Prefix-querying with an L_1 distance metric for time-series subsequence matching under time warping

Sanghyun Park

Department of Computer Science, Yonsei University, Korea

Sang-Wook Kim

School of Information and Communications, Hanyang University, Korea

Received 13 September 2005
Revised 18 November 2005

Abstract.

This paper discusses the way of processing time-series subsequence matching under time warping. Time warping enables sequences to be found with similar patterns even when they are of different lengths. The *prefix-querying method* is the first index-based approach that efficiently performs time-series subsequence matching under time warping without false dismissals. This method employs the L_∞ distance metric as a base distance function so as to allow users to issue queries conveniently. In this paper, we extend the prefix-querying method for absorbing L_1, which is the most widely used as a base distance function in time-series subsequence matching under time warping, instead of L_∞. We formally prove that the prefix-querying method with the L_1 distance metric does not incur any false dismissals in the subsequence matching. To show its superiority, we conduct performance evaluation via a variety of experiments. The results reveal that our method achieves significant performance improvement over the previous methods, up to 10.7 times, with a data set containing real-world Korean stock data sequences, and up to 180 times with data sets containing a very large volume of synthetic data sequences.

Correspondence to: Sang-Wook Kim, School of Information and Communications, Hanyang University, Korea. E-mail: wook@hanyang.ac.kr

Keywords: time-series databases; subsequence matching; time warping

1. Introduction

A *time-series 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 changes, product sales rates, and company growth rates are typical examples of time-series databases [2–4]. *Sequence matching* is an operation that finds sequences or subsequences whose changing patterns are similar to that of a given query sequence. It is of growing importance in many new applications such as data mining and data warehousing [1, 2, 4–6].

In order to measure the similarity of any two sequences of length n, say X (=[x_1, x_2, . . . , x_n]) and Y (=[y_1, y_2, . . ., y_n]), most earlier approaches map the sequences into points in n-dimensional space and compute the distance function L_p(X, Y) defined as follows.

$$L_p(X,Y) = \sqrt[p]{\sum_{i=0}^{n} (|x_i - y_i|)^p}$$

The L_p distance function has been widely used to measure the similarity of two sequences X and Y of the same length. L_1 is the *Manhattan distance*, L_2 is the *Euclidean distance*, and L_∞ is the *maximum distance* in any pair of corresponding elements [7]. In real applications, two sequences X and Y are considered similar to each other when L_p(X, Y) is smaller than a given tolerance ε [1, 4, 6, 8–11].

Sequence matching performed only with the L_p distance function, however, often fails to search for the data sequences that are actually similar to a query sequence in the user's perspective. To overcome this problem, various types of transformation can be used so as to properly define the similarity model appropriate to target applications. For this purpose, recent work has proposed methods to support transformations such as scaling [2, 9], shifting [2, 9], normalization [10, 12–13], moving average [6, 14], and time warping [15–20].

Among these, time warping is a transformation that allows any sequence element to replicate itself as many times as needed without extra costs [20]. The time warping distance is defined as the smallest distance between two sequences transformed by time warping. Given two sequences S and Q, the time warping distance D_{tw}(S, Q) is defined recursively as follows [18, 20–21]:

Definition 1. *(1)* $D_{tw}(<>, <>) = 0$,
(2) $D_{tw}(S, <>) = D_{tw}(<>, Q) = \infty$,
(3) $D_{tw}(S, Q) = (|L_p(First(S), First(Q))|^p + |min(D_{tw}(S, Rest(Q)), D_{tw}(Rest(S), Q), D_{tw}(Rest(S), Rest(Q)))|^p)^{1/p}$

Here, First(S) is the first element of S, and Rest(S) is a subsequence of S that includes the elements from position 2 to the end. <> denotes a null sequence. The min() is a function to take the minimum value amongst the three arguments. L_p can be properly chosen according to applications, but the Manhattan distance L_1 is the most popular one for time warping distance.

The time warping for two sequences proceeds towards minimizing the time warping distance between the two sequences after the transformation. For example, two sequences S = <20, 21, 21, 20, 20, 23, 23, 23> and Q = <20, 20, 21, 20, 23> can be identically transformed into <20, 20, 21, 21, 20, 20, 23, 23, 23> by time warping, which results in D_{tw}(S, Q) of 0.

As described earlier, the L_p distance can be applied only to two sequences of the same length, However, the time warping distance fits well with databases where sequences are of varying lengths and so the L_p distance cannot be applied [17].[1] The time warping distance has actually been used in voice recognition [22] and electrocardiogram analysis [20], and can also be applied to the analysis of stock prices, temperature changes, and company growth rates [21] in a similar way.

References [17] and [21] proposed an index-based sequence matching method under time warping, and reference [19] proposed the *prefix-querying method* for subsequence matching by extending the index-based method. These two methods employ L_∞ as a base distance function for users' convenience in querying. However, in discussions with other researchers working in the same area, we had been recommended to extend those index-based methods to employ L_1, which is most popular for time warping [15–16, 18, 20, 23]. This paper is a discussion of the points arising from this recommendation [24].

This paper discusses an index-based subsequence matching under time warping. The existing prefix-querying method [19] has not been verified to work correctly when used with L_1 for a base distance function. This paper addresses an extension of the prefix-querying method to take L_1 as a base distance function. We formally prove that the proposed method does not incur any false dismissals in subsequence matching. To show the superiority of our method, we conduct performance evaluation via extensive experiments.

This paper is organized as follows. Section 2 reviews the previous methods for (sub)sequence matching under time warping, and Section 3 discusses and formalizes the extension of the prefix-querying method. Section 4 evaluates the performance of the prefix-querying method. Finally, Section 5 summarizes and concludes the paper.

2. Related work

This section reviews the previous methods for sequence matching under time warping: Naive-Scan, LB-Scan, ST-Filter and LB-Filter. For each method, we discuss (1) the whole matching strategy, (2) the subsequence matching strategy[2] and (3) the characteristics.

2.1. Naive-Scan

2.1.1. Whole matching. It reads all the data sequences from disk and computes the time warping distance D_{tw}(S, Q) between a data sequence S and a query sequence Q using the dynamic programming technique [15]. When D_{tw}(S, Q) is smaller than a given tolerance ε, S is considered to be similar to Q. In computing the time warping distance between S and Q using the dynamic programming technique, each element $T(i, j)$ of the *cumulative distance table T* is constructed by the *recurrence relation* as follows [15]. The dynamic programming technique fills in the cumulative distance table T successively from the bottom to the top as the computation proceeds.

$$T(0, 0) = 0,\ T(0, j) = T(i, 0) = \infty\ (i \geq 1, j \geq 1)$$

$$T(i, j) = |Q[i] - S[j]| + \min(T(i-1, j), T(i, j-1), T(i-1, j-1))\ (i \geq 1, j \geq 1)$$

Table 1 shows a process of computing the time warping distance between S and Q using the cumulative distance table, where L_1 is employed as a base distance function. As a result of those computations, D_{tw}(S, Q) becomes 12.

