A Single Index Approach for Distortion-Free Time-Series Subsequence Matching

Yang-Sae Moon†, Woong-Kee Loh‡, and Jinho Kim†

†Department of Computer Science, Kangwon National University
192-1, Hyoja2-Dong, Chunchun, Kangwon 200-701, Korea

‡Department of Computer Science & Engineering, University of Minnesota
4-192, EE/CS Building, 200 Union Street SE, Minneapolis, MN 55455

ysmoon@kangwon.ac.kr, lohw@cs.umn.edu, jhkim@kangwon.ac.kr

Abstract

In this paper we propose a new method for *distortion-free* time-series subsequence matching. Our method is distortion-free in the sense that it performs preprocessing on time-series to remove the distortions of offset translation and amplitude scaling at the same time. We call this preprocessing as *normalization transform* in this paper. Previous work on the *normalization-transformed subsequence matching* has a problem of incurring index overhead since it should build multiple indexes for supporting arbitrary query lengths. To overcome this index overhead, we adopt a single index approach into the normalization-transformed subsequence matching. For the single index approach, we first provide the new notion of *inclusion-normalization transform* by generalizing the original definition of normalization transform. We then formally prove correctness of the proposed normalization-transformed subsequence matching method that exploits the inclusion-normalization. We also describe subsequence matching and index building algorithms to implement the proposed method. Experimental results for real stock data show that our method improves performance by up to 2.5-2.8 times over the previous method. We believe that our single index approach for distortion-free subsequence matching can be widely used in many applications for finding 'real' similar time-series.

1 Introduction

Typical examples of time-series data include stock prices, biomedical measurements, network traffic data, and financial data [1, 4, 5, 6, 14]. The time-series data stored in a database are called *data sequences*, and those given by users are called *query sequences*. And, finding data sequences similar to the given query sequence from the database is called *similar sequence matching* [1, 4, 9, 15]. Two sequences $X = \{X[1], X[2], ..., X[n]\}$ and $Y = \{Y[1], Y[2], ..., Y[n]\}$ are said to be *similar* if the distance $D(X, Y)$ is less than or equal to the user-specified *tolerance* ϵ [1, 4, 9, 11]. In this paper, we use the Euclidean distance ($= \sqrt{\sum_{i=1}^{n}(X[i] - Y[i])^2}$)[1], which has been widely used in [1, 3, 4, 7, 8, 9, 10], as the distance function $D(X, Y)$, and define that X and Y are in ϵ-*match* if $D(X, Y)$ is less than or equal to ϵ.

In this paper we propose a new method for *distortion-free* time-series subsequence matching. Our method is distortion-free in the sense that it performs preprocessing [6, 8, 13, 15] on time-series to remove the distortions of offset translation and amplitude scaling at the same time. We call this preprocessing as *normalization transform* [8, 13] in this paper. Thus, in other words, in this paper we focus on the subsequence matching that supports normalization transform. Here, the *subsequence matching*[4, 7, 9] is the problem of finding subsequences similar to a query sequence of arbitrary length. Normalization transform, which is known to be very useful for finding the overall trend of time-series data [8], converts a given sequence into a new sequence by using the *mean* and the *standard deviation* of the sequence. The normalization transform enables finding sequences with similar fluctuation patterns even though they are not close to each other before the normalization transform [8]. Including the definition of normalization transform, we summarize the notation to be used throughout the paper in Table 1.

The *normalization-transformed subsequence matching* is defined as a similarity model that uses the distance between two normalization-transformed sequences $\overline{Q}$ and $\overline{S[i:j]}$ to determine whether two sequences are in ϵ-match or not [8]. That is, given a query sequence Q and tolerance ϵ, the normalization-transformed subsequence matching is defined as the problem of finding $\overline{S[i:j]}$ that satisfies $D(\overline{Q}, \overline{S[i:j]}) \leq \epsilon$. Loh et al. [8] have proposed a novel solution for the normalization-transformed subsequence matching. They have introduced the notion of index interpolation that constructs multiple indexes for mul-

[1] In addition to the Euclidean distance, any L_p distance ($= \sqrt[p]{\sum_{i=1}^{n}|X[i] - Y[i]|^p}$) including the Manhattan distance ($= L_1$) and the maximum distance ($= L_\infty$) can be also used as a similarity measure [7].

