Segment-Based Approach for Subsequence Searches in Sequence Databases

Sanghyun Park
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
shpark@cs.ucla.edu

Sang-Wook Kim
Department of Computer, Information, and Communications Engineering
Kangwon National University, Korea
wook@cc.kangwon.ac.kr

Wesley W. Chu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA
wwc@cs.ucla.edu

Keywords

Similarity search, Sequence database, Segment, Time warping, Indexing technique

ABSTRACT

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 Searches (SBASS), which modifies the similarity measure from time warping to piecewise time warping and limits the number of possible subsequences to be compared with a query sequence.

For efficient retrieval of similar subsequences without false dismissal[1], we extract feature vectors from all data segments exploiting their monotonically changing properties, and build a multi-dimensional index such as R-tree or R*-tree. 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.

1. INTRODUCTION

The sequence database is a set of data sequences, each of which is an ordered list of elements [1]. Similarity search is an operation that finds sequences or subsequences whose changing patterns are similar to that of a given query sequence [1, 2, 5]. Similarity search is of growing importance in many new applications, such as data mining and data warehousing [11].

There have been many research efforts [1, 5, 11] for efficient similarity searches in sequence databases using the Euclidean distance as a similarity measure. However, recent techniques [6, 8, 9, 12] 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 [12]. 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. 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. 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 the 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.

Therefore, subsequence searching performance would degrade seriously in a typical database environment where a

[1] Our approach guarantees no false dismissal with the piecewise time warping similarity measure. It is possible that a subsequence similar to a query in terms of the original time warping distance may not be included in the answer set in our approach.

16th ACM Symposium on Applied Computing Las Vegas, NV USA, 2001

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 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 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.

2. RELATED WORK

Approaches in [4, 8, 12] 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 [12], the time warping distance is used as a similarity measure with the two-step filtering process: a FastMap 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 [12] just focus on whole sequence searches. [8] 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, [8] retrieves similar subsequences without false dismissals [1]. However, its computation complexity is still quadratic to the average length of data sequences.

Recently, segment-based subsequence searching algorithms have been proposed in [6, 9]. [6] 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. [9] also suggests the segment-based subsequence searching technique with the accumulated time warping distance as a similarity measure. For efficient query processing, [9] extracts feature vectors from data sequences and builds a suffix tree from categorized representation of feature vectors. Though [9] 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, which modifies the similarity measure from time warping to piece-wise time warping for reducing 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 k 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 k 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, $\vec{x}^S[i]$ and $\vec{q}^S[i]$. If every segment pair has the time warping distance less than or equal to the user-specified tolerance, the SBASS makes a decision that $\vec{x}$ 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 $\vec{\alpha} = \langle \alpha_1, ..., \alpha_N \rangle$ 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.

3.2 Similarity measure

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:

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

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, 10] 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 is given [10].

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 nonqualifying subsequences rapidly.

Let us first define the notations used in the following sections. The maximum and the minimum element values of a segment $\vec{\alpha} = \langle \alpha_1, ..., \alpha_N \rangle$ 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.

4.1 Feature extraction

Given a segment $\vec{\alpha} = \langle \alpha_1, ..., \alpha_N \rangle$, a 6-tuple feature vector $F(\vec{\alpha}) = (B, L, N, H, DV_{max}, DV_{min})$ 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, $H = \sum_{i=1}^{N} h_i = \sum_{i=1}^{N} (\alpha_i - min(\vec{\alpha}))$.
- DV_{max} is the non-negative maximum deviation value from the interpolation line of $\vec{\alpha}$.
- DV_{min} 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 $IP(i) = (\frac{\alpha_N - \alpha_1}{N-1})i + (\alpha_1 - \frac{\alpha_N - \alpha_1}{N-1})$. 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 non-negative maximum one is assigned to DV_{max} and the non-positive minimum one is assigned to DV_{min}.

4.2 Index construction

We build a multi-dimensional index structure such as R-tree or R*-tree using the set of feature vectors extracted from data segments. From a feature vector $F(\vec{\alpha}) = (B,\ L,\ N,\ H,\ DV_{max},\ DV_{min})$, we use only the first and the last elements (B and L) as organizing attributes for determining a minimum bounding rectangle (MBR). The remaining four features are kept only in leaf nodes for further filtering.

