Comput Syst Sci & Eng (2007) 1-2: 37-46
© 2007 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

SBASS: Segment based approach for subsequence searches in sequence databases

Sanghyun Park*, S-W Kim† and W W Chu‡

*Department of Computer Science, Yonsei University, Korea. Email sanghyun@cs.yonsei.ac.kr
†Sang-Wook Kim, College of Information and Communications, Hanyang University, Korea. Email: wook@hanyang.ac.kr
‡Wesley W. Chu, Department of Computer Science, University of California, Los Angeles. Email: wwc@cs.ucla.edu

This paper investigates the subsequence searching problem under time warping in sequence databases. Time warping enables to find sequences with similar changing patterns even when they are of different lengths. Our work is motivated by the observation that subsequence searches slow down quadratically as the total length of data sequences increases. To resolve this problem, we propose the Segment-Based Approach for Subsequence Searching Technique (SBASS), which modifies the similarity measure from time warping to piece-wise time warping and limits the number of possible subsequences to be compared with a query sequence. That is, the SBASS divides a data sequence $\vec{X}$ and a query sequence $\vec{q}$ into piece-wise segments and compares $\vec{q}$ with only those subsequences which consist of n consecutive segments of $\vec{X}$. Here, n is the number of segments in $\vec{q}$. For efficient retrieval of similar subsequences, we extract feature vectors from all data segments exploiting their monotonically changing properties, and build a multi-dimensional index. Using this index, queries are processed with four steps: (1) index filtering, (2) feature filtering, (3) successor filtering, and (4) post-processing. The effectiveness of our approach is verified through experiments on synthetic data sets.

Keywords: Similarity Search, Sequence Database, Time Warping Distance, Segmentation

1. INTRODUCTION

The sequence database is a set of data sequences, each of which is an ordered list of elements [1]. Sequences of stock prices, money exchange rates, temperature data, product sales data, and company growth rates are the typical examples of sequence databases [2, 8]. Similarity search is an operation that finds sequences or subsequences whose changing patterns are similar to that of a given query sequence [1, 2, 8]. Similarity search is of growing importance in many new applications such as data mining and data warehousing [6, 17].

There have been many research efforts [1, 7, 8, 10, 17] for efficient similarity searches in sequence databases using the Euclidean distance as a similarity measure. However, recent techniques [13–15, 18] 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 transformation that allows any sequence element to replicate itself as many times as needed without extra costs [18]. For example, two sequences $\vec{X} = \langle 20, 21, 21, 20, 20, 23, 23, 23\rangle$ and $\vec{Q} = \langle 20, 20, 21,20, 23\rangle$ can be identically transformed into $\langle 20, 20, 21, 21, 20, 20, 23, 23, 23\rangle$ by time warping. The time warping distance is defined as the smallest distance between two sequences transformed by time warping. While the Euclidean distance can be used only when two sequences compared are of the same length, the time warping distance can be applied to any two sequences of arbitrary lengths. Therefore, the time warping distance fits well with the databases where sequences are of different lengths.

The time warping distance can be applied to both whole sequence and subsequence searches. Let us first 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 $O(|\vec{X}|\ |\vec{Q}|)$ [3] where $|\vec{X}|$ and $|\vec{Q}|$ are the lengths of $\vec{X}$ and $\vec{Q}$, respectively. With m data sequences whose average length is n, whole sequence searches require the computation complexity $O(mn|\vec{Q}|)$. Thus, the cost increases linearly both in the total number of data sequences and in the average length of data sequences.

Let us now consider the computation cost for subsequence searches. Let $\vec{q}$ be a query sequence submitted for subsequence searches. $\vec{X}[i:j]$ denotes a subsequence of $\vec{X}$ containing elements in positions i through j. $\vec{X}[i:-]$ denotes a suffix of $\vec{X}$ starting at i^{th} position. That is, $\vec{X}[i:-] = \vec{X}[i:|\vec{X}|]$. The time warping distance between a subsequence $\vec{X}[i:j]$ and a query sequence $\vec{q}$ can be obtained during the distance computation between $\vec{X}[i:-]$ and $\vec{q}$. Therefore, the number of distance computations for subsequence searches between $\vec{X}$ and $\vec{q}$ is same as the number of suffixes in $\vec{X}$. Since $\vec{X}$ has $|\vec{X}|$ suffixes whose average length is $O(|\vec{X}|)$, subsequence searches require the computation complexity $O(|\vec{X}|^2|\vec{q}|)$. Thus the computation complexity is $O(mn^2|\vec{q}|)$ in a database of m sequences with average length n. As a result, 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 Figures 1 and 2. Figure 1 shows the elapsed times of scan-based approach 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.

Therefore, subsequence searching performance would degrade seriously in a 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 the performance of subsequence matching linear to the average length of data sequences, thereby improving the scalability significantly, especially without seriously deforming the original similarity model based on time warping. 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 modifies the similarity measure from time warping to piece-wise time warping and limits the number of possible subsequences to be compared with a query sequence. That is, the SBASS divides a data sequence $\vec{X}$ and a query sequence $\vec{q}$ into piece-wise segments and compares $\vec{q}$ with only those subsequences which consist of n consecutive segments of $\vec{X}$. Here, n is the number of segments in $\vec{q}$. By dividing a problem into smaller subproblems as such, the SBASS achieves a considerable performance improvement in subsequence matching. Also, the SBASS preserves the properties of the original similarity model since it basically uses the time warping distance in its similarity model.