2.1.2. Subsequence matching. It reads all the data sequences from disk and computes the time warping distance of *every subsequence* S[*i*:*j*], contained in each data sequence S, to a query sequence Q using the dynamic programming technique.

2.1.3. Characteristics. The processing time is excessive since Naive-Scan does not go through the filtering step [17]. That is, there is a serious overhead to access all the data sequences from disk. Also, the CPU processing time for computing D_{tw} between the (sub)sequences X and D is O($|X| * |Q|$), which is quite large. Here, $|X|$ and $|Q|$ denote the sizes of X and Q, respectively. As a result, its search performance degrades seriously in large databases.

2.2. LB-Scan

2.2.1. Whole matching. The filtering step is performed using a *lower-bound function* D_{lb} which returns a value always smaller than the time warping distance D_{tw}. That is, after a data sequence S is accessed from disk, D_{lb}(S, Q) is applied for S and a query sequence Q. If D_{lb}(S, Q) is smaller than a tolerance ε, the post-processing step is performed to compute its actual time warping distance D_{tw}(S, Q). For the computation of D_{tw}(S, Q), a similar way to that of Naive-Scan is used with the dynamic programming technique.

2.2.2. Subsequence matching. After a data sequence S is accessed from disk, D_{lb}(S[*i*:*j*], Q) is applied for *every subsequence* S[*i*:*j*] contained in S and a query sequence Q. Here, if D_{lb}(S[*i*:*j*], Q) returns a value smaller than a tolerance ε, the post-processing step is performed to compute its time warping distance D_{tw}(S[*i*:*j*], Q) using the dynamic programming technique.

2.2.3. Characteristics. Since all the data sequences are accessed from disk without any index structure in the filtering step, the disk access time of this method is equal to that of Naive-Scan. In the filtering step, D_{lb} is computed between all the (sub)sequences X and a query sequence Q. The CPU processing time for computing D_{lb}(X, Q) is O($|X|+|Q|$), which is much smaller than O($|X| * |Q|$) for D_{tw}(X, Q) [7]. The processing time for the post-processing step can be significantly improved since the (sub)sequences that are unlikely to be included in a final result are eliminated in advance through the filtering step [17]. Thus, when many (sub)sequences are excluded via the filtering step, the performance improves considerably.

Table 1
An example of computing the time warping distance between S = <4, 5, 6, 7, 6, 6> and Q = <3, 4, 3>

6	16	11	12
6	13	9	10
7	10	7	8
6	6	4	5
5	3	2	3
4	1	1	2
S/Q	3	4	3

2.3. ST-Filter

2.3.1. Whole matching. For the filtering step, the elements of data sequences in a database are transformed into symbols, and the symbol sequences are stored in a *suffix tree* [25]. By using the suffix tree traversal, the filtering step chooses candidate sequences S whose D_{tw} to a query sequence Q is likely to be smaller than a tolerance ε. For every S, the post-processing step computes D_{tw}(S, Q) using the dynamic programming technique.

2.3.2. Subsequence matching. For the filtering step, the elements of all the suffixes within every data sequence are converted into symbols, and these symbolized suffixes are stored in the suffix tree. Via the suffix tree traversal, the filtering step finds candidate subsequences S[*i*:*j*] whose D_{tw} to a query sequence Q is likely to be smaller than a tolerance ε. For every S[*i*:*j*], the post-processing step computes D_{tw}(S[*i*:*j*], Q) using the dynamic programming technique.

2.3.2. Characteristics. Owing to employing the suffix tree traversal in the filtering step, ST-Filter needs to access only a part of the suffix tree, rather than entire data sequences, from disk. However, the suffix tree is usually much larger than the whole volume of data sequences. Also, optimal *categorization* [18] cannot be easily obtained to provide good performance in sequence matching [17]. Furthermore, there is a big difference in the performances of subsequence matchings with ST-Filter on the same database, depending on query sequences even when their lengths are the same.

2.4. LB-Filter

2.4.1. Whole matching. For performance improvement, a multi-dimensional index is used. Feature vectors consisting of the first, last, greatest, and smallest elements are extracted from a query sequence and a data sequence, respectively. The feature vectors of data sequences are stored in a four-dimensional tree, and the candidate sequences S are filtered by traversing the tree with the feature vector of a query sequence. In the post-processing step, for every S, D_{tw}(S, Q) is computed by dynamic programming.

2.4.2. Subsequence matching. The subsequence matching with LB-Filter is called *prefix-querying* [19]. Prefix-querying extracts a number of windows of a fixed size from every data sequence, obtains a feature vector from each window as in whole matching, and then constructs a four-dimensional tree on all the feature vectors. For query processing, *prefixes* are extracted from a query sequence Q, and then a feature vector is obtained from each prefix. The candidate sequences S[*i*, *j*] are filtered by the tree traversal with every feature vector. In the post-processing step, for every S[*i*, *j*], D_{tw}(S[*i*, *j*], Q) is computed by dynamic programming.

2.4.3. Characteristics. By using the multi-dimensional index, LB-Filter performs the filtering step much faster than LB-Scan. Also, the size of the tree used in ST-Filter is larger than that of data sequences, but the prefix-querying method does not need a large space for the tree, because a feature vector employs only four element values. Furthermore, LB-Filter does not incur any false dismissals in subsequence matching.

3. Extension of the prefix-querying method

This section addresses an extension of the prefix-querying method [19] for the use of L_1 as a base distance function. In reference [17], it has been proven that LB-Filter does not incur false dismissals by showing that D_{tw_lb} with L_∞ is not only the lower-bound function of D_{tw} but also satisfies triangular inequality. In this paper, we extend the prefix-querying method to employ L_1 instead of L_∞ and also prove that this extension produces no false dismissals in subsequence matching.

Definition 2. *D_{tw_lb}(S, Q) = L_1(Feature(S), Feature(Q)), where Feature(S) = <First(S), Last(S), Greatest(S), Smallest(S)>, Feature(Q) = <First(Q), Last(Q), Greatest(Q), Smallest(Q)>.*

Next, based on Theorems 1 and 2, we prove that the function D_{tw_lb} is not only the lower-bound function of D_{tw} with L_1 but also satisfies triangle inequality. Assumption 1 is made for the processes of proving Theorems. For the case where the assumption does not hold, Theorem 5 suggests solutions.