Table 1. Summary of notation.

Symbols	Definitions
$S[i:j]$	Subsequence of S, including entries from the i-th one to the j-th
$\mu(S), \sigma(S)$	Mean and standard deviation of sequence S
$\overline{S}$	Normalization transformed sequence of S $\left(\overline{S}[i] = \frac{S[i]-\mu(S)}{\sigma(S)}\right)$
$\overline{S}[i:j]$	Subsequence of $\overline{S}$, including entries from the i-th one to the j-th
$\overline{S[i:j]}$	Normalization transformed sequence of subsequence $S[i:j]$ (by using $\mu(S[i:j])$ and $\sigma(S[i:j])$)
s_i	The i-th disjoint window of sequence S $(= S[(i-1)*\omega+1 : i*\omega])$
$\overline{s_i}$	The i-th disjoint window of $\overline{S}$ $(= \overline{S}[(i-1)*\omega+1 : i*\omega])$
ω	Length of the *sliding/disjoint windows*[4, 9]

tiple window sizes. Their solution, however, causes a serious problem that, as the number of indexes increases, index maintenance overhead would also increase to maintain multiple indexes.

To overcome this index overhead, in this paper we adopt a single index approach [11] into the normalization-transformed subsequence matching. To explain our single approach, we first provide the new notion of *inclusion-normalization transform* that normalizes a window $S[a:b]$ against $S[i:j]$ $(a \le i < j \le b)$. That is, to normalize a window $S[a:b]$, the inclusion-normalization transform uses $S[i:j]$ that includes the window while the original normalization transform uses $S[a:b]$ itself. We then formally show that, if constructing only one index using the inclusion-normalization transform, we can perform the normalization-transformed subsequence matching correctly. That is, we build only one index as the following steps: 1) data sequences are divided into fixed-size windows; 2) by using inclusion-normalization transform, each window is converted into multiple transformed windows for every possible subsequence length; and 3) each transformed window is mapped into a lower-dimensional point in a multidimensional index. After building the index, we can perform the normalization-transformed subsequence matching by searching the index.

By applying the inclusion-normalization transform to Faloutsos et al.'s subsequence matching method (we simply call it *FRM* by taking authors' initials.) [4, 9], we propose a new normalization-transformed subsequence matching method. That is, we propose the FRM-based normalization-transformed subsequence matching method by using the notion of inclusion-normalization transform. Also, to guarantee correctness of our method, we present a theorem and formally prove it. We then present index building and subsequence matching algorithms to implement the FRM-based normalization-transformed subsequence matching method. Through experiments, we also show that our method improves performance compared with the previous method by Loh et al. [8].

The rest of this paper is organized as follows. Section 2 describes related work and explains motivation of the research. Section 3 proposes a single index approach for the normalization transformed subsequence matching. Section 4 presents the results of performance evaluation. Section 5 summarizes and concludes the paper.

2 Related Work

We first review Agrawal et al.'s whole matching solution [1] that many other subsequence matching solutions have evolved from. In the index building algorithm, each data sequences of length n is transformed into f-dimensional points ($f \ll n$, we call it *lower-dimensional transformation*), and the transformed points are stored into an R*-tree [2]. Here, the function used for lower-dimensional transformation is called the *feature extraction function* [3, 4, 9, 10]. In the similar sequence matching algorithm, a query sequence is similarly transformed to an f-dimensional point, and a range query is constructed using the point and the tolerance ϵ. Then, by evaluating the range query using the index, the *candidates* that are potentially in ϵ-match with the query sequence are identified. This method guarantees there be no *false dismissal* (i.e., it does not miss a sequence in the result set), but may cause *false alarms* (i.e., candidates that do not qualify) because it uses only f features instead of n. Thus, for each candidate sequence obtained, the actual data sequence is accessed from the disk; the distance from the query sequence is computed; and the candidate is discarded if it is a false alarm. This last step, which eliminates false alarms, is called the *post-processing step* [1, 4, 9, 10].