In the matching of similar sequences, it is important to prevent the occurrence of *false dismissals* [1]. A false dismissal refers to a (sub)sequence that is actually similar to a query sequence but which is judged dissimilar by the system. For efficient retrieval of similar subsequences without *false dismissals*, we extract feature vectors from all data segments exploiting their monotonically changing properties, and build a multi-dimensional index on them. Using this index structure, queries are processed with four steps: (1) `index filtering` that retrieves the set of candidate segments similar to a segment in a query sequence, (2) `feature filtering` that further refines candidate segments, (3) `successor filtering` that selects candidate subsequences exploiting the ordering relationship of candidate segments, and (4) `post-processing` that retrieves final answers after discarding *false alarms* [1]. A false alarm refers to a (sub)sequence that is included in the set of candidate (sub)sequences but which is not similar to a query sequence.

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,

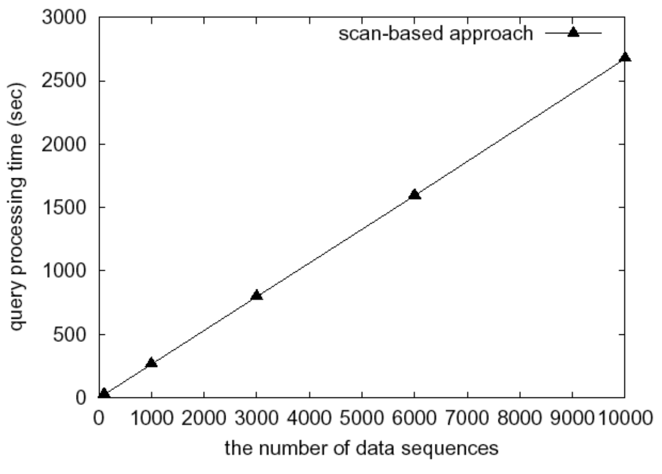

Figure 1 Subsequence searches with increasing numbers of data sequences. The average length of data sequences is 200.

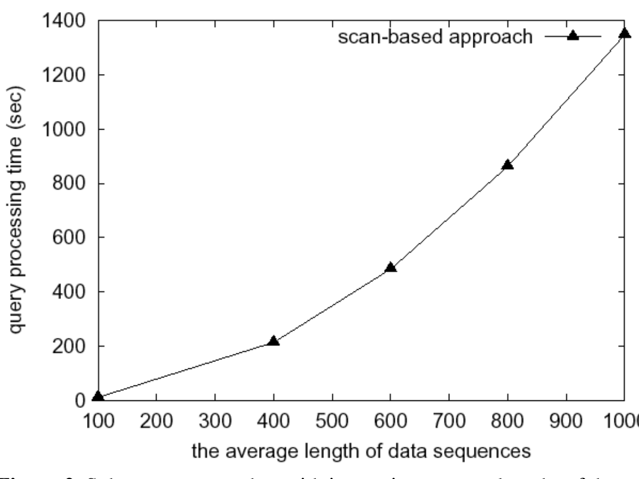

Figure 2 Subsequence searches with increasing average lengths of data sequences. The total number of data sequences is 200

Section 7 summarizes the paper and suggests future research directions.

2. RELATED WORK

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 [8]. 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, 15, 18] permit the matching of sequences with different lengths. [4] employs the modified version of the edit distance, and considers two sequences similar if a majority of elements match. In [18], the time warping distance is used as a similarity measure with the two-step filtering process: a FastMap [9] 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 [18] just focus on whole sequence searches. [15] 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, [15] retrieves similar subsequences without false dismissals. However, its computation complexity is still quadratic to the average length of data sequences.

Recently, segment-based subsequence searching algorithms have been proposed in [13, 14]. [13] converts a data sequence into an ordered list of piece-wise linear segments using the best fitting line and applies the modified time warping similarity measure sequentially. [14] also suggests the segment-based subsequence searching technique with the accumulated time warping distance as a similarity measure. For efficient query processing, [14] extracts feature vectors from data sequences and builds a suffix tree from categorized representation of feature vectors. Though [14] 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. SBASS: SEGMENT-BASED APPROACH FOR SUBSEQUENCE SEARCHES

We introduce the SBASS that modifies the similarity measure from time warping to piece-wise time warping and reduces the number of subsequences to be compared with a query sequence. The SBASS first converts each data sequence into an ordered list of piece-wise segments. At query processing time, the SBASS also converts a query sequence into an ordered list of piece-wise segments. Let n be the number of segments in a query sequence $\vec{q}$. The SBASS compares $\vec{q}$ with only those subsequences $\vec{x}$ which consist of n consecutive segments of a data sequence $\vec{X}$. Let $\vec{x}^S$ and $\vec{q}^S$ be the segmented representations of $\vec{x}$ and $\vec{q}$, respectively. $\vec{x}^S[i]$ denotes the i^{th} segment of $\vec{x}$ and $\vec{q}^S[i]$ the i^{th} segment of $\vec{q}$. The similarity of $\vec{x}$ and $\vec{q}$ is not determined by the time warping distance between $\vec{x}$ and $\vec{q}$ but by the time warping distance of each segment pair, $x^S[i]$ and $\vec{q}^S[i]$. If every segment pair has the time warping distance less than or equal to the user-specified threshold, the SBASS makes a decision that x and $\vec{q}$ are similar.