Assumption 1. *If the time warped sequences of a data sequence S and a query sequence Q are denoted as S′ and Q′, respectively, a value of s_1', s_k', q_1', q_k' in S′ = <s_1', s_2', . . . , s_k'> and Q′ = <q_1', q_2', . . . , q_k'> is neither the maximum nor the minimum within the corresponding sequence.*

Lemma 1. *For two arbitrary sequences S = <s_1, s_2, . . . , s_n> and Q = <q_1, q_2, . . . , q_m>, the following is always true:*

$$D_{tw}(S, Q) \geq L_1(<First(S), Last(S)>, <First(Q), Last(Q)>)$$

Proof. Note that, in time warping, data sequences S and Q are transformed into the sequences to have the minimum L_1. We denote these transformed sequences as S′ and Q′, where |S′| = |Q′| = k(|S| ≤ k, |Q| ≤ k).

$$\begin{aligned} D_{tw}(S, Q) &= L_1(S', Q') \\ &= L_1(<s_1', s_2', \ldots, s_k'>, <q_1', q_2', \ldots, q_k'>) \\ &= L_1(<s_2', s_3', \ldots, s_{k-1}'>, <q_2, q_3', \ldots, q_{k-1}'>) + \\ &\quad L_1(<s_1', s_k'>, <q_1', q_k'>) \\ &= L_1(<s_2', s_3', \ldots, s_{k-1}'>, <q_2, q_3', \ldots, q_{k-1}'>) + \\ &\quad L_1(<\text{First(S), Last(S)}>, <\text{First(Q), Last(Q)}>) \\ &\geq L_1(<\text{First(S), Last(S)}>, <\text{First(Q), Last(Q)}>) \end{aligned}$$

Hence, Lemma 1 always holds.

Lemma 2. *For two arbitrary sequences S = <s_1, s_2, . . . , s_n> and Q = <q_1, q_2, . . . , q_m>, the following is always true:*

$$D_{tw}(S, Q) \geq L_1(<Greatest(S), Smallest(S)>, <Greatest(Q), Smallest(Q)>)$$

Proof. As in Lemma 1, the time warped sequences of S and Q are denoted S′ and Q′, respectively. Also, let

Greatest_Match(Q′) and Smallest_Match(Q′) be the elements of Q′ matched with Greatest(S′) and Smallest(S′) after transformation, respectively. The meanings of Greatest_Match(S′) and Smallest_Match(S′) are analogous.

In this proof, as shown in Figure 1, we are to prove that Lemma 2 holds for all the three cases which may occur according to the value ranges of the elements in the two sequences. For convenience, it is assumed to be Greatest(S′) ≥ Greatest(Q′). In the opposite case, i.e. Greatest(S′) ≤ Greatest(Q′), we can prove it in the same way by a simple role exchange.

3.1. Case 1: when S′ and Q′ are disjoint

$$\begin{aligned} D_{tw}(S, Q) &= L_1(S', Q') \\ &\geq |\text{Greatest}(S') - \text{Greatest\_Match}(Q')| + |\text{Smallest}(Q') - \text{Smallest\_Match}(S')| \\ &\geq |\text{Greatest}(S') - \text{Greatest}(Q')| + |\text{Smallest}(Q') - \text{Smallest}(S')| \\ &= L_1(<\text{Greatest}(S'), \text{Smallest}(S')>, <\text{Greatest}(Q'), \text{Smallest}(Q')>) \\ &= L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>) \end{aligned}$$

3.2. Case 2: when S′ encloses Q′

$$\begin{aligned} D_{tw}(S, Q) &= L_1(S', Q') \\ &\geq |\text{Greatest}(S') - \text{Greatest\_Match}(Q')| + |\text{Smallest}(S') - \text{Smallest\_Match}(Q')| \\ &\geq |\text{Greatest}(S') - \text{Greatest}(Q')| + |\text{Smallest}(S') - \text{Smallest}(Q')| \\ &= L_1(<\text{Greatest}(S'), \text{Smallest}(S')>, <\text{Greatest}(Q'), \text{Smallest}(Q')>) \\ &= L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>) \end{aligned}$$

3.3. Case 3: when S′ and Q′ overlap

$$\begin{aligned} D_{tw}(S, Q) &= L_1(S', Q') \\ &\geq |\text{Greatest}(S') - \text{Greatest\_Match}(Q')| + |\text{Smallest}(Q') - \text{Smallest\_Match}(S')| \\ &\geq |\text{Greatest}(S') - \text{Greatest}(Q')| + |\text{Smallest}(Q') - \text{Smallest}(S')| \\ &= L_1(<\text{Greatest}(S'), \text{Smallest}(S')>, <\text{Greatest}(Q'), \text{Smallest}(Q')>) \\ &= L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>) \end{aligned}$$

Since Lemma 2 is true for every case, the proof is completed.

Theorem 1. *For two arbitrary sequences* $S = <s_1, s_2, \ldots, s_n>$ *and* $Q = <q_1, q_2, \ldots, q_m>$*, the following is always true:*

$$D_{tw}(S, Q) \geq D_{tw\_lb}(S, Q)$$

Proof. Let S′ and Q′ be the time warped sequences of S and Q, respectively.

$$\begin{aligned} D_{tw\_lb}(S, Q) &= L_1(\text{Feature}(S), \text{Feature}(Q)) \\ &= L_1(<\text{First}(S), \text{Last}(S), \text{Greatest}(S), \text{Smallest}(S)>, <\text{First}(Q), \text{Last}(Q), \text{Greatest}(Q), \text{Smallest}(Q)>) \\ &= L_1(<\text{First}(S), \text{Last}(S)>, <\text{First}(Q), \text{Last}(Q)>) \\ &\quad + L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>) \\ D_{tw}(S, Q) &= L_1(S', Q') \\ &= L_1(<s'_1, s'_2, \ldots, s'_k>, <q'_1, q'_2, \ldots, q'_k>) \\ &= L_1(<\text{First}(S), \text{Last}(S)>, <\text{First}(Q), \text{Last}(Q)>) \\ &\quad + L_1(<s'_2, s'_3, \ldots, s'_{k-1}>, <q'_2, q'_3, \ldots, q'_{k-1}>) \end{aligned}$$

By Assumption 1, the values of s'_1, s'_k, q'_1, q'_k are neither the maximum nor the minimum in S′ and Q′. Thus, by Lemma 2, it holds that

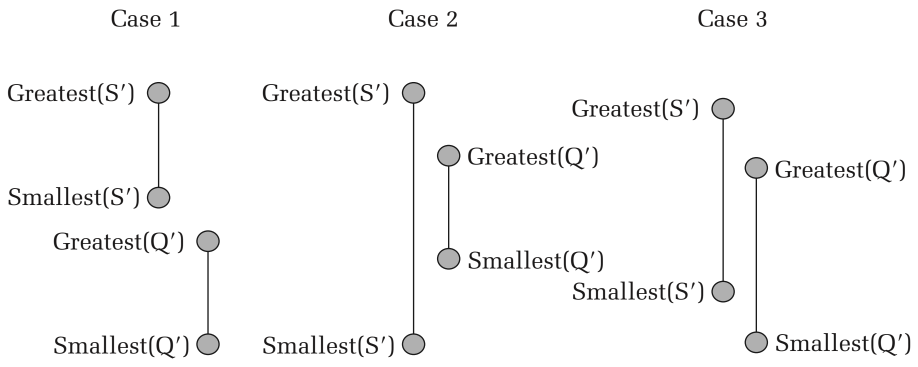

Fig. 1. Three possible arrangements of ranges of S′ and Q′.

$$L_1(<s'_2, s'_3, \ldots, s'_{k-1}>, <q'_2, q'_3, \ldots, q'_{k-1}>) \geq$$
$$L_1(<\text{Greatest(S); Smallest(S)}>, <\text{Greatest(Q); Smallest(Q)}>). \text{ Therefore, } D_{tw}(S, Q) \geq D_{tw\_lb}(S, Q)$$

Using Theorem 1, the following Corollary 1 can easily be induced.