Faloutsos et al.[4] have proposed a subsequence matching method as a generalization of the whole matching. FRM uses the window construction method of dividing data sequences into sliding windows and a query sequence into disjoint windows. In the index building algorithm, FRM transforms each data window to an f-dimensional point and stores the point into the R*-tree. However, dividing data sequences into sliding windows causes a serious problem of generating too many points stored into the index [4, 9]. To

solve this problem, FRM does not store individual points directly into the R*-tree, but stores only MBRs (minimum bounding rectangles) that contains hundreds or thousands of the f-dimensional points. In the subsequence matching algorithm, FRM performs the matching correctly based on the following Eq. (1). In Eq. (1), $p = \lfloor Len(Q)/\omega \rfloor$.

$$D(S,Q) \le \epsilon \implies \bigvee_{i=1}^{p} D(s_i, q_i) \le \epsilon/\sqrt{p} \qquad (1)$$

According to Eq. (1), FRM divides a query sequence into disjoint windows; transforms each window to an f-dimensional point; makes a range query using the point and the tolerance $\epsilon/\sqrt{p}$; and constructs a candidate set by searching the R*-tree. Finally, it performs the post-processing step to eliminate false alarms.

DualMatch [9, 7] and GeneralMatch [10] improve performance in range subsequence matching by using different window construction methods from FRM. By introducing the notion of *duality* in constructing windows, DualMatch performs subsequence matching by dividing the data sequences into disjoint windows and the query sequence into sliding windows. GeneralMatch defines *J-sliding windows* and *J-disjoint windows* by the generalization of sliding windows and disjoint windows, respectively, and performs subsequence matching using these generalized windows. Except for a difference in the window construction mechanism, index building and subsequence matching algorithms of DualMatch and GeneralMatch are similar to those of FRM.

Loh et al. [8] have shown that FRM cannot be directly used for the normalization-transformed subsequence matching, i.e., we cannot use FRM for the distortion-free subsequence matching directly. The reason is that Eq. (1) in FRM do not hold any more for the normalization transformed sequences. To solve the problem, Loh et al. [8] have proved that the following Eq. (2) holds and used Eq. (2) for the normalization-transformed subsequence matching.

$$D(\overline{S}, \overline{Q}) \le \epsilon \implies D(\overline{S[i:j]}, \overline{Q[i:j]}) \le \epsilon' \qquad (2)$$

In Eq. (2), ϵ' is $\sqrt{2\omega - 2\sqrt{\omega^2 - \omega \cdot \epsilon^2 \cdot \frac{\sigma^2(Q)}{\sigma^2(Q[i:j])}}}$, and ω is $Len(Q[i:j])$. By using Eq. (2), if a normalization-transformed query window $\overline{Q[i:j]}$ is in ϵ'-match with a normalization-transformed data window $\overline{S[i:j]}$, the method regards the sequence $\overline{S}$ containing the window $\overline{S[i:j]}$ as a candidate sequence. That is, it stores the normalization-transformed data windows in the index, and uses the new tolerance ϵ' instead of the given tolerance ϵ when searching the index. The method by Loh et al. has a serious problem that overall performance becomes much worse if the query sequence length is two or more times longer than the window size. It is because, as the query sequence length increases, the index search range ϵ' becomes much larger than $\epsilon/\sqrt{p}$ used in FRM. To solve this search range problem, Loh et al. have proposed the notion of index interpolation, which constructs multiple indexes for multiple window sizes and selects an appropriate index for the given query sequence length. The index interpolation, however, causes another critical problem of incurring index overhead both in storage space and in update maintenance due to use of multiple indexes. (Refer to [12] for the more detailed motivation of the research.)

3 The Proposed Single Index Approach

In this section we adopt a single index approach [11] into the normalization-transformed subsequence matching. The single index approach has been proposed to solve the problem of index overhead in the moving average transform-based subsequence matching. We exploit this single index concept for the normalization-transformed subsequence matching.

3.1 Inclusion-Normalization Transform and the Proposed Method

We first formally define the notion of inclusion-normalization transform.