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

Our index-based query processing algorithm consists of three filters and a post-processing as shown in Figure 1. 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 by 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

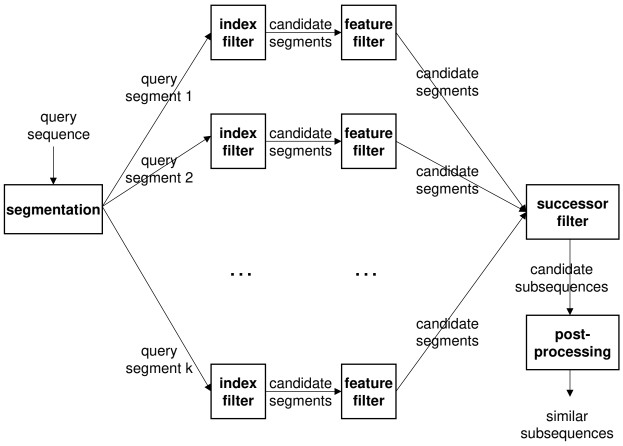

Figure 1: Index-based query processing algorithm

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 $[L-\varepsilon, L+\varepsilon]$, 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. The proof of this theorem is given in [7].

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 $D_{tw}(\vec{\alpha}, \vec{\beta}) > \varepsilon$.*

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 new distance function D_{ft}.

5.2.1 LB_{α_i} and UB_{α_i}

Given a feature vector $F(\vec{\alpha}) = (B, L, N, H, DV_{max}, DV_{min})$, the lower-bound and the upper-bound values of α_i are computed as following. (The derivation procedure is described in [7].)

$$
\begin{aligned}
LB_{\alpha_i} &= max\ (IP(i) + DV_{min},\ min(\vec{\alpha})) \\
UB_{\alpha_i} &= min\ (IP(i) + DV_{max},\ max(\vec{\alpha}))
\end{aligned}
$$

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 derivation process is described in [7]).

$$p = \begin{cases} \text{ceil}((\frac{N-1}{L-B})(v - DV_{max} - B + \frac{L-B}{N-1}) - 1) \text{ (increasing)} \\ \text{floor}((\frac{N-1}{L-B})(v - DV_{max} - B + \frac{L-B}{N-1}) + 1) \text{ (decreasing)} \end{cases}$$

Similarly, let $\{\overline{v}\}$ be the set of element positions whose lower-bound values are larger than v. We want to find the element position $q \in \{\overline{v}\}$ 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 [7].)

$$q = \begin{cases} \text{floor}((\frac{N-1}{L-B})(v - DV_{min} - B + \frac{L-B}{N-1}) + 1) \text{ (increasing)} \\ \text{ceil}((\frac{N-1}{L-B})(v - DV_{min} - B + \frac{L-B}{N-1}) - 1) \text{ (decreasing)} \end{cases}$$

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 [12]. D_{ft} rephrases D_{lb} in terms of feature vectors, $F(\vec{\alpha}) = (B, L, N, H, DV_{max}, DV_{min})$ and $F(\vec{\beta}) = (B', L', N', H', DV'_{max}, DV'_{min})$. 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. When $\vec{\alpha}$ and $\vec{\beta}$ are disjoint, D_{ft} is defined as follows:

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

The next case is that $\vec{\alpha}$ and $\vec{\beta}$ overlap. 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.

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

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 $\vec{\beta}$.

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

To guarantee no false dismissal [1], 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}. The proof of this theorem is given in [7].

THEOREM 2. *Given two segments, $\vec{\alpha}$ and $\vec{\beta}$, and their feature vectors, $F(\vec{\alpha})$ and $F(\vec{\beta})$, if D_{ft} $(F(\vec{\alpha}), F(\vec{\beta})) > \varepsilon$, then D_{tw} $(\vec{\alpha}, \vec{\beta}) > \varepsilon$.*

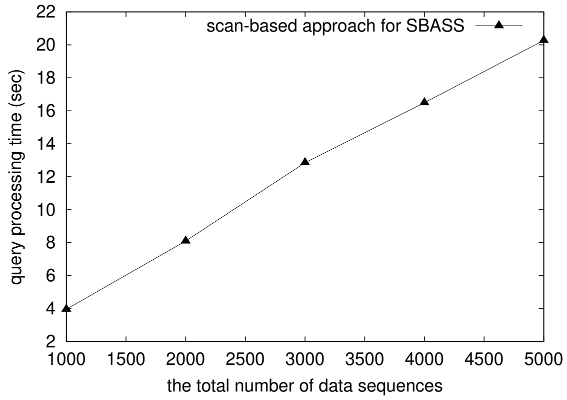

Figure 2: Elapsed times of the scan-based approach for the SBASS scheme with increasing numbers of data sequences. The average length of data sequences is 500.

5.3 Successor filter

The successor filter stitches candidate segments returned from feature filters to construct the set of candidate subsequences by 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 and $\vec{\alpha}$ 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. Those candidate segments that fail to form candidate subsequences are filtered out in this successor filter.

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 index-based query processing approach with that of the scan-based approach.

6.1 Performance of the SBASS

We implemented the scan-based approach for the SBASS scheme and measured its performance with random-walk data sequences [1]. 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. The answer ratio is defined as the number of answers divided by the total number of data subsequences to be compared. As shown in Figure 2 and Figure 3, the elapsed times of the scan-based approach for the SBASS scheme increased almost linearly with increasing numbers and lengths of data sequences.