Corollary 1. *For two arbitrary sequences S = <s_1, s_2, . . . , s_n> and Q = <q_1, q_2, . . . , q_m>, and for an arbitrary value ε, the following is always true.*

$$D_{tw}(S, Q) \leq \varepsilon \Rightarrow D_{tw\_lb}(S, Q) \leq \varepsilon$$

Theorem 2. *For three arbitrary sequences X, Y, and Z, the following always holds.*

$$D_{tw\_lb}(X, Z) \leq D_{tw\_lb}(X, Y) + D_{tw\_lb}(Y, Z)$$

Proof. D_{tw_lb}(S, Q) = L_1(Feature(S), Feature(Q)). Also, any L_1 satisfies triangular inequality. Hence, Theorem 2 always holds.

Next, it is shown that the prefix-querying method does not incur false dismissals in the case of applying L_1 to D_{tw_lb}.

Theorem 3. *For two arbitrary sequences s and q, and for an arbitrary positive value w(1 ≤ w ≤ |s|), if the time warping distance between s and q is less than ε, there exists a prefix of q whose time warping distance to a prefix of s, s[1:w], is less than ε. That is, the following holds in all cases.*

$$D_{tw}(s, q) \leq \varepsilon \Rightarrow (\exists x)(D_{tw}(s[1{:}w], q[1{:}x]) \leq \varepsilon)$$

Proof. Let p = <p[1], p[2], . . . , p[|p|]> be the warping path that minimizes D_{tw} between s and q, where |s| ≤ |p| and |q| ≤ |p|. Also, let $p[h]$ = ($s[i_h]$, q[j_h]) be an element of the warping path, and let s' (=($s[i_h]$)), q' (=($q[j_h]$)) be the time warped sequences of s and q, respectively, which exist on the warping path, where 1 ≤ h ≤ |p|, 1 ≤ i_h ≤ |s|, and 1 ≤ j_h ≤ |q|. Here, D_{tw} for s and q is calculated as follows.

$$D_{tw}(\text{s, q}) = L_1(s', q')$$

Also, according to the *monotonic and continual properties* of the warping path, the following always holds.

$$(\exists x)\ (\text{p}[x] = (\text{s}[w], \text{q}[j_x])), \text{ where } 1 \leq w \leq |s|,$$
$$1 \leq x \leq |p|, 1 \leq j_x \leq |q|$$

D_{tw} with L_∞ takes the maximum difference between any pair of elements existing on the warping path. On the other hand, D_{tw} with L_1 takes the sum of the differences between all the pairs of two elements. Accordingly, the sub-warping path from $p[1]$ to $p[x]$ returns a time warping distance which is always less than that of p, the whole warping path. Thus, if the warping path p returns a distance less than ε, then the sub-warping path from $p[1]$ to $p[x]$ also returns a distance less than ε. Since $D_{tw}(s[1{:}w], q[1{:}j_x])$ denotes the time warping distance for the sub-warping path from $p[1]$ to $p[x]$, Theorem 3 holds.

Since D_{tw_lb} used by LB-Filter is the lower-bound function of the time warping distance D_{tw}, Corollary 2 can be easily induced from Theorem 3.

Corollary 2. *For two arbitrary sequences s and q, and for an arbitrary positive value w(1 ≤ w ≤ |s|), the following holds in all cases.*

$$D_{tw}(s, q) \leq \varepsilon \Rightarrow (\exists \text{x})(D_{tw\_lb}(s[1{:}\text{w}], q[1{:}\text{x}]) \leq \varepsilon),$$
$$\text{where } 1 \leq \text{x} \leq |q|$$

3.3.1. Multi-role elements in feature vectors. Until now, we have assumed that the values of s'_1, s'_k, q'_1, q'_k are neither the maximum nor the minimum in the time warped sequences S′ = <s'_1, s'_2, . . . , s'_k> and Q′ = <q'_1, q'_2, . . . , q'_k>. In practice, however, they could be the maximum or the minimum, although it is not that frequent. Here, we investigate this issue and propose the solution to it.

If the values of s'_1, s'_k are the maximum or the minimum in a time warped sequence S′ = <s'_1, s'_2, . . . , s'_k>, it implies that, in Feature(S) = <First(S), Last(S), Greatest(S), Smallest(S)>, First(S) = Greatest(S) or Smallest(S), or Last(S) = Greatest(S) or Smallest(S). It is analogous in a query sequence. In this situation, we take two different cases into consideration as follows.

3.4. Case 1: One of $s'_2, s'_3, \ldots, s'_{k-1}$ is equal to s'_1 (or s'_k)

This is the case that the first value s'_1 (or the last value s'_k) of a sequence is the maximum or the minimum but one of s'_2, s'_3, . . . , s'_{k-1} is equal to s'_1 (or s'_k). In such cases, although s'_1 (or s'_k) is the maximum or the minimum, since there exist other elements which have the same value, the feature vector is to contain four distinct elements. Thus, the problem situation does not occur since the following formula used in the proof of Theorem 1 is satisfied.

$$L_1(<s'_2, s'_3, \ldots, s'_{k-1}>, <q'_2, q'_3, \ldots, q'_{k-1}>) \geq$$
$$L_1(<\text{Greatest(S), Smallest(S)}>, <\text{Greatest(Q), Smallest(Q)}>)$$

3.5. Case 2: None of $s'_2, s'_3, \ldots, s'_{k-1}$ is equal to s'_1 (or s'_k)

This is the case that the first value s'_1 (or the last value s'_k) of a sequence is the maximum or the minimum, and does not appear in the remaining part of the time warped sequence. In such cases, if a feature vector is constructed from a sequence, an identical element is represented as two features. We call this a *multi-role element*. If there exist multi-role elements in a feature vector, two features in a vector are extracted from the same element in the time warped sequence. In this case, Greatest(S) or Smallest(S) does not appear in $<s'_2, s'_3, \ldots, s'_{k-1}>$. Thus, Theorem 1 does not hold in this case.

Example 1. It happens that $D_{tw} < D_{tw\_lb}$ due to a multi-role element in a feature vector.