Definition 1 The *inclusion-normalization transform*ed sequence of $S[a:b]$ against $S[i:j]$ $(i \le a < b \le j)$, denoted by $\overline{S_{\{i:j\}}[a:b]}$, is defined as a new sequence whose entry $\overline{S_{\{i:j\}}[k]}(a \le k \le b)$ is set to $\frac{S[k]-\mu(S[i:j])}{\sigma(S[i:j])}$. □

According to Definition 1, for a window $S[a:b]$, the inclusion-normalization transform uses the mean and the standard deviation of $S[i:j]$ rather than those of $S[a:b]$. The notion of inclusion-normalization transform is based on the observation that, for a window $S[a:b]$ and a subsequence $S[i:j]$, $\mu(S[i:j])$ and $\sigma(S[i:j])$ would be very similar to $\mu(S[a:b])$ and $\sigma(S[a:b])$, respectively, since all entries in $S[a:b]$ are also contained in $S[i:j]$. Analogously, two transformed windows $\overline{S_{\{i:j\}}[a:b]}$ and $\overline{S_{\{i':j'\}}[a:b]}$ will also have the similar entry values even if $S[i:j]$ differs from $S[i':j']$.

We now propose a new normalization-transformed subsequence matching method, called *FRM-NT*(FRM that supports normalization transform)[2], which is devised from FRM by using the inclusion-normalization transform. While FRM converts each sliding window of data sequences to a point in the index, FRM-NT converts each

[2]We can also devise new matching methods that apply the inclusion-normalization transform to DualMatch [9] or GeneralMatch [10]. According to analysis, however, performance improvement of the methods is not significant compared with FRM-NT. Therefore, we present only FRM-NT that can be most easily derived from FRM.

window to an MBR that contains multiple points. That is, for each data window, FRM-NT first generates multiple transformed windows by performing the inclusion-normalization transform using every possible subsequence, then maps each transformed window to a lower-dimensional point, and finally constructs an MBR by containing all the points. Thus, each window is mapped into an MBR rather than a point, and the MBR is stored in the multidimensional index.

To show correctness of FRM-NT, we first present two lemmas. Lemma 1 shows the case where the subsequence to be compared with query sequence includes only one disjoint window, and Lemma 2 the case where the subsequence includes two or more disjoint windows.

Lemma 1 *If a data subsequence $S[i:j]$ includes a window $S[a:b]$, and Q is a query sequence, then the following Eq.* (3) *holds:*

$$D(\overline{S[i:j]}, \overline{Q}) \le \epsilon \Longrightarrow D(\overline{S_{\{i:j\}}[a:b]}, \overline{Q}[a-i+1:b-i+1]) \le \epsilon \quad (3)$$

PROOF: Refer to the reference [12]. □

Lemma 2 *If a data subsequence $S[i:j]$ includes p disjoint windows, $S[a_0 : a_1 - 1], S[a_1 : a_2 - 1], \cdots, S[a_{p-1} : a_p - 1]$, of length ω, and Q is a query sequence, then the following Eq.* (4) *holds:*

$$D(\overline{S[i:j]}, \overline{Q}) \le \epsilon \Longrightarrow \bigvee_{k=1}^{p} D(\overline{S_{\{i:j\}}[a_{k-1} : a_k - 1]}, \overline{Q}[a_{k-1} - i + 1 : a_k - i]) \le \frac{\epsilon}{\sqrt{p}} \quad (4)$$

PROOF: Refer to the reference [12]. □

Lemma 1 guarantees that the candidate set consisting of the subsequences $\overline{S[i:j]}$ such that $\overline{S_{\{i:j\}}[a:b]}$ and $\overline{Q}[a-i+1:b-i+1]$ are in ϵ-match contains no false dismissal. Also, Lemma 2 guarantees that the candidate set consisting of $\overline{S[i:j]}$ such that at least one of the window pairs $(\overline{S_{\{i:j\}}[a_{k-1} : a_k - 1]}, \overline{Q}[a_{k-1} - i + 1 : a_k - i])$ are in $\epsilon/\sqrt{p}$-match contains no false dismissal. (See Figures 2 and 3 in [12] for the more detailed explanation.)

We now explain correctness of FRM-NT. FRM-NT performs the normalization-transformed subsequence matching correctly based on the following Theorem 1.