Let us analyze the computation cost for the SBASS. Let c be the average number of elements in a segment. The cost for computing the time warping distance of each segment pair is $O(c^2)$. The number of segments in $\vec{q}$ is $|\vec{q}|/c$. Therefore, the cost for computing the similarity of $\vec{x}$ and $\vec{q}$ is $O(c^2|\vec{q}|/c) = O(c|\vec{q}|)$. Since the number of segments in $\vec{X}$ is $|\vec{X}|/c$, the number of subsequences consisting of $|\vec{q}|/c$ consecutive segments of $\vec{X}$ is $|\vec{X}|/c - |\vec{q}|/c + 1$. Thus the cost for subsequence searches between $\vec{X}$ and $\vec{q}$ is $(|\vec{X}|/c - |\vec{q}|/c + 1)\ (c|\vec{q}|) = |\vec{q}|(|\vec{X}| - |\vec{q}| + c)$. Therefore, the cost becomes linear to the length of $\vec{X}$. Now, we present detailed description of the SBASS.

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 $\vec{\alpha} = (\alpha_1, ..., \alpha_N)$ has a monotonically changing pattern if $\alpha_1 \leq .... \leq \alpha_N$ (monotonically increasing pattern) or $\alpha_1 \geq .. \geq \alpha_N$ (monotonically decreasing pattern). For example, a data sequence $\vec{X} = \langle 4, 5, 8, 8, 8, 8, 9, 11, 8, 4, 3, 7, 10 \rangle$ is segmented to $\vec{X}^S = \langle\langle 4, 5, 8, 8, 8, 8, 9, 11\rangle, \langle 8, 4, 3\rangle, \langle 7, 10\rangle\rangle$ in our segmentation scheme.

The process of segmentation begins with an empty segment α. It then tries to append into α each element $\vec{X}[i]$ of a data sequence $\vec{X}$ in the increasing order of i. If α becomes non-monotonic when $\vec{X}[i]$ has just been appended to α, $\vec{X}[i]$ is taken out of α and a new segment begins with $\vec{X}[i]$ as its first element. This segmentation process guarantees that an ordered list of piece-wise segments is determined 'uniquely' from any data sequence.

Given a segment $\vec{\alpha} = \langle \alpha_1, ..., \alpha_N \rangle$, we can define the *interpolation line* connecting the first and the 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 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.

3.2 Noise reduction

Data sequences may contain some noises and the proposed segmentation scheme is very sensitive to small noises. As a result, segments may get very short and/or, more critically, similar sequences may yield totally different segments. To alleviate the impact of this problem, it would be better to apply an appropriate noise reduction method as a pre-processing step of the segmentation process. The simplest noise reduction method just ignores the element $\vec{X}[i]$ when its changing ratio from its preceding element $\vec{X}[i-1]$ is smaller than the pre-defined threshold. It is also possible to employ *k*-moving average transformation [5, 12] for noise reduction. Given a data sequence $\vec{X}$ and a moving average coefficient k, each element $\vec{X}[i]$ is transformed into $\vec{X_k}[i]$ by k-moving average transformation:

$$\vec{X_k}[i] = \frac{\vec{X}[i] + \vec{X}[i+1] + \ldots + \vec{X}[i+k-1]}{k} = \frac{\sum_{j=i}^{i+k-1} \vec{X}[j]}{k}$$

Categorization [15] can also be utilized as an alternative for noise reduction. Categorization is an operation which divides the value ranges of elements into a set of non-overlapping categories. Via categorization, each element is converted to the symbol of the category to which the element belongs. Note that two elements whose values are slightly different from each other may be represented by the same symbol. As a result, the impact of small noises can be reduced. We assume that data sequences are delivered to the segmentation process after being preprocessed through one of the above noise reduction methods.

3.3 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 of different lengths.

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

$$D(\vec{x}, \vec{y}) = \max \begin{cases} D_{tw}(\vec{x^S}[1], \vec{y^S}[1]) \\ \ldots \\ D_{tw}(\vec{x^S}[k], \vec{y^S}[k]) \end{cases}$$

where $D_{tw}(\vec{x^S}[i], \vec{y^S}[i])$ is the time warping distance function for two segments $\vec{x^S}[i]$ and $\vec{y^S}[i]$. This implies that if $D(\vec{x}, \vec{y}) \leq \varepsilon$ every segment pair is within the time warping distance ε. The time warping distance function [3, 16] 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 [16] is given below.

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

$$D_{tw}(\langle\rangle, \langle\rangle) = 0$$

$$D_{tw}(\vec{\alpha}, \langle\rangle) = D_{tw}(\langle\rangle, \vec{\beta}) = \infty$$

$$D_{tw}(\vec{\alpha}, \vec{\beta}) = |\alpha_1 - \beta_1| + min \begin{cases} D_{tw}(\vec{\alpha}, \vec{\beta}[2:-1]) \\ D_{tw}(\vec{\alpha}[2:-], \vec{\beta}) \\ D_{tw}(\vec{\alpha}[2:-], \vec{\beta}[2:-]) \end{cases}$$

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

4. INDEXING

The SBASS scheme can be processed using the scan-based approach. The scan-based approach reads all the data sequences from a database and computes $D(\vec{x}, \vec{q})$ between a data subsequence $\vec{x}$ and a query sequence $\vec{q}$. However, for efficient query processing, we propose an index-based approach, which uses a multi-dimensional index to discard non-qualifying subsequences rapidly.

Let us first define the notations used in the following sections. The maximum and the minimum element values of a segment $\vec{\alpha} = \langle\alpha_1, \ldots, \alpha_N\rangle$ 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 ith element. Then, $\vec{\alpha}$ can be rewritten as $\vec{\alpha} = \langle\alpha_1, min(\vec{\alpha}) + h_2, \ldots, min(\vec{\alpha}) + h_{N-1}, \alpha_N\rangle$.