S = <100, 20, 15, 5, 30>, Q = <15, 20, 15, 5, 30>

Feature(S) = <100, 30, 100, 5>, Feature(Q) = <15, 30, 30, 5>

S′ = <100, 20, 15, 5, 30>, Q′ = <15, 20, 15, 5, 30>

D_{tw}(S, Q) = 85, D_{tw_lb}(S, Q) = 155

Thus

D_{tw}(S, Q) < D_{tw_lb}(S, Q)

Theorem 4. *To avoid prefix-querying with L_1 incurring any false dismissals, no two elements of Feature(S) (=<First(S), Last(S), Greatest(S), Smallest(S)>) should be extracted from the same element in the time warped sequence* $S' = <s'_1, s'_2, s'_3, \ldots, s'_{k-1}, s'_k>$.

Proof. If no two elements of Feature(S) are extracted from the same element in S′, Smallest(S) can be extracted from $s'_2, s'_3, \ldots, s'_{k-1}$. If the same condition is also satisfied for a query sequence Q, the feature vectors for S and Q satisfy the conditions used in the proof of Theorem 1. Therefore, the proof of Theorem 4 is completed.

In this paper, we propose a solution to the case in <Example 1>. This solution is to guarantee no false dismissals by performing index searching in the range of an enlarged ε.

Theorem 5. *If either of a data sequence S and a query sequence Q has a multi-role element in its feature vector, index searching with 2ε makes prefix-querying with L_1 guarantee no false dismissals.*

Proof. By Lemma 1 and Lemma 2, for two arbitrary sequences $S = <s_1, s_2, \ldots, s_n>$ and $Q = <q_1, q_2, \ldots, q_m>$, the following are always true.

$$D_{tw}(S, Q) \geq L_1(<\text{First}(S), \text{Last}(S)>, <\text{First}(Q), \text{Last}(Q)>)$$

$$D_{tw}(S, Q) \geq L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>)$$

As shown in Theorem 1, D_{tw_lb}(S, Q) can be defined as follows.

$$\begin{aligned} D_{tw\_lb}(S, Q) &= L_1(\text{Feature}(S), \text{Feature}(Q)) \\ &= L_1(<\text{First}(S), \text{Last}(S)>, <\text{First}(Q), \text{Last}(Q)>) + L_1(<\text{Greatest}(S), \text{Smallest}(S)>, <\text{Greatest}(Q), \text{Smallest}(Q)>) \end{aligned}$$

According to Lemma 1, Lemma 2, and Theorem 1, regardless of appearances of multi-role elements in feature vectors, the following formula holds in all cases.

$$2D_{tw}(S, Q) \geq D_{tw\_lb}(S, Q)$$

Hence, by this formula and Corollary 2, we complete the proof of Theorem 5.

Prefix-querying with L_1 does not incur false dismissals even for the feature vectors having multi-role elements. However, the number of false alarms increases due to an enlarged ε. To resolve this problem, we maintain two different trees T_1 and T_2 in indexing feature vectors. T_1 contains feature vectors that do not contain multi-role elements, T_2 contains feature vectors that may contain multi-role elements. However, we cannot recognize whether multi-role elements exist in a sequence exactly because we do not have the time warped sequences at the time of index construction. So, we build T_2 on sequences whose first (or last) value is the maximum or minimum one. The same idea is applied to the case of a query sequence. If a query sequence does not have any multi-role elements in its feature vector, index searches are performed on T_1 with ε and T_2 with 2ε. On the other hand, if a query sequence has multi-role elements in its feature vector, index searches are performed on both T_1 and T_2 with 2ε. In the latter case, the number of false alarms increases, thus the processing time gets large. It is noted, however, that multi-role elements are infrequent when we consider that sequences are long and have a variety of values.

4. Performance evaluation

This section evaluates the performance of the prefix-querying method in comparison with Naive-Scan, LB-Scan, and ST-Filter. In Section 4.1, the experiment environment for performance evaluation is described, and in Section 4.2, experimental results are presented.

4.1. Experiment environment

For performance evaluation, we used two data sets: K-Stock-Data and Syn-Data. K-Stock-Data is a real-world

Korean stock data set that consists of 620 data sequences of length 300. Syn-Data is a synthetic data set that contains sequence S = $<s_1, s_2, \ldots, s_n>$ generated by the following random walk expression.

$$s_i = s_{i-1} + z_i$$

Here, z_i is a random variable that takes a value within the range of [–0.1, 0.1], and s_1, the first element of the sequence, takes a random value within the range of [1, 10]. For a scalability test, we generated four sets of Syn-Data that include 1000, 2000, 3000, and 4000 data sequences of length 200, respectively, and another four sets of Syn-Data that include 1000 data sequences of lengths 200, 300, 400, and 500, respectively.

Also, we generated query sequences Q by taking an arbitrary subsequence of length Len(Q) among the sequences chosen from a database. To form a query, we adjusted ε to make the query meet the specified query selectivity defined as below.

$$\text{query selectivity} = \left(\frac{\text{number of subsequences in the final set}}{\begin{array}{c}\text{number of subsequences whose length is appropriate}\\ \text{to be matched to Q of Len(Q) in a database}\end{array}} \right)$$

For performance evaluation, a PC with 1.7GHz Pentium IV CPU and 1.2GB RAM was used as a hardware platform, and the Linux kernel of version 2.4.18 and Glibc 2.2.4 were used as a software platform. To prevent interference with other processes during experiments, the operating system was set to a single-user mode. Also, for categorization in ST-Filter, we used the *maximum entropy method*, resulting in a domain of 50 intervals.

4.2. Results and analyses

In Experiment 1, we used the selectivities of 7.10×10^{-8}, 2.13×10^{-7}, 3.55×10^{-7} and 4.97×10^{-7} and compared the performance of the prefix-querying method with the existing methods. The length of the query sequences used was 110, and 2, 6, 10, and 14 sequences were returned as final results depending on query selectivities. Figure 2 shows the results.

With the increase of selectivities, the total elapsed time increases in all the methods. In particular, ST-Filter shows a great increasing rate. This is because, with L_1, the amount of CPU operations required to construct the cumulative distance table in searching the suffix-tree for the candidates is greatly affected by ε. LB-Scan does not show good performance since it needs to access all data sequences in searching for the candidate sequences.

Note that Naive-Scan is always better than LB-Scan and ST-Filter in all cases. This is because the optimized version of Naive-Scan [23] was used in our experiments for CPU processing. This Naive-Scan is able to maximize CPU performance by removing most redundant calculations required in computing the time warping distance of a query sequence against data subsequences. As a result, Naive-Scan resolves performance bottlenecks incurred in CPU processing, showing a high performance as in our experiments. Such a performance inversion is shown consistently in all our

