# Discovering Longest-lasting Correlation in Sequence Databases

Yuhong Li #1, Leong Hou U #2, Man Lung Yiu *3, Zhiguo Gong #4

#Department of Computer and Information Science, University of Macau
Av. Padre Tomás Pereira Taipa, Macau
1yb27407@umac.mo 2ryanlhu@umac.mo 4fstzgg@umac.mo

*Department of Computing, Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
3csmlyiu@comp.polyu.edu.hk

## ABSTRACT

Most existing work on sequence databases use correlation (e.g., *Euclidean distance* and *Pearson correlation*) as a main component for various analytical tasks. Typically, it requires users to set a `length` $\ell$ for the similarity queries. However, there is no steady way to define the `length` $\ell$ on different application needs. In this work we focus on discovering longest-lasting highly correlated subsequences in sequence databases, which is particularly useful in helping those analyses without prior knowledge about the query length $\ell$. Surprisingly, there has been limited work on this problem. A baseline solution is to calculate the correlations for every possible subsequence combination. Obviously, the brute force solution is not scalable for large datasets. In this work we study an index that gives a tight correlation bound for subsequences of similar length and offset. To the best of our knowledge, this is the first index to support normalized distance metric of arbitrary length subsequences. Inspired by the correlation bound, a batch processing is applied to discover the longest-lasting correlated result. Extensive experimental evaluation on both real and synthetic sequence datasets verifies the efficiency and effectiveness of our proposed methods.

## 1. INTRODUCTION

Sequence data can be found in a variety of applications nowadays, such as network analysis, image processing, financial data analysis, and sensor network monitoring. As a consequence, processing and mining of the sequence data have been developed for these applications in the last decade, such as similarity and correlation[1] analysis [7, 10, 11, 34], subsequence matching [3, 12, 30], motif discovery [28], and shapelet mining [27]. One key aspect among these works is to exploit the correlation in sequence data, which enables data analysts with the ability to explore, digest, and interpret the trends in the sequence data.

[1]In this work the terms 'correlation' and 'similarity' are interchangeable.

The problem in this paper is to discover longest-lasting correlated subsequences (LCS). A subsequence pair $(q, o)$ is longest-lasting correlated if and only if the length of the subsequence $\ell$ is maximized subject to $\rho(q, o, \tau, \ell) \geq \delta$, where $\rho(q, o, \tau, \ell)$ is the correlation of $q$ and $o$ in sequence segment $[\tau, \tau + \ell - 1]$. Surprisingly, there has been limited work on this problem. In this work, we focus on lock-step correlation measures (e.g., Euclidean distance that only compares $i$-th point of one sequence to the $i$-th point of another) as the underlying distance measure, as suggested by a pioneer time series research group [11].[2]

Longest-lasting correlated subsequences are particularly useful in helping those analyses without prior knowledge of the query length $\ell$. For instance, a stock analyst wants to find a stock whose price variance is similar to Google, Inc. in some segment from 2008 to 2011. This question can be answered easily by subsequence matching [12] if we have prior knowledge about the segment length $\ell$. However, $\ell$ is not easy to be specified as the most appropriate value of $\ell$ heavily depends on queries, time, data, and application domains. Instead of finding fixed length results, our task is to return the longest-lasting segment whose correlation is larger than a threshold $\delta$. We claim that the correlation threshold $\delta$ is a more natural parameter than the segment length $\ell$ since analysts can evaluate how the correlation score reflects the relevance of the result in their application domain.

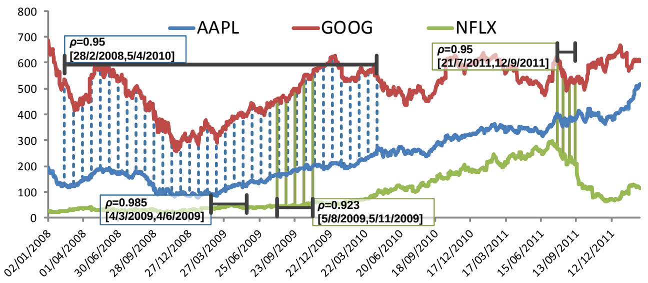


**Figure 1: Example of finance analysis**

[2]As reported in [11], there is no statistically significant difference in accuracy between the lock-step measures and the elastic measures (e.g., dynamic time warping that compares $i$-th point of one sequence to $j$-th point of another) when the dataset contains more than few hundred objects. More importantly, some application is more applicable to the lock-step measures if the values are collected periodically without error.

**Applications.** We demonstrate our longest-lasting correlated subsequence queries using the stock data collected from Yahoo! Finance [3]. Figure 1 illustrates the price variance of $GOOG$ (Google, Inc.), $NFLX$ (Netflix, Inc.), and $AAPL$ (Apple, Inc.) in 2008 - 2010. A typical analysis query is '*find the most correlated stock to GOOG for every 3 months data in 2008-2011*'. This query returns $AAPL$ in [4/3/2009,4/6/2009] as the result where the correlation $\rho$ is 0.985. The query result may be more interesting to analysts if it becomes '*find the longest-lasting period of a stock who performs similar to GOOG in 2008-2011*'. Suppose that the correlation threshold $\rho$ is set to 0.95, this query returns $AAPL$ in [28/2/2008,5/4/2010] as the result. It is not surprising that their prices change similarly over such long period as both of them are the leading companies in IT sector. Besides, the second type of queries can identify prominent periods more precisely based on the correlations. For instance, the longest correlated time span of $GOOG$ and $NFLX$ fulfilled the correlation threshold is [21/7/2011,12/9/2011] while the fixed length query ($\ell$=3 months) returns a lower correlation ($\rho = 0.923$) in time span [5/8/2009,5/11/2009].