Theorem 1 *If $\overline{Q}$ is in ϵ-match with $\overline{S[i:j]}$, then at least one k-th disjoint window of $\overline{Q}$ is in $\epsilon/\sqrt{p}$-match with the corresponding sliding window of $\overline{S[i:j]}$, where $1 \le k \le p$, and $p = \lfloor Len(Q)/\omega \rfloor$. That is, the following Eq.* (5) *holds:*

$$D(\overline{S[i:j]}, \overline{Q}) \le \epsilon \Longrightarrow \bigvee_{k=1}^{p} D(\overline{S_{\{i:j\}}[i+(k-1)\omega : i+k\omega-1]}, \overline{q_k}) \le \frac{\epsilon}{\sqrt{p}} \quad (5)$$

PROOF: According to Lemma 2, the following Eq. (6) holds:

$$D(\overline{S[i:j]}, \overline{Q}) \le \epsilon \Longrightarrow \bigvee_{k=1}^{p} D(\overline{S_{\{i:j\}}[a_{k-1} : a_k - 1]}, \overline{Q}[a_{k-1} - i + 1 : a_k - i]) \le \frac{\epsilon}{\sqrt{p}} \quad (6)$$

Let $a_0 = i, a_1 = i+\omega, \cdots$, and $a_{p-1} = i+(p-1)\omega$, i.e., let $a_{k-1} = i + (k-1)\omega$. Then, Eq. (6) can be represented as Eq. (7):

$$D(\overline{S[i:j]}, \overline{Q}) \le \epsilon \Longrightarrow \bigvee_{k=1}^{p} D(\overline{S_{\{i:j\}}[i+(k-1)\omega : i+k\omega-1]}, \overline{Q}[(k-1)\omega+1 : k\omega]) \le \frac{\epsilon}{\sqrt{p}} \quad (7)$$

According to the notation in Table 1, $\overline{Q}[(k-1)\omega+1 : k\omega]$ is denoted by $\overline{q_k}$. Thus, Eq. (7) is also represented as Eq. (5), and this completes the proof. □

Theorem 1 guarantees correctness of FRM-NT. That is, it guarantees that the candidate set consisting of the subsequences $\overline{S[i:j]}$ such that the disjoint window $\overline{q_k}$ and the corresponding sliding window $\overline{S_{\{i:j\}}[i+(k-1)\omega : i+k\omega-1]}$ are in $\epsilon/\sqrt{p}$-match (i.e., satisfying the necessary condition of Eq. (5)) contains no false dismissal.

3.2 Index Building and Subsequence Matching Algorithms

In this subsection we present index building and subsequence matching algorithms of FRM-NT. Figure 1 shows the index building algorithm of FRM-NT. In Step (1), we divide a data sequence S into sliding windows of length ω. In Steps (2) $\sim$ (6), for each sliding window, we construct an MBR and store the MBR into the multidimensional index. First, in Step (3), we make a set of transformed windows $\overline{S_{\{i:j\}}[a:b]}$ from each sliding window $S[a:b]$ by performing inclusion-normalization transform on every possible subsequence $S[i:j]$. (For the reason why we perform the inclusion-normalization transform for every possible position, refer to Figure 5 in [12].) Next, in Step (4), we construct an f-dimensional MBR by using the lower-dimensional transformation on the set of windows. Finally, in Step (5), we store the MBR into the multidimensional index with the starting offset of the corresponding sliding window.

Like FRM, however, FRM-NT has a problem of generating too many MBRs to be stored in the index since it divides data sequences into sliding windows. To solve this problem, FRM-NT also constructs an MBR that contains multiple MBRs corresponding to multiple sliding windows. That is, in the index building algorithm, we first construct an MBR that represents multiple consecutive sliding windows, and then store the MBR with the starting offsets of the first and the last windows into the index. For easy explanation and understanding, however, we describe the algorithm in Figure 1 as that FRM-NT stores an individual MBR for each sliding window directly. (For the detailed process of constructing the index, refer to Figure 6 in [12].)