4.1 Feature extraction

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

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

The interpolation line for a segment $\vec{\alpha}$ is obtained by connecting the first and the last elements, and thus is expressed as:

$$IP(i) = \left(\frac{\alpha_N - \alpha_i}{N-1}\right) i + \left(\alpha_1 - \frac{\alpha_N - \alpha_i}{N-1}\right)$$

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:

$$IP(i) = \left(\frac{L-B}{N-1}\right) i + \left(B - \frac{L-B}{N-1}\right)$$

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 non-negative maximum one is assigned to *Eu* and the non-positive minimum one is assigned to *Ed*.

As an example, 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. $H = \sum_{i=1}^{N} (\alpha_i - min(\vec{\alpha})) = (4-4) + (5-4) + (8-4) +$

(8 – 4) + (8 – 4) + (8 – 4) + (9 – 4) + (11 – 4) = 29. *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) g = {0, 0, 2, 1, 0,–1,–1}, *Eu* is assigned 2 and *Ed* is assigned –1. Therefore, $F(\vec{\alpha}) = \langle 4, 11, 8, 29, 2, -1 \rangle$.

4.2 Index construction

To filter out segments that are not similar to a segment in a query sequence, we build an R-tree using the set of feature vectors extracted from data segments. A R-tree [11] is a height-balanced spatial index structure that efficiently supports both range queries and point queries. Each feature vector occupies a single entry in leaf nodes. For more effective filtering, we slightly change the structure of the R-tree. The modified R-tree uses only the first and the 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

This section presents a query processing algorithm for efficient retrieval of those subsequences that are within a tolerance ε from a query sequence $\vec{q}$. Remember that the distance of *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 3. The algorithm first converts a query sequence $\vec{q}$ into its segmented representation $\vec{q^S}$. Then, each query segment is sent to the index filter. Using the first and the last values of segments, the index filter retrieves the set of candidate segments that are similar to a passed query segment. The index filters could run in the serial or parallel fashion depending on environments. The feature filter further refines the output of the index filter exploiting all the features in leaf nodes of the index. The successor filter 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 Index filter

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

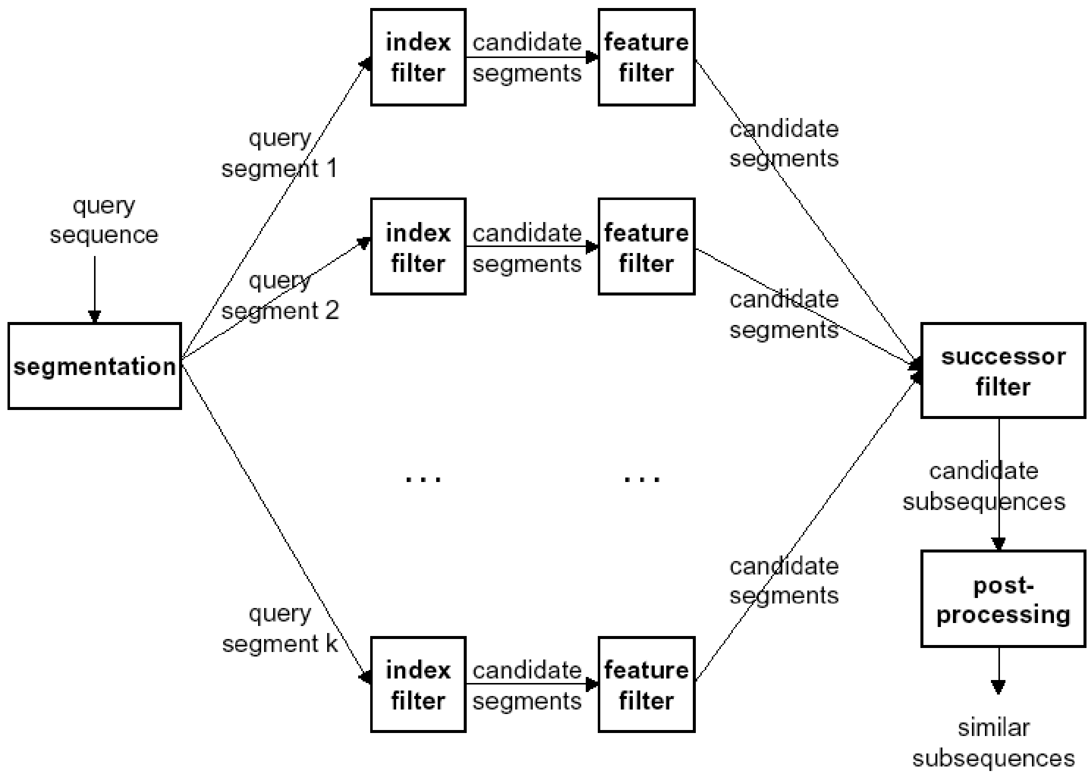

Figure 3 Query processing

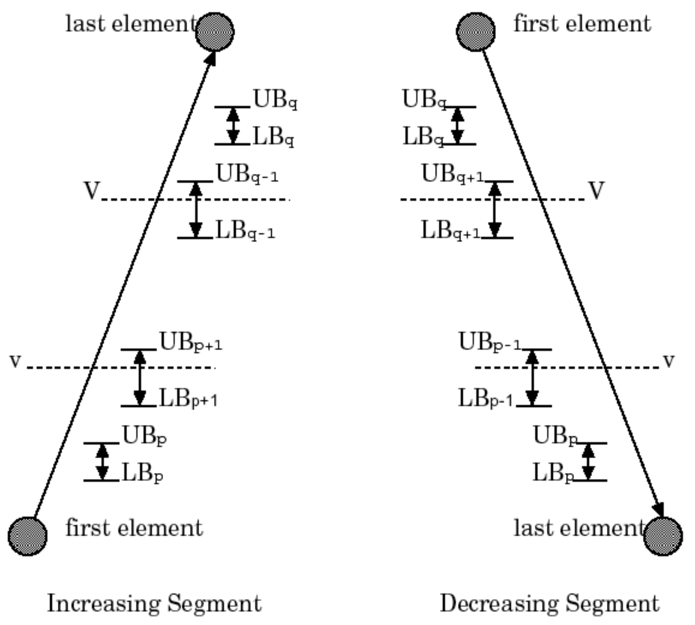

Figure 4 The element position $p \in \{\underline{v}\}$ whose upper-bound value is the closet to v and the element position $q \in \{\overline{V}\}$ whose lower-bound value is the closest to V .