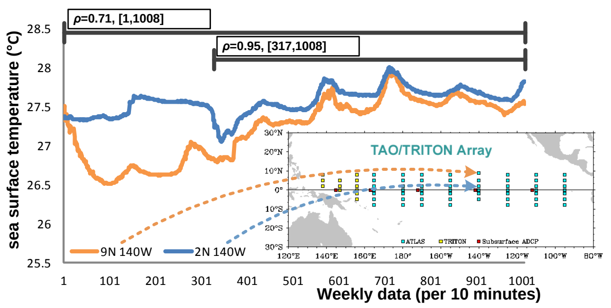


**Figure 2: Example of climate analysis**

Longest-lasting correlated subsequences is particularly helpful in climatology. Figure 2 shows the configuration of Tropical Atmosphere Ocean (TAO) array in Pacific Ocean [4]. Given the sea surface temperature of a specified sensor over a period of time (e.g., one week), one possible query is to find *the most correlated sensor during the same period*. However, there may be no such pair where their correlation can be viewed as *correlated* (e.g., $\rho < 0.9$ for all pairs). In Figure 2, the correlation of the entire weekly data of sensors '9N 140W' and '2N 140W' is 0.71. However, the correlation of a segment in weekly data (from 317 to 1008) is much higher (i.e., 0.95), where the meteorologists may be interested in this period and further investigate the reason behind this.

Besides finance and climatology analysis, longest-lasting correlated subsequences are also useful in hyperspectral imaging exploitation [9]. A hyperspectral image is captured by remote sensors that collect image data simultaneously in dozens or hundreds of narrow, adjacent spectral bands. These spectral bands make it possible to derive a continuous spectrum for each image cell. One research task of hyperspectral imaging is to discover the spectral signature of materials. Typically, a spectral signature consists of a subset of consecutive spectral bands. Two image cells may refer to the same material if their spectral signatures are similar. Given an image cell and a specified correlation threshold, the longest-lasting correlated subsequence query returns a signature candidate that can be used for further exploitation.

[3]http://finance.yahoo.com/

[4]http://www.pmel.noaa.gov/tao/

**Problem Definition.** In this paper we use Pearson correlation, which is equivalent to $Z$-normalized Euclidean distance (see Sec. 2.1 for the detail), to measure the closeness of sequences. The definition of Pearson correlation between two subsequences on a specific interval $[\tau, \tau+\ell-1]$ is defined as follows.

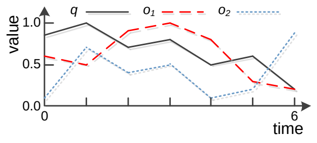
DEFINITION 1 (SUBSEQUENCE PEARSON CORRELATION). *Given a query $q$, an object sequence $o$, and a specific segment $[\tau, \tau+\ell-1]$, the Pearson correlation between $(q[\tau], ..., q[\tau+\ell-1])$ and $(o[\tau], ..., o[\tau+\ell-1])$ is defined as*

$$\rho(q,o,\tau,\ell) = \frac{\ell \sum_{i=\tau}^{\tau+\ell-1} q[i]o[i] - \sum_{i=\tau}^{\tau+\ell-1} q[i] \sum_{i=\tau}^{\tau+\ell-1} o[i]}{\sqrt{\ell \sum_{i=\tau}^{\tau+\ell-1} q[i]^2 - (\sum_{i=\tau}^{\tau+\ell-1} q[i])^2}\sqrt{\ell \sum_{i=\tau}^{\tau+\ell-1} o[i]^2 - (\sum_{i=\tau}^{\tau+\ell-1} o[i])^2}} \quad (1)$$

Based on Def. 1, we formally define the Longest-lasting Correlated Subsequence Query (LCS) as follows.

DEFINITION 2. (LONGEST-LASTING CORRELATED SUBSEQUENCE QUERY, LCS) *Given a query $q$, a sequence database $O$, and a threshold $\delta$, a Longest-lasting Correlated Subsequence Query (LCS) returns an object subsequence $o_{lcs}(\tau_{lcs}, \ell_{lcs}), o_{lcs} \in O$ such that $\rho(q, o_{lcs}, \tau_{lcs}, \ell_{lcs}) > \delta$ and the length $\ell_{lcs}$ is the longest among all possible subsequences $o_i(\tau_i, \ell_i), o_i \in O$ having $\rho(q, o_i, \tau_i, \ell_i) > \delta$.*

Fig. 3(a) illustrates a query sequence and a sequence database having two objects, where their raw values can be found in Fig. 3(b). For every subsequence combination (i.e., varying $\tau$ and $\ell$), we can use Eq. 1 to calculate the corresponding correlation of $(q, o_1)$ (shown in Fig. 3(c)) and $(q, o_2)$ (shown in Fig. 3(d)), respectively. Given a threshold $\delta = 0.95$, LCS$(q, \{o_1, o_2\}, \delta)$ returns $o_2(1,5)$ as the result since there is no subsequence having length $\ell > 5$ and correlation $\rho > 0.95$.