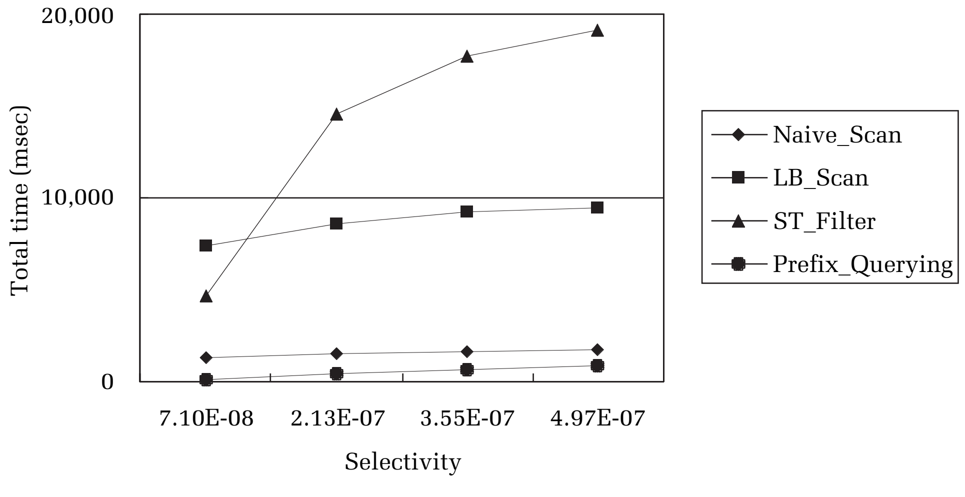

Fig. 2. Performance results with different selectivities.

experiments, which coincides with the results of [23]. The prefix-querying method shows a superior performance over all other methods, and it shows up to 10.7 times better performance than Naive-Scan, the best of existing methods. This is because the filtering step is greatly improved by using the index in prefix-querying.

The term of *maxWarpRatio* represents how many times each element of a sequence can replicate itself by time warping [15]. In Experiment 2, for *maxWarpRatio* 3, 6, 9, and 12, we compared the performance of the prefix-querying method with existing ones. Here, the length of query sequences was 110 while the selectivity was 2.13×10^{-7}. Figure 3 shows the results.

As *maxWarpRatio* increases, the number of candidate subsequences returned from the filtering step grows in all the methods. Thus, we can see that the total elapsed time is gradually increasing. In the case of the prefix-querying method, the minimum length of query sequences(*minQLen*) [19] divided by *maxWarpRatio* was used as a window size, i.e. $w = \lceil minQLen/maxWarpRatio \rceil$. Accordingly, the increase of *maxWarpRatio* brings about the growth in the number of candidate subsequences as well as the number of the prefixes, thereby increasing the post-processing time. In this experiment, the prefix-querying method shows at best a 4.4 times better performance than Naive-Scan, the best one among existing methods.

In Experiment 3, we evaluated the performance of the prefix-querying method against existing ones with different *minQLen* of 10, 40, 70, and 100. The length of query sequences was 110 and the selectivity was 2.13×10^{-7}. Figure 4 shows these results.

ST-Filter shows a smaller total elapsed time with a larger *minQLen*. The increase of *minQLen* leads to the reduction of the suffixes in the suffix tree, consequently reducing the computation time in the filtering step. Other methods show almost the same performance even with different values of *minQLen*. The prefix-querying method shows an up to 3.8 times better performance than Naive-Scan.

In Experiment 4, we evaluated the performance of the prefix-querying method against existing methods with changing query sequence lengths of 80, 140, 200, and 260. The query selectivity used is 2.13×10^{-7}. Figure 5 shows the experiment results.

As the length of query sequences increases, the cost for computing the distances goes up, which leads to the increase of the time spent in the filtering step and the post-processing step. With increasing lengths of query sequences, ST-Filter and LB-Scan show drastic growth in elapsed time while Naive-Scan and the prefix-querying method observe smooth growth. In particular, the prefix-querying method shows the best performance, and achieves up to 3.6 times speed-up over Naive-Scan.

The stock data set used in the previous experiments is relatively small. For scalability tests, we generated synthetic data sets of various numbers of sequences with varying lengths for the following experiments. In Experiment 5, while fixing the length of data sequences at 200, we evaluated the performance of the prefix-querying method against existing methods while

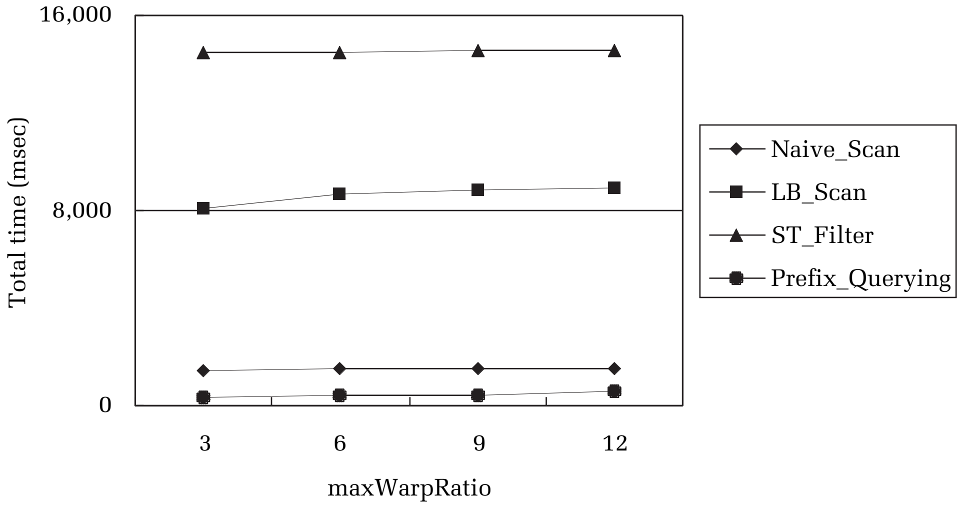

Fig. 3. Performance values of results with different *maxWarpRatio*.

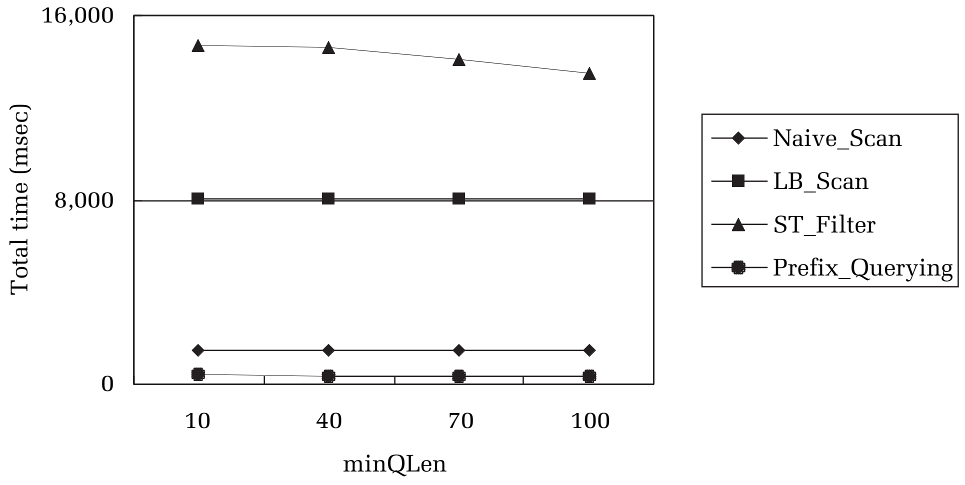

Fig. 4. Performance results with different values of *minQLen*.