respectively. Through the following theorem, we claim that data segments outside a query rectangle always have the time warping distance larger than 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 $|B - B'| > \varepsilon$ or $|L - L'| > \varepsilon$ then $D_{tw}(\vec{\alpha}, \vec{\beta}) > \varepsilon$. ■

Proof: Let $m = (m_1, m_2, ..., m_r)$ be the best element mappings from which the time warping distance $D_{tw}(\vec{\alpha},\vec{\beta})$ is computed. Each mapping m_k $(k = 1,...,r)$ represents a pair of elements $(\alpha_{f(k)}, \beta_{g(k)})$ where $f(k)$ and $g(k)$ are warping functions whose ranges are $\{1, ..., |\vec{\alpha}|\}$ and $\{1, ..., |\vec{\beta}|\}$, 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 [3] 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 $|B - B'| > \varepsilon$ or $|L - L'| > \varepsilon$, then $D_{tw}(\vec{\alpha}, \vec{\beta}) > \varepsilon$. □

5.2 Feature filter

The feature filter performs the second-round filtering on the output of the index filter by estimating the distance of two segments more accurately using the distance function D_{ft}. Before describing the distance function used in the feature filter, let us present the basic notations and concepts.

5.2.1 $LB_{\alpha i}$ and $UB_{\alpha i}$

Given a feature vector $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$, let us derive the lower-bound and the upper-bound values of α_i. It is apparent that α_i lines between $IP(i) + Ed$ and $IP(i) + Eu$. The range of α_i can be bounded narrower using the obvious fact that α_i cannot have a value smaller than $min(\vec{\alpha})$ or larger than $max(\vec{\alpha})$. Therefore, α_i is between $max(ip(i) + Ed, min(\vec{\alpha}))$ and $min(IP(i) + Eu, max(\vec{\alpha}))$. $LB\alpha_i$ and $UB\alpha_i$ denote the lower-bound and the upper-bound values of α_i. Then, $LB\alpha_i = max(IP(i) + Ed, min(\vec{\alpha}))$ and $UB\alpha_i = min(IP(i) + Eu, max(\vec{\alpha}))$.

5.2.2 $\{\underline{v}\}$ and $\{\overline{V}\}$

Let $\{\underline{v}\}$ be the set of element positions whose upper-bound values are smaller than v. We want to find the element position $p \in \{\underline{v}\}$ whose upper-bound value is the closest to v. When a segment has an increasing pattern, p is the largest one in $\{\underline{v}\}$. When a segment has a decreasing pattern, p is the smallest one in $\{\underline{v}\}$. The element position p is directly obtained from a feature vector. (The detailed derivation is described in Appendix A.)

$$p = \begin{cases} \text{ceil}\left(\left(\frac{N-1}{L-B}\right)\left(v - Eu - B + \frac{L-B}{N-1}\right) - 1\right) & \\ & \text{for an increasing pattern} \\ \text{floor}\left(\left(\frac{N-1}{L-B}\right)\left(v - Eu - B + \frac{L-B}{N-1}\right) + 1\right) & \\ & \text{for a decreasing pattern} \end{cases}$$

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

Once the element position p is determined, the number of elements in $\{\underline{v}\}$ is easily obtained. When a segment has a increasing pattern, $UB_{\alpha i} < v$ for every i from 1 through p. Thus, $|\{\underline{v}\}| = p$. When a segment has a decreasing pattern, $UB_{\alpha i} < v$ for every i from p through N. Therefore $|\{\underline{v}\}| = N - p + 1$.

Similarly, let $\{\underline{v}\}$ be the set of element positions whose lower-bound values are larger than v. We want to find the element position $p \in \{\overline{v}\}$ whose lower-bound value is the closest to v. When a segment has an increasing pattern, q is the smallest one in $\{\overline{v}\}$. When a segment has a decreasing pattern, q is the largest one in $\{\overline{v}\}$. The element position q is directly obtained from a feature vector. (The detailed derivation is described in Appendix A.)

$$p = \begin{cases} \text{floor}\left(\left(\frac{N-1}{L-B}\right)\left(v - Ed - B + \frac{L-B}{N-1}\right) + 1\right) \\ \qquad\qquad \text{for an increasing pattern} \\ \text{ceil}\left(\left(\frac{N-1}{L-B}\right)\left(v - Ed - B + \frac{L-B}{N-1}\right) - 1\right) \\ \qquad\qquad \text{for a decreasing pattern} \end{cases}$$

When a segment has a increasing pattern, $LB_{\alpha i} > v$ for every i from q through N. Thus, $|\{\bar{v}\}| = N - q + 1$. When a segment has a decreasing pattern, $LB_{\alpha i} > v$ for every i from 1 through q. Therefore $|\{\bar{v}\}| = q$.

5.2.3 Distance function D_{ft}

The distance function D_{ft} of the feature filter is adapted from the lower-bound distance function D_{lb} introduced in [18]. Let us describe D_{lb} first. Without loss of generality, we assume that $max(\vec{\alpha}) \geq max(\vec{\beta})$. If it does not hold, we exchange their roles. According to the ranges of $\vec{\alpha}$ and $\vec{\beta}$, there are three relationships; disjoint, overlap and enclose. The definition of D_{lb} is given below.