After building the multidimensional index, we perform the normalization-transformed subsequence matching using the algorithm presented in Figure 2. In Steps (1) and (2),

Procedure *FRM-NT-BuildIndex*(Data Sequence *S*, Window size ω)
(1) Divide *S* into sliding windows of length ω;
(2) **for** each sliding window $S[a:b]$ **do**
(3) Make a set of inclusion-normalization transformed windows $\overline{S_{[i:j]}[a:b]}$ for each possible query length and each possible position of $S[i:j]$;
(4) Construct an *f*-dimensional MBR *f-D MBR* by using lower-dimensional transformations on a set of $\overline{S_{[i:j]}[a:b]}$'s;
(5) Make a record <*f-D MBR, offset=a*>, and store it into the index;
(6) **endfor**

Figure 1. Index building algorithm of FRM-NT.

Procedure *FRM-NT-SubsequenceMatching* (Query Sequence *Q*, Window size ω)
(1) Make $\overline{Q}$ from *Q* by using the normalization transform;
(2) Divide $\overline{Q}$ into disjoint windows $\overline{q_i}(1 \le i \le p)$ of length ω;
(3) **for** each disjoint window $\overline{q_i}$ **do**
(4) Transform the window to an *f*-dimensional point by using the feature extraction function;
(5) Construct a range query using the point and $\varepsilon/\sqrt{p}$;
(6) Search the index and find the records of the form <*f-D MBR, offset*>;
(7) Include in the candidate set the subsequences *S*[*i*:*j*] for each possible position;
// where $i = (offset + l) - k \cdot \omega$, $j = i + Len(Q) - 1$, $0 \le l \le \omega - 1$, and $0 \le k \le \lfloor q_{len}/\omega \rfloor - 1$
(8) **endfor**
(9) Do the post-processing step;

Figure 2. Subsequence matching algorithm of FRM-NT.

for a given query sequence Q, we obtain p disjoint windows $\overline{q_i}$ from the normalization-transformed sequence $\overline{Q}$. In Steps (3) $\sim$ (8), for each window $\overline{q_i}$, we find candidate subsequences by searching the index. First, in Step (4), we transform the corresponding window to an f-dimensional point using lower-dimensional transformation. Second, in Step (5), we construct a range query using the point and $\epsilon/\sqrt{p}$. Third, in Step (6), we search the index using the range query and find the MBRs that are in $\epsilon/\sqrt{p}$-match with the point. Last, in Step (7), we obtain candidate subsequences using *offset*, which is stored in the record with the MBR as the starting position of the corresponding sliding window. After obtaining a candidate set, we finally select only similar subsequences by discarding false alarms from the candidate set in Step (9).

4 Performance Evaluation

4.1 Experimental Data and Environment

We have performed experiments using two types of data sets. The first data set, a real stock data set used in earlier works [4, 9, 10] for subsequence matching, consists of 329,112 entries. We call this data set *STOCK-DATA*. The second data set, also used in the earlier works as a synthetic data set, contains random walk data consisting of one million entries: the first entry is set to 1.5, and subsequence entries are obtained by adding a random value in the range (-0.001,0.001) to the previous one. We call this data set *WALK-DATA*.

We evaluate two methods: FRM-NT proposed in Section 3 and the previous one by Loh et al. [8] (we simply call it *LKW* by taking authors' initials.). As the multidimensional index, we use the R*-tree [2] for both methods. And, we use 256 $\sim$ 1024 as the query sequence lengths to be given, and accordingly, we set the minimum window size of both methods to 256 [4]. Also, we use DFT(Discrete Fourier Transform) [4, 7, 9] as the feature extraction function and use six features [4, 9, 10]. The hardware platform is a PC equipped with an Intel Pentium IV 2.80GHz CPU, 512MB RAM, and a 70.0GB hard disk. The software platform is GNU/Linux Version 2.6.6 operating system.

As the metric of efficiency, we measure the elapsed time of each method. We generate query sequences from the data sequence by taking subsequences of length $Len(Q)$ starting from random offsets [4, 10]. To avoid effects of noise, we experiment with 10 different query sequences of the same length and use the average as the result. As the selectivity [4], we use 10^{-5} since lower selectivities will be much more important than higher ones for very large databases [8, 9].

4.2 Experimental Results

We have conducted two experiments: Experiment 1) uses only one index, and Experiment 2) multiple indexes.

