min(IP(p) + Eu, \max(\vec{\alpha}))$, either $IP(p) + Eu < v$ or $\max(\vec{\alpha}) < v$ should hold to satisfy $UB_p < v$. However, $\max(\vec{\alpha}) < v$ cannot hold because v is between $\min(\vec{\alpha})$ and $\max(\vec{\alpha})$. Therefore,

$$IP(p) + Eu < v.$$

Because $IP(p) = (\frac{L-B}{N-1})p + (B - \frac{L-B}{N-1})$,

$$(\frac{L-B}{N-1})p + (B - \frac{L-B}{N-1}) + Eu < v.$$

When $\vec{\alpha}$ has a increasing pattern ($B < L$),

$$p < (\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1}).$$

To make $UB_p < v$ and $UB_{p+1} \geq v$, p should have the maximum satisfying the inequality. Thus,

$$p = \text{ceil}((\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1}) - 1).$$

Likewise, when $\vec{\alpha}$ has a decreasing pattern ($B > L$), p should have the minimum satisfying $p > (\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1})$. Thus,

$$p = \text{floor}((\frac{N-1}{L-B})(v - Eu - B + \frac{L-B}{N-1}) + 1).$$

- **Case 2: expression for the element position q**

By applying the same logic used in case 1 to LB_q , the following inequality is obtained when $\vec{\alpha}$ has an increasing pattern.

$$q > (\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}).$$

To make $LB_q > v$ and $LB_{q-1} \leq v$, q should have the minimum satisfying the inequality. Thus,

$$q = \text{floor}((\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) + 1).$$

Likewise, if the segment has a decreasing pattern, then q should have the maximum satisfying $q < (\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1})$. Thus,

$$q = \text{ceil}((\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) - 1).$$

■

B Appendix: Proof of Theorem 2

Given $\vec{\alpha}$, $\vec{\beta}$, and $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$, and $F(\vec{\beta}) = (B', L', N', H', Eu', Ed')$, we prove the theorem:

$$\text{if } D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) > \epsilon, \text{ then } D_{tw}(\vec{\alpha}, \vec{\beta}) > \epsilon.$$

The above theorem can be proved by showing that $D_{tw}(\vec{\alpha}, \vec{\beta}) \geq D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ for any segment pair $\vec{\alpha}$ and $\vec{\beta}$. With $m = \langle m_1, m_2, \dots, m_r \rangle$ as the best element mappings, $D_{tw}(\vec{\alpha}, \vec{\beta}) = |B - B'| + |L - L'| + \sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}|$. $f(k)$ and $g(k)$ are functions having the following properties : 1) for each i ($2 \leq i \leq N-1$), there is at least one k where $f(k) = i$, and 2) for each j ($2 \leq j \leq N'-1$), there is at least one k' where $g(k') = j$. With the assumption that $\max(\vec{\alpha}) \geq \max(\vec{\beta})$, the proofs are given separately for each arrangement of ranges.

- **Case 1:** When $\vec{\alpha}$ and $\vec{\beta}$ are disjoint ($\min(\vec{\alpha}) > \max(\vec{\beta})$),

$$\sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| \geq \sum_{k=2}^{r-1} |\alpha_{f(k)} - \max(\vec{\beta})| \geq \sum_{i=2}^{N-1} |\alpha_i - \max(\vec{\beta})|.$$

Similarly,

$$\sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| \geq \sum_{k=2}^{r-1} |\min(\vec{\alpha}) - \beta_{g(k)}| \geq \sum_{j=2}^{N'-1} |\min(\vec{\alpha}) - \beta_j|.$$

Therefore,

$$\sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| \geq \max\left(\sum_{i=2}^{N-1} |\alpha_i - \max(\vec{\beta})|, \sum_{j=2}^{N'-1} |\min(\vec{\alpha}) - \beta_j|\right).$$

Since $\alpha_i = \min(\vec{\alpha}) + h_i$ and $\beta_j = \min(\vec{\beta}) + h_j$,

$$\begin{aligned} \sum_{i=2}^{N-1} |\alpha_i - \max(\vec{\beta})| &= \sum_{i=2}^{N-1} |\min(\vec{\alpha}) - \max(\vec{\beta})| + \sum_{i=2}^{N-1} h_i = (N-2)(\min(\vec{\alpha}) - \max(\vec{\beta})) + H. \\ \sum_{j=2}^{N'-1} |\min(\vec{\alpha}) - \beta_j| &= \sum_{j=2}^{N'-1} |\min(\vec{\alpha}) - \min(\vec{\beta})| - \sum_{j=2}^{N'-1} h_j = (N'-2)(\min(\vec{\alpha}) - \min(\vec{\beta})) - H'. \end{aligned}$$

Thus,

$$\sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| \geq \max \left\{ \begin{array}{l} (N-2)(\min(\vec{\alpha}) - \max(\vec{\beta})) + H \\ (N'-2)(\min(\vec{\alpha}) - \min(\vec{\beta})) - H' \end{array} \right.$$

As a result, we conclude that

$$D_{tw}(\vec{\alpha}, \vec{\beta}) \geq D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$$

- **Case 2:** when $\vec{\alpha}$ and $\vec{\beta}$ overlap ($\min(\vec{\beta}) \leq \min(\vec{\alpha}) \leq \max(\vec{\beta})$),