Definition 3: Given two segments $\vec{\alpha} = \langle \alpha_1, ..., \alpha_N \rangle$ and $\vec{\beta} = \langle \beta_1, ..., \beta_N \rangle$, the distance function D_{lb} is defined as follows [18]:

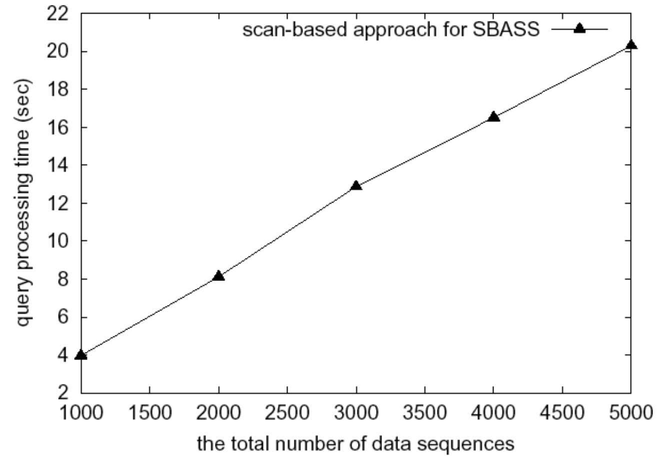

Figure 5 Query processing times for the SBASS with increasing numbers of data sequences. The average length of data sequences is 500

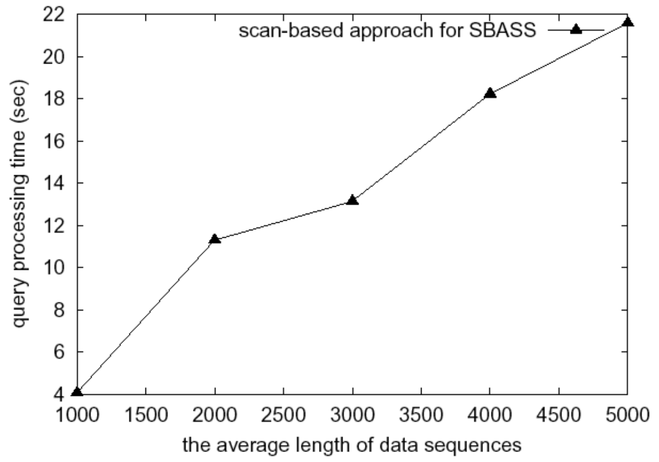

Figure 6 Query processing times for the SBASS with increasing lengths of data sequences. The total number of data sequences is 500.

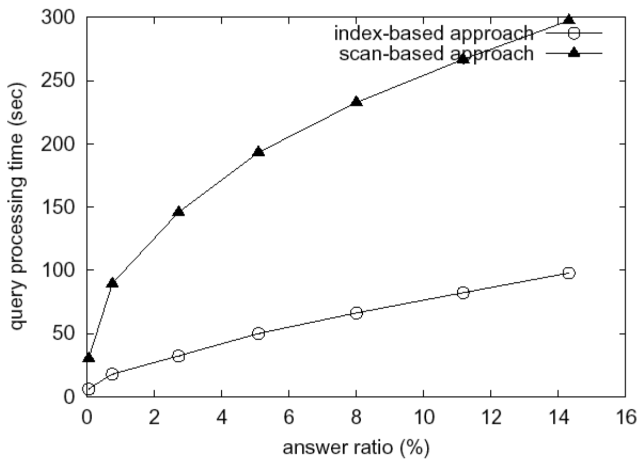

Figure 7 Query processing times of our method and sequential scan with increasing tolerance values

$$D_{lb}(\vec{\alpha}, \vec{\beta}) = \begin{cases} max\left(\sum_{i=1}^{N}(\alpha_i - max(\vec{\beta})), \sum_{j=1}^{N'}(min(\vec{\alpha}) - (\beta_j)\right) \\ \qquad\qquad \text{for disjoint} \\ \sum_{\alpha_i > max(\vec{\beta})}(\alpha_i - max(\vec{\beta})) + \sum_{\beta j < min(\vec{\alpha})}(min(\vec{a}) - \beta_j) \\ \qquad\qquad \text{for overlap} \\ \sum_{\alpha_i > max(\vec{\beta})}(\alpha_i - max(\vec{\beta})) + \sum_{\alpha_i < min(\vec{\beta})}(min(\vec{\beta}) - \alpha_i) \\ \qquad\qquad \text{for enclose} \end{cases}$$

D_{lb} consistently underestimates the time warping distance D_{tw} for preventing the occurrence of false dismissals. Since $D_{lb}(\vec{\alpha}, \vec{\beta})$ reads all elements in α and β for the distance computation, $D_{lb}(\vec{\alpha}, \vec{\beta})$ has the complexity $O(|\vec{\alpha}| + |\vec{\beta}|)$. We now define a distance function D_{ft} of the feature filter. What we want to do is to rephrase D_{lb} in terms of feature vectors, $F(\vec{\alpha}) = (B, L, N, H, Eu, Ed)$ and $F(\vec{\beta}) = (B, L, N, H, Eu, Ed)$. Let us consider the case that $\vec{\alpha}$ and $\vec{\beta}$ are disjoint.