6.2 Performance comparison

We implemented our index-based query processing approach with R-tree and compared its query processing time with that of the scan-based approach using a “Pseudo Pe-

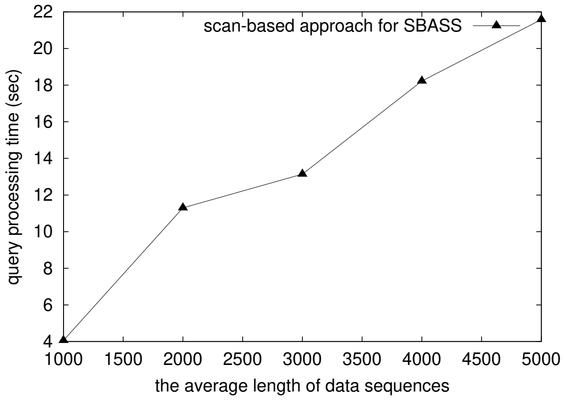

Figure 3: Elapsed times of the scan-based approach for the SBASS scheme with increasing lengths of data sequences. The total number of data sequences is 500.

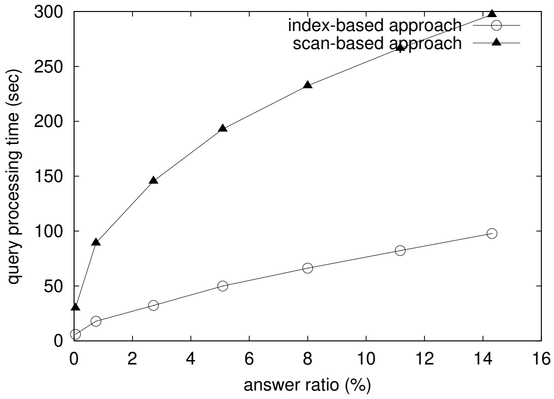

Figure 4: Elapsed times of the index-based approach and the scan-based approach for the SBASS scheme with increasing answer ratios. The answer ratio increases as the corresponding tolerance ε increases.

riodic Synthetic" data set from UC Irvine KDD Archive (http://kdd.ics.uci.edu).

We generated 100 data sequences whose lengths were all 10,000. Query sequences with average length 1,000 were generated using the same function. Figure 4 shows the experimental results with increasing answer ratios. Our index-based approach consistently outperformed the scan-based approach 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 since the search time increases quadratically to the length of data sequences. Therefore, the performance of subsequence searches degrades significantly in a database with long data sequences.

To alleviate this problem, we proposed the segment-based approach for subsequence searches (SBASS), which modifies the similarity measure from time warping to piece-wise time warping to reduce the number of data sequences to be compared. We also proposed an index-based approach which uses filters to reduce the query processing time. The experimental results revealed that the query processing time of the SBASS scheme increased linearly with the numbers and lengths of data sequences even for the scan-based approach. However, the index-based approach yielded significantly lower query processing time as compared to the scan-based approach.

Our segmentation and indexing schemes assume that 1) every segment has a monotonically changing pattern, and 2) the distance of any two subsequences is defined as the maximum distance among all the segment pairs in the subsequences. If these assumptions are not satisfied, we need to use a suffix tree [9] which requires a much larger index space.

Acknowledgments: This research was supported in part by NSF grant (IRI-9619345), NIH grant (PO1CA51198), and the 2000 Basic Research Program (Grant No. 2000-1-51200-006-1) of the KOSEF(Korea Science and Engineering Foundation). Sang-Wook Kim would also like to thank Jung-Hee Seo, Suk-Yeon Hwang, and Grace(Ju-Young) Kim for their encouragement and support.

8. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. FODO*, pages 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. In *Proc. VLDB*, pages 490–501, 1995.

[3] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAAI/MIT, 1996.

[4] T. Bozkaya, N. Yazdani, and M. Ozsoyoğlu. Matching and indexing sequences of different lengths. In *Proc. ACM CIKM*, pages 128–135, 1997.

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

[6] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proc. PKDD*, pages 1–11, 1999.

[7] S. Park, S. Kim, and W. W. Chu. Segment-based approach for subsequence searches in sequence databases. In *Technical Report of UCLA Computer Science Dept. (UCLA-CSD-TR-200022)*, 2000.

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

[9] S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *Proc. 3rd IEEE Knowledge and Data Engineering Exchange Workshop (KDEX)*, pages 60–67, 1999.

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

[11] D. Rafiei and A. Mendelzon. Similarity-based queries for time-series data. In *ACM SIGMOD*, pages 13–24, 1997.

[12] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. IEEE ICDE*, pages 201–208, 1998.