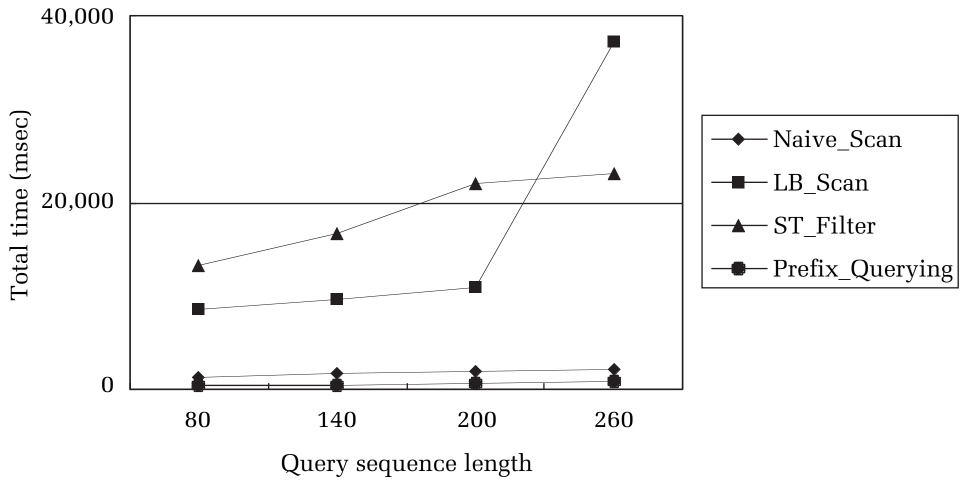

Fig. 5. Performance results with different lengths of query sequences.

changing the number of data sequences to 1000, 2000, 3000, and 4000. The length of query sequences used was 60, and the query selectivity was 1.47×10^{-7}. Figure 6 shows the results.

As the number of data sequences increases, we see that the total elapsed time increases linearly in all the methods. ST-Filter and LB-Scan show a drastic degradation in performance while Naive-Scan and the prefix-querying method show a relatively smooth degradation in performance. In this experiment, the prefix-querying method shows up to 180 times better performance than the best among existing ones.

In Experiment 6, while fixing the number of data sequences at 1000, we ran experiments with different lengths of data sequences of 200, 300, 400, and 500. The query selectivity used was 1.47×10^{-7}. Figure 7 shows the results.

As the length of data sequences increases, the total elapsed time increases in all the methods. The overall performance tendency of ST-Filter and LB-Scan is very similar to that in Experiment 5. LB-Scan shows exactly the same results as in Experiment 5 even for the case with long data sequences because it needs to access all the elements of sequences in the filtering step. In this experiment, the prefix-querying method shows the best performance, and reveals performance up to 180 times better than Naive-Scan.

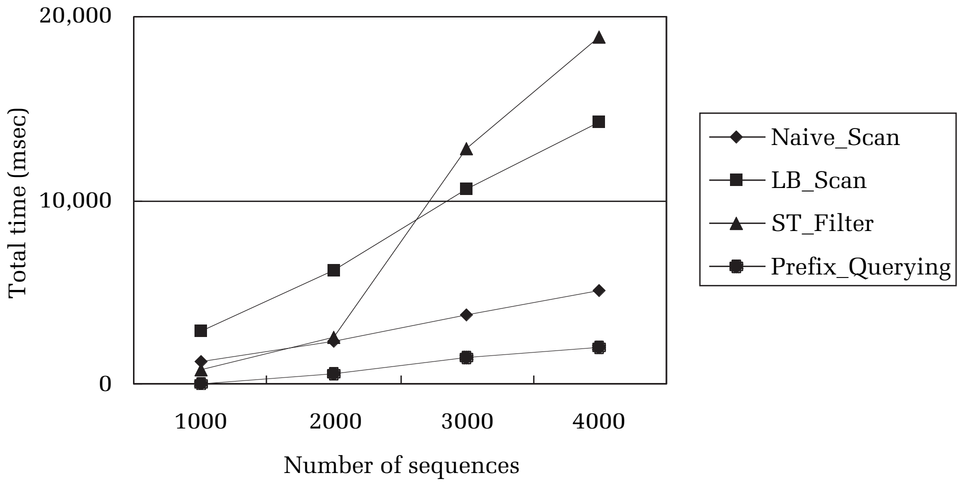

Fig. 6. Performance results with different numbers of data sequences.

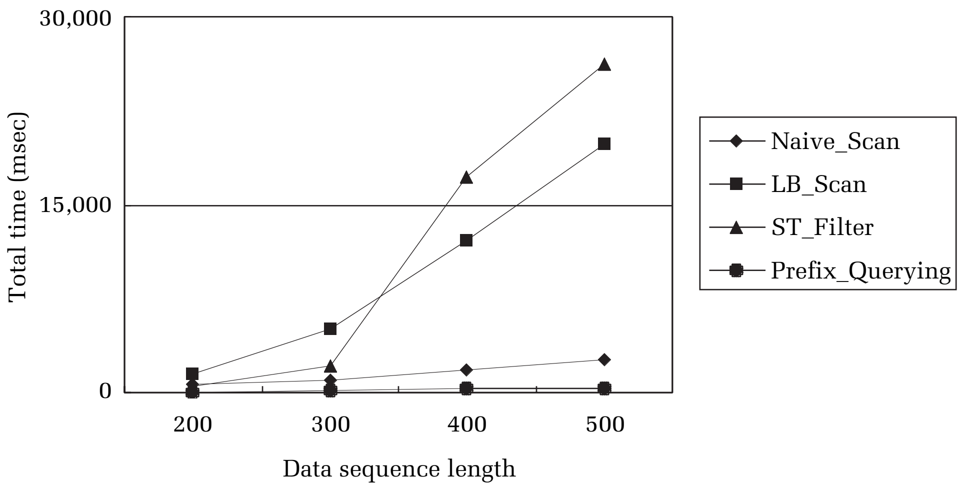

Fig. 7. Performance results with different lengths of data sequences.

5. Conclusions

Prefix-querying is the first index-based method to perform time-series subsequence matching under time warping without false dismissals. It employs L_∞ as a base distance function for users' convenience in forming queries. In this paper, we have addressed an extension of the prefix-querying method to the use of L_1, which is most popular for time-series subsequence matching under time warping, instead of L_∞ as a base distance function. The contributions of this paper are summarized as follows.

- We have extended the prefix-querying method to employ L_1 instead of L_∞ as a base distance function.
- We have formally proven that the prefix-querying method guarantees no false dismissals in subsequence matching under time warping.
- We have empirically shown the superiority of the prefix-querying method via extensive experiments. The results reveal that our method achieves significant performance improvement over the previous methods, by up to 10.7 times with a data set containing real-world Korean stock data sequences, and up to 180 times with data sets containing a very large volume of synthetic data sequences. According to the experiment results, the prefix-querying method improves the prior methods dramatically.