Experiment 1): Figure 3 shows the experimental results when we use a single index for both methods. As depicted,

if the query sequence length is 256, i.e., if it equals to the window size, LKW is slightly better than FRM-NT in performance. It is because, while LKW uses the smallest search range when the query sequence and the window have the same size, MBR sizes in FRM-NT are generally larger than those in LKW due to use of inclusion-normalization transforms. For the cases where the query sequence length is longer than 256, however, our FRM-NT outperforms LKW. That reason is that, as we explained in Section 3, as the query sequence length increases, the index search range ϵ' of LKW will be greater than ϵ while that of FRM-NT will be reduced to $\epsilon / \sqrt{p}$.

In summary, we say that LKW is optimized for a specific query sequence length; in contrast, FRM-NT is evenly optimized for query sequences of arbitrary length. Thus, LKW shows the best performance when the query sequence and the window have the same size, but it becomes worse as the query sequence length increases compared with the window size. In contrast, FRM-NT shows relatively better performance than LKW in many cases where the query sequences are longer than the window. In summary of experimental results in Figure 3, FRM-NT improves performance by up to 2.8 times for STOCK-DATA and by up to 1.4 times for WALK-DATA compared with LKW.

Experiment 2): Figure 4 shows the results when we use multiple indexes as in [8] rather than a single index. We build three indexes for window sizes of 256, 512, and 1024. As depicted, for the case where the query sequence length is one of 256, 512, and 1024, i.e., where an index is built for the length, LKW shows slightly better performance than FRM-NT. It is because, as we explained in Experiment 1), LKW shows the best performance when the query sequence and the window have the same size. On the other hand, for the case where the query sequence length is one of 384, 640, 768, and 896, i.e., where an index is not built for the length, our FRM-NT shows better performance than LKW. It is because FRM-NT is evenly optimized for every query sequence length as well as for the window size. Likewise, we note that FRM-NT shows a comparable performance with LKW even if we use multiple indexes rather than a single index.

5 Conclusions

Normalization transform is known to be very useful for finding the overall trend of time-series data since it enables finding sequences with similar fluctuation patterns. In this paper we use this normalization transform for the distortion-free subsequence matching. Previous works with normalization transform, however, have a critical problem of incurring index overhead both in storage space and in update maintenance due to use of multiple indexes. To solve this problem, we have presented the new notion of *inclusion-normalization transform* and, based on the notion, proposed a single index approach for the normalization transformed subsequence matching. Using the single index approach, we can reduce the storage space and the index maintenance overhead.

The contribution of the paper can be summarized as follows. First, we have analyzed the problems of the previous work. Second, we have formally defined the inclusion-normalization transform by generalizing the original definition of normalization transform. Third, we have presented a related theorem to guarantee correctness of the inclusion-normalization transform-based subsequence matching and formally proved the theorem. Fourth, we have proposed subsequence matching and index building algorithms to implement the proposed method. Last, we have empirically shown superiority of our method. Experimental results for real stock data show that our method improves performance by up to 2.8 times over the previous method.

References

[1] Agrawal, R., Faloutsos, C., and Swami, A., "Efficient Similarity Search in Sequence Databases," In *Proc. the 4th Int'l Conf. on Foundations of Data Organization and Algorithms*, pp. 69-84, Oct. 1993.

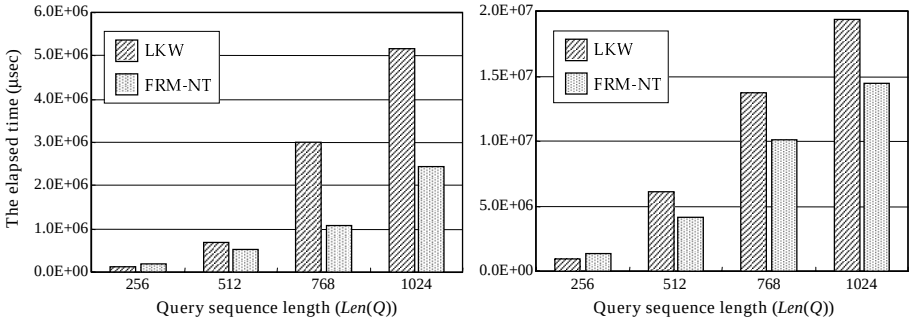
[2] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 322-331, May 1990.