We consider only the case where both $\vec{\alpha}$ and $\vec{\beta}$ are increasing. The other cases are proved in the same fashion. Let p be the element position satisfying $UB_p = \max(UBset(F(\vec{\beta}), \min(\vec{\alpha})))$ and let

q be the element position satisfying $LB_q = \min(LBset(F(\vec{\alpha}), \max(\vec{\beta})))$. Then,

$$\begin{aligned} \sum_{i=q}^{N-1} |\alpha_i - \max(\vec{\beta})| &\geq \sum_{i=q}^{N-1} |LB_i - \max(\vec{\beta})| \geq (N-q)(LB_q - \max(\vec{\beta})). \\ \sum_{j=2}^p |\min(\vec{\alpha}) - \beta_j| &\geq \sum_{j=2}^p |\min(\vec{\alpha}) - UB_j| \geq (p-1)(\min(\vec{\alpha}) - UB_p). \end{aligned}$$

In the best element mappings $m = \langle m_1, m_2, \dots, m_r \rangle$, we can find the set of k where $f(k)$ is in $\{q, q+1, \dots, N-1\}$ and the set of k' where $g(k')$ is in $\{2, 3, \dots, p\}$. Note that two sets of such k and k' are mutually exclusive. Therefore,

$$\begin{aligned} \sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| &\geq \sum_{i=q}^{N-1} |\alpha_i - \max(\vec{\beta})| + \sum_{j=2}^p |\min(\vec{\alpha}) - \beta_j| \\ &\geq (N-q)(LB_q - \max(\vec{\beta})) + (p-1)(\min(\vec{\alpha}) - UB_p). \end{aligned}$$

Note that $(N-q) = |LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1$, $LB_q = \min(LBset(F(\vec{\alpha}), \max(\vec{\beta})))$, $(p-1) = |UBset(F(\vec{\beta}), \min(\vec{\alpha}))| - 1$, and $UB_p = \max(UBset(F(\vec{\beta}), \min(\vec{\alpha})))$. Therefore,

$$\begin{aligned} (N-q)(LB_q - \max(\vec{\beta})) &= (|LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1)(\min(LBset(F(\vec{\alpha}), \max(\vec{\beta}))) - \max(\vec{\beta})). \\ (p-1)(\min(\vec{\alpha}) - UB_p) &= (|UBset(F(\vec{\beta}), \min(\vec{\alpha}))| - 1)(\min(\vec{\alpha}) - \max(UBset(F(\vec{\beta}), \min(\vec{\alpha}))))). \end{aligned}$$

As a result, we conclude that

$$D_{tw}(\vec{\alpha}, \vec{\beta}) \geq D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$$

- **Case 3:** when $\vec{\alpha}$ encloses $\vec{\beta}$ ($\min(\vec{\alpha}) < \min(\vec{\beta})$),

We consider only the case where both $\vec{\alpha}$ and $\vec{\beta}$ are increasing. The other cases are proved in the same fashion. Let p be the element position satisfying $UB_p = \max(UBset(F(\vec{\alpha}), \min(\vec{\beta})))$ and let q be the element position satisfying $LB_q = \min(LBset(F(\vec{\alpha}), \max(\vec{\beta})))$. Then,

$$\begin{aligned} \sum_{i=q}^{N-1} |\alpha_i - \max(\vec{\beta})| &\geq \sum_{i=q}^{N-1} |LB_i - \max(\vec{\beta})| \geq (N-q)(LB_q - \max(\vec{\beta})). \\ \sum_{i=2}^p |\min(\vec{\beta}) - \alpha_i| &\geq \sum_{i=2}^p |\min(\vec{\beta}) - UB_i| \geq (p-1)(\min(\vec{\beta}) - UB_p). \end{aligned}$$

In the best element mappings $m = \langle m_1, m_2, \dots, m_r \rangle$, we can find the set of k where $f(k)$ is in $\{q, q+1, \dots, N-1\}$ and the set of k' where $f(k')$ is in $\{2, 3, \dots, p\}$. Note that the two sets of such k and k' are mutually exclusive. Therefore,

$$\begin{aligned} \sum_{k=2}^{r-1} |\alpha_{f(k)} - \beta_{g(k)}| &\geq \sum_{i=q}^{N-1} |\alpha_i - \max(\vec{\beta})| + \sum_{i=2}^p |\min(\vec{\beta}) - \alpha_i| \\ &\geq (N-q)(LB_q - \max(\vec{\beta})) + (p-1)(\min(\vec{\beta}) - UB_p). \end{aligned}$$

Note that $(N-q) = |LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1$, $LB_q = \min(LBset(F(\vec{\alpha}), \max(\vec{\beta})))$, $(p-1) = |UBset(F(\vec{\alpha}), \min(\vec{\beta}))| - 1$, and $UB_p = \max(UBset(F(\vec{\alpha}), \min(\vec{\beta})))$. Therefore,

$$\begin{aligned} (N-q)(LB_q - \max(\vec{\beta})) &= (|LBset(F(\vec{\alpha}), \max(\vec{\beta}))| - 1)(\min(LBset(F(\vec{\alpha}), \max(\vec{\beta}))) - \max(\vec{\beta})). \\ (p-1)(\min(\vec{\beta}) - UB_p) &= (|UBset(F(\vec{\alpha}), \min(\vec{\beta}))| - 1)(\min(\vec{\beta}) - \max(UBset(F(\vec{\alpha}), \min(\vec{\beta}))))). \end{aligned}$$

As a result, we conclude that

$$D_{tw}(\vec{\alpha}, \vec{\beta}) \geq D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$$

■