$$\begin{aligned} D_{lb}(\vec{\alpha}, \vec{\beta}) &= max\left(\sum_{i=1}^{N}(\alpha_i - max(\vec{\beta})), \sum_{j=1}^{N'}(min(\vec{\alpha}) - (\beta_j)\right) \\ &= max\left(\sum_{i=1}^{N}(h_i + min(\vec{\alpha}) - max(\vec{\beta})), \right. \\ &\qquad \left. \sum_{j=1}^{N'}(min(\vec{\alpha}) - h'_j - min(\vec{\beta}))\right) \\ &= max(H + N(min(\vec{\alpha}) - max(\vec{\beta})), -H' + N' \\ &\qquad (min(\vec{\alpha}) - min(\vec{\beta}))) \\ &= D_{ft}(F(\vec{\alpha}), F(\vec{\beta})) \end{aligned}$$

The next case is that $\vec{\alpha}$ and $\vec{\beta}$ overlap. Since we cannot obtain α_i and β_j from $F(\vec{\alpha})$ and $F(\vec{\beta})$, we use their lower-bound and upper-bound values instead. Let $\breve{A}$ be the set of element positions in $\vec{\alpha}$ whose lower-bound values are larger than $max(\vec{\beta})$ and $q \in \breve{A}$ be the element position whose lower-bound value is the closest to $max(\vec{\beta})$. Likewise, let $\breve{B}$ be the set of element positions in $\vec{\beta}$ whose upper-bound values are smaller than $min(\vec{\alpha})$ and $p \in \breve{B}$ be the element position whose upper-bound is the closest to $min(\vec{\alpha})$. We are now ready to derive D_{ft} for the case that $\vec{\alpha}$ and $\vec{\beta}$ overlap.

$$\begin{aligned} D_{lb}(\vec{\alpha}, \vec{\beta}) &= \sum_{\alpha_i > max(\vec{\beta})}(\alpha_i - max(\vec{\beta})) + \sum_{\beta j < min(\vec{\alpha})}(min(\vec{\alpha}) - \beta_j) \\ &\geq \sum_{i \in \breve{A}}(LB_{\alpha_i} - max(\vec{\beta})) + \sum_{j \in \breve{B}}(min(\vec{\alpha}) - UB_{\beta_j}) \\ &\geq \sum_{i \in \breve{A}}(LB_{\alpha_q} - max(\vec{\beta})) + \sum_{j \in \breve{B}}(min(\vec{\alpha}) - UB_{\beta_p}) \end{aligned}$$

Table 1 Query processing times of our method and sequential scan with in creasing tolerance values

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

$$= |\breve{A}|\,(LB_{\alpha_q} - max(\vec{\beta})) + |B|(min(\vec{\alpha}) - UB_{\beta p})$$
$$= D_{ft}(F(\vec{\alpha}),F(\vec{\beta}))$$

The final case is that $\vec{\alpha}$ encloses $\vec{\beta}$. Let $\breve{A}$ be the set of element positions in $\vec{\alpha}$ whose lower-bound values are larger than $max(\vec{\beta})$ and $q \in \breve{A}$ be the element position whose lower-bound value is the closest to $max(\vec{\beta})$. Likewise, let $\breve{B}$ be the set of element positions in $\vec{\alpha}$ whose upper-bound values are smaller than $min(\vec{\beta})$ and $p \in \breve{B}$ be the element position whose upper-bound is the closest to $min(\vec{\beta})$. We are now ready to derive D_{ft} for the case that $\vec{\alpha}$ encloses $\vec{\beta}$.

$$\begin{aligned} D_{lb}(\vec{\alpha},\vec{\beta}) &= \textstyle\sum_{\alpha_i > max(\vec{\beta})}(\alpha_i - max(\vec{\beta})) + \sum_{\alpha_i < min(\vec{\beta})}(min(\vec{\beta}) - \alpha_i) \\ &\geq \textstyle\sum_{i\in\breve{A}}(LB_{\alpha_i} - max(\vec{\beta})) + \sum_{i\in\breve{B}}(min(\vec{\beta}) - UB\alpha_i) \\ &\geq \textstyle\sum_{i\in\breve{A}}(LB_{\alpha_q} - max(\vec{\beta})) + \sum_{i\in\breve{B}}(min(\vec{\beta}) - UB_{\alpha_p}) \\ &= |\breve{A}|(LB_{\alpha_q} - max(\vec{\beta})) + |\breve{B}|(min(\vec{\beta}) - UB_{\alpha_p}) \\ &= D_{ft}(F(\vec{\alpha}),F(\vec{\beta})) \end{aligned}$$

To guarantee no false dismissal, D_{ft} should consistently underestimate the time warping distance D_{tw}. Theorem 2 shows that D_{ft} is a lower-bound function of D_{tw}.

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

Proof: The 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 $\vec{\beta}$. By the definition of $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$, $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ is a lower-bound distance function of $D_{lb}(\vec{\alpha}, \vec{\beta})$. Yi *et al.* [18] proved that $D_{lb}(\vec{\alpha}, \vec{\beta})$ is a lower-bound distance function of $D_{tw}(\vec{\alpha}, \vec{\beta})$. As a result, $D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ is also a lower-bound distance function of $D_{lb}(\vec{\alpha}, \vec{\beta})$. Therefore, $D_{tw}(\vec{\alpha}, \vec{\beta}) \geq D_{ft}(F(\vec{\alpha}), F(\vec{\beta}))$ for any segment pair $\vec{\alpha}$ and $\vec{\beta}$. □

5.3 Successor filter

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

In this process, we stitch two candidate segments $\vec{\alpha}$ and $\vec{\beta}$ into a subsequence $\vec{\alpha} \cdot \vec{\beta}$ 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 and α 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. Those candidate segments that fail to form candidate subsequences are filtered out in this successor filter.