[3] Chan, K.-P., Fu, A. W.-C., and Yu, C. T., "Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 15, No. 3, pp. 686-705, Jan./Feb. 2003.

[4] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., "Fast Subsequence Matching in Time-Series Databases," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 419-429, May 1994.

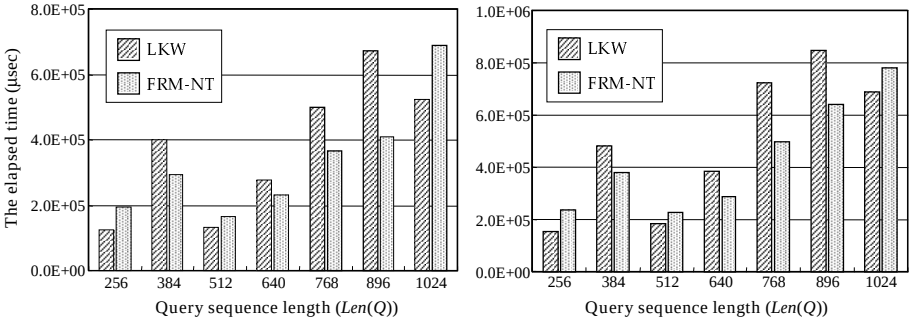
[5] Keogh, E. J. et al., "LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," In *Proc. Int'l Conf. on Very Large Data Bases*, Seoul, Korea, pp. 882-893, Sept. 2006.

[6] Kim, S.-W., Park, S., and Chu, W. W., "Efficient Processing of Similarity Search Under Time Warping in Sequence Databases: An Index-based Approach," *Information Systems*, Vol. 29, No. 5, pp. 405-420, July 2004.



(a) Results for STOCK-DATA (b) Results for WALK-DATA

Figure 3. Performance comparison of FRM-NT and LKW using a single index.



(a) Results for STOCK-DATA (b) Results for WALK-DATA

Figure 4. Performance comparison of FRM-NT and LKW using multiple indexes.

[7] Lim, S.-H., Park, H.-J., and Kim, S.-W., "Using Multiple Indexes for Efficient Subsequence Matching in Time-Series Databases," In *Proc. of the 11th Int'l Conf. on Database Systems for Advanced Applications*, pp. 65-79, Apr. 2006.

[8] Loh, W.-K., Kim, S.-W., and Whang, K.-Y., "A Subsequence Matching Algorithm that Supports Normalization Transform in Time-Series Databases," *Data Mining and Knowledge Discovery*, Vol. 9, No. 1, pp. 5-28, July 2004.

[9] Moon, Y.-S., Whang, K.-Y., and Loh, W.-K., "Duality-Based Subsequence Matching in Time-Series Databases," In *Proc. the 17th Int'l Conf. on Data Engineering (ICDE)*, IEEE, pp. 263-272, April 2001.

[10] Moon, Y.-S., Whang, K.-Y., and Han, W.-S., "General Match: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 382-393, June 2002.

[11] Moon, Y.-S. and Kim, J., "A Single Index Approach for Time-Series Subsequence Matching that Supports Moving Average Transform of Arbitrary Order," in *Proc. of the 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pp. 739-749, Apr. 2006.

[12] Moon, Y.-S. and Kim, J., "Fast Normalization-Transformed Subsequence Matching in Time-Series Databases," Technical Report, Kangwon National University, July 2006 (Also available at http://cs.kangwon.ac.kr/∼ysmoon/papers/norm-trans.pdf).

[13] Rafiei, D. and Mendelzon, A. O., "Querying Time Series Data Based on Similarity," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 12, No. 5, pp. 675-693, Sept./Oct. 2000.

[14] Wei, L., Keogh, E., Van Herle, H., and Mafra-Neto, A., "Atomic Wedgie: Efficient Query Filtering for Streaming Time Series," In *Proc. of the 5th IEEE Int'l Conf. on Data Mining*, pp. 490-497, Nov. 2005.

[15] Yi, B.-K., Jagadish, H. V., and Faloutsos, C., "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. the 14th Int'l Conf. on Data Engineering (ICDE)*, Orlando, Florida, pp. 201-208, Feb. 1998.