In the case of large databases, even with the prefix-querying method, the performance is not quite satisfactory due to a large number of candidate subsequences returned after the filtering step. Thus, we need more sophisticated techniques to reduce the number of candidate subsequences in the filtering step. As a further study, we are considering devising a lower-bound function that fits the time warping distance more tightly.

Acknowledgement

This research was partially supported by the MIC (Ministry of Information and Communication) of Korea under the ITRCC Information Technology Research Centre support program supervised by the IITA (IITA-2005-C1090-0502-0009).

Notes

(1) Sequences of different lengths need to be compared in the following situations [13]: (a) when sequences have different sampling rates, for example, one sequence may be sampled every minute while another sequence is sampled every hour; and (b) when sequences have different starting points; for example, a sequence may start today while another sequence began a year ago. The time warping distance is very useful in these situations. Instead of focusing on individual elements of sequences, the time warping distance compares their fluctuation patterns along the time axis.

(2) Whereas ST-Filter was originally designed for subsequence matching, Naive-Scan and LB-Scan were for whole matching. In this section, we simply extend them for subsequence matching, retaining their basic ideas.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami., Efficient similarity search in sequence databases. In: D.B. Lomet (ed.), *Proceedings of the International Conference on Foundations of Data Organization and Algorithms (FODO) Chicago, October 1993* (Springer, London, 1993) 69–84.

[2] R. Agrawal et al., Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In: U. Dayal et al. (eds) , *Proceedings of the International Conference on Very Large Data Bases (VLDB), Zurich, September 1995* (Morgan Kaufmann, San Francisco, 1995) 490–501.

[3] C. Chatfield, *The Analysis of Time-Series: an Introduction* (Chapman and Hall, New York, 1984).

[4] C. Faloutsos et al., Fast subsequence matching in time-series databases. In: R.T. Snodgrass and M. Winslett (eds), *Proceedings of the International Conference on Management of Data, ACM SIGMOD, Minneapolis, Minnesota, May 1994* (ACM Press, New York, 1994) 419–29.

[5] M.S. Chen et al., Data mining: an overview from database perspective, *IEEE Transactions on Knowledge and Data Engineering* 8(6) (1996) 866–83.

[6] D. Rafiei and A. Mendelzon, Similarity-based queries for time-series data. In: *Proceedings of the International Conference on Management of Data, ACM SIGMOD, Tucson, Arizona, June 1997* (ACM Press, New York, 1997) 13–24.

[7] B.K. Yi and C. Faloutsos, Fast time sequence indexing for arbitrary *Lp* norms. In: A. El Abbadi et al. (eds), *Proceedings of the International Conference on Very Large Data Bases, VLDB 2000* (Morgan Kaufmann, San Francisco, 2000) 385–94.

[8] K.P. Chan and A.W.C. Fu, Efficient time-series matching by wavelets. In: *Proceedings of the International Conference on Data Engineering (ICDE), Sydney, March 1999* (IEEE, Washington, DC, 1999) 126–33.

[9] K.K.W. Chu and M.H. Wong, Fast time-series searching with scaling and shifting. In: V. Vianu and C. Papadimitriou (eds), *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Philadelphia, Pennsylvania, May 1999* (ACM Press, New York, 1999) 237–48.

[10] D.Q. Goldin and P.C. Kanellakis, On similarity queries for time-series data: constraint specification and implementation. In: U. Montanari and F. Rossi (eds), *Proceedings of the International Conference on Principles and Practice of Constraint Programming, Cassis, France, September 1995* (Springer, London, 1997) 137–53.

[11] D. Rafiei, On similarity-based queries for time-series data. In: *Proceedings of the International Conference on Data Engineering (ICDE), Sydney, Australia, March 1999* (IEEE, Washington, DC, 1999) 410–17.

[12] G. Das, D. Gunopulos and H. Mannila, Finding similar time-series. In: H.J. Komorowski and J. Zytkow (eds), *Proceedings of the European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD) 1997* (Springer, London, 1997) 88–100.

[13] W.K. Loh, S.W. Kim and K.Y. Whang, Index interpolation: an approach for subsequence matching supporting normalization transform in time-series databases. In: A. Agah et al. (eds), *Proceedings of the ACM International Conference on Information and Knowledge Management, ACM CIKM 2000* (ACM Press, New York, 2000) 480–7.

[14] W.K. Loh, S.W. Kim and K.Y. Whang, Index interpolation: a subsequence matching algorithm supporting moving average transform of arbitrary order in time-series databases, *IEICE Transactions on Information and Systems*, E84-D 1 (January, 2001) 76–86.

[15] D.J. Berndt and J. Clifford, Finding patterns in time-series: a dynamic programming approach. In: U.M. Fayyad et al. (eds), *Advances in Knowledge Discovery and Data Mining* (AAAI/MIT Press, Menlo Park, 1996) 229–48.
[16] E. Keogh, Exact indexing of dynamic time warping. In: *Proceedings of the International Conference on Very Large Data Bases (VLDB) 2002* (Morgan Kaufmann, San Francisco, 2002) 406–17.
[17] S.W. Kim, S.H. Park and W.W. Chu, An index-based approach for similarity search supporting time warping in large sequence databases. In: *Proceedings of the International Conference on Data Engineering, IEEE ICDE 2001* (IEEE, Washington, DC, 2001) 1607–14.
[18] S.H. Park et al., Efficient searches for similar subsequences of difference lengths in sequence databases. In: *Proceedings of the International Conference on Data Engineering, IEEE ICDE 2000* (IEEE, Washington, DC, 2000) 23–32.
[19] S.H. Park, S.W. Kim, J.S. Cho and S. Padmanabhan, Prefix-querying: an approach for effective subsequence matching under time warping in sequence databases. In: H. Paques et al. (eds), *Proceedings of the ACM International Conference on Information and Knowledge Management, ACM CIKM 2001* (ACM Press, New York, 2001) 255–62.
[20] B.K. Yi, H.V. Jagadish and C. Faloutsos, Efficient retrieval of similar time sequences under time warping. In: *Proceedings of the International Conference on Data Engineering, IEEE ICDE 1998* (IEEE, Washington, DC, 1998) 201–8.
[21] S.W. Kim, S.H. Park and W.W. Chu, Efficient processing of similarity search under time-warping in sequence databases: an index-based approach, *Information Systems* 29(5) (2004) 405–20.
[22] L. Rabiner and H.H. Juang, *Fundamentals of Speech Recognition* (Prentice Hall, Englewood Cliffs, 1993).
[23] S.W. Kim, S.W. Kim and M.Y. Shin, Optimization of subsequence matching under time warping in time-series databases. In: *ACM Symposium on Applied Computing, April 2005* (ACM Press, New York, 2005) 581–6.
[24] C. Faloutsos, *Private Communication* (2001).
[25] G.A. Stephen, *String Searching Algorithms* (World Scientific Publishing, Singapore, 1994).