Finally, the post-processing step (1) receives candidate sub sequences $\bar{x}$ from the successor filter, (2) accesses data sequences containing $\bar{x}$ from a database, and (3) applies the similarity measure $D(\bar{x}, q)$ 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 and measured its performance with random-walk data sequences. The expression for generating data sequences is:

$$\vec{X}[0] = \texttt{rand}([10, 100])$$
$$\vec{X}[i] = \vec{X}[i - 1] + \texttt{rand}([-10, 10])$$

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 Figures 5 and 6, the query processing times for the SBASS increased almost linearly both with increasing numbers of data sequences and with increasing lengths of data sequences.

To evaluate the relative merits of different filters proposed in this paper, we measured the filtering ratios of the index filter alone, the combination of the first two filters (i.e., index filter + feature filter), and the combination of all filters (i.e., index filter + feature filter + successor filter). Note that a distance tolerance ε was determined to get 10^{-2}% answer-ratio. The experimental results revealed that their filtering ratios are 62.2%, 88.9% and 97.2%, respectively, on an average.

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 a "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 1 and Figure 7 show the experimental results. Our approach consistently outperformed the sequential scan and achieved up to 4.98 speed-up.

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 sub-sequence 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 the index filter, feature filter, and successor filter. 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 tha accelerate the query processing.

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 icrease 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 index filter 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 index node entry, selectivity of query segments can be easily determined by inspecting several top nodes of the index.

ACKNOWLEDGEMENT

This research was supported in part by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program (Grant: IITA-2005-C1090-0502-0009) supervised by the IITA (Institute of Information Technology Assessment).

REFERENCES

1 **R. Agrawal, C. Faloutsos and A. Swami** Effcient Similarity Search in Sequence Databases, *Proc. FODO*, pp. 69–84, 1993.

2 **R. Agrawal, K. Lin, H. S. Sawhney and K. Shim** Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases, *Proc. VLDB*, pp. 490–501, 1995.

3 **D. J. Berndt and J. Clifford** Finding Patterns in Time Series: A Dynamic Programming Approach, *Advances in Knowledge Discovery and Data Mining, AAAI/MIT*, pp. 229–248, 1996.

4 **T. Bozkaya, N. Yazdani and M. Özsoyoğlu** Matching and Indexing Sequences of Different Lengths, *Proc. ACM CIKM*, pp. 128–135, 1997.

5 **C. Chatfield** *The Analysis of Time-Series: An Introduction*, 3rd Edition, Chapman and Hall , 1984.

6. **M. S. Chen, J. Han and P. S. Yu** Data Mining: An Overview from Database Perspective, *IEEE TKDE*, Vol. 8, No. 6, pp. 866–883, 1996.

7 **K. W. Chu and M. H. Wong** Fast Time-Series Searching with Scaling and Shifting, *Proc. ACM PODS*, pp. 237–248, 1999.

8 **C. Faloutsos, M. Ranganathan and Y. Manolopoulos** Fast Subsequence Matching in Time-Series Databases, *Proc. ACM SIGMOD*, pp. 419-429, 1994.

9 **C. Faloutsos and K. Lin** FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets, *Proc. ACM SIGMOD*, pp. 163–174, 1995.

10 **D. Q. Goldin and P. C. Kanellakis** On Similarity Queries for Time-Serie Data: Constraint Specification and Implementation, *Proc. Constrain Programming*, pp. 137–153, 1995.

11 **A. Guttman** R-trees: A Dynamic Index Structure for Spatial Searching, *Proc. ACM SIGMOD*, pp. 47–57, 1984.

12 **M. Kendall** Time-Siries, 2nd Edition, *Charles Griffin and Company*, 1979.

13 **E. J. Keogh and M. J. Pazzani** Scaling up Dynamic Time Warping to Massive Datasets, *Proc. Principles and Practice of Knowledge Discovery in Databases*, 1999.

14 **S. Park, D. Lee and 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.

15 **S. Park, W. W. Chu, J. Yoon and C. Hsu** Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases, *Proc. IEEE ICDE*, pp. 23–32, 2000.

16 **L. Rabiner and B.-H. Juang** *Fundamentals of Speech Recognition*, Prentice Hall , 1993.

17 **D. Raei and A. Mendelzon** Similarity-Based Queries for Time-Series Data, *Proc. ACM SIGMOD*, pp. 13–24, 1997.

18 **B.-K. Yi, H. V. Jagadish and C. Faloutsos** Efficient Retrieval of Similar Time Sequences Under Time Warping, *Proc. IEEE ICDE*, pp. 201–208, 1998.

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 $p \in \{\underline{v}\}$ whose upper-bound value is the closest to v and the element position $q \in \{\bar{v}\}$ whose lower-bound value is the closest to v is given below. Here $\{\underline{v}\}$ is the set of element positions whose upper-bound values are smaller than v, and $\{\bar{v}\}$ be the set of element positions whose lower-bound values are larger than v.

Case 1: expression for the element position p

Since UB_{α_p} is $min(IP(p) + Eu; max(\vec{\alpha}))$, either $IP(p) + Eu < v$ or $max(\vec{\alpha}) < v$ should hold to satisfy $UB_{\alpha_p} < v$. 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 $UB_{\alpha_p} < v$ and $UB_{\alpha_{p+1}} \geq v$, 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

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

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_{α_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 $LB_{\alpha_q} > v$ and $LB_{\alpha_{q-1}} \leq v$, q should have the minimum satisfing 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

$$q < (\frac{N-1}{L-B})\,(v - Ed - B + \frac{L-B}{N-1}). \text{ Thus,}$$

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