(a) Stock sequences

| time | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $q$ | 0.8 | 1.0 | 0.7 | 0.8 | 0.5 | 0.6 | 0.2 |
| $o_1$ | 0.6 | 0.5 | 0.9 | 1.0 | 0.8 | 0.3 | 0.2 |
| $o_2$ | 0.1 | 0.7 | 0.4 | 0.5 | 0.1 | 0.2 | 0.9 |

(b) Stock prices

| $\ell$ | 7 | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|
| $\tau$=0 | 0.44 | -0.07 | -0.52 | -0.72 | -0.89 | -1.00 |
| 1 | - | 0.46 | -0.04 | -0.52 | -0.87 | -1.00 |
| 2 | - | - | 0.77 | 0.50 | **0.98** | 1.00 |
| 3 | - | - | - | 0.73 | 0.45 | 1.00 |
| 4 | - | - | - | - | 0.42 | -1.00 |
| 5 | - | - | - | - | - | 1.00 |

(c) $\rho(q, o_1, \tau, \ell)$

| $\ell$ | 7 | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|
| $\tau$=0 | -0.20 | 0.77 | 0.75 | 0.61 | 0.65 | 1.00 |
| 1 | - | -0.11 | **0.99** | 0.99 | 1.00 | 1.00 |
| 2 | - | - | -0.51 | 0.99 | 1.00 | 1.00 |
| 3 | - | - | - | -0.53 | 1.00 | 1.00 |
| 4 | - | - | - | - | -0.94 | 1.00 |
| 5 | - | - | - | - | - | -1.00 |

(d) $\rho(q, o_2, \tau, \ell)$

**Figure 3: A concrete example of $k$LCS**

A naïve extension of LCS to $k$LCS is to return $k$ subsequences such that there is no other subsequence being longer than the result subject to the correlation constraint $\rho > \delta$. However, this simple definition may return a lot of *contained* subsequences [5]. In our running example, the first two results are $o_2(1,5)$ and $o_2(1,4)$

[5]As reported by [31], a query is trivially correlated to two close subsequences which may give meaningless result.

where $o_2(1,4)$ is *contained* inside $o_2(1,5)$. To address this, we propose an alternative definition, denoted as $k$-longest-lasting correlated subsequence query, that eliminates all contained subsequences from the result. This is formally defined as follows.

DEFINITION 3. ($k$-LONGEST-LASTING CORRELATED SUBSEQUENCE QUERY, $k$LCS) *Given a query q, a sequence database O, a threshold $\delta$, a k-Longest-lasting Correlated Subsequence Query (kLCS) returns a set of k non self-contained subsequences, R, such that each $o_i(\tau,\ell) \in R$ satisfies (1) $\rho(q,o_i,\tau,\ell) > \delta$ and (2) $\forall o_j(\tau',\ell') \notin R$, if $\rho(q,o_j,\tau',\ell') > \delta$, then $\ell' \leq \ell$.*

The first condition is to secure the correlation threshold and the second condition is to secure there is no other subsequence having longer length than the subsequence of the result set $R$. In the running example, 2LCS$(q,\{o_1,o_2\},0.95)$ returns $o_2(1,5)$ and $o_1(2,3)$ as the results according to Def. 3. Although both $o_2(1,4)$ and $o_2(2,4)$ fulfill the correlation constraint and their length $\ell$ is longer than the second result $o_1(2,3)$, these subsequences are eliminated since they are contained inside the first result $o_2(1,5)$.

**Challenges.** A brute force solution is to calculate the correlations between query and objects for every possible subsequence combination. It is obvious that the computation complexity is $O(Cnm^2)$, where $C$ is the complexity of a correlation computation, $n$ and $m$ are the cardinality and the maximum length of the sequence data, respectively. The brute force solution is not scalable as both $n$ and $m$ may be large in some applications. For instance, $n$ may refer to the number of signatures in hyperspectral image databases (e.g., 100k) and $m$ may refer to the hourly stock price variances (e.g., $\sim$2500 per year).

To reduce the huge search space, we can compute the correlations by various indexing techniques, such as k-gram indexing [14], suffix trees [25], and vector space indexing [12, 13, 17, 36, 37]. Most of these index structures only support fixed-length correlation queries (i.e., $\ell$ is fixed). A simple solution is to build an index for every possible subsequence combination (i.e., building an index $I_{\tau,\ell}$ for one specific $\tau$ and $\ell$ combination). However, precomputing and storing these indices make this approach infeasible as the total number of combinations is $\frac{m(m+1)}{2}$.

Another category of solutions is to build a unified index that is capable of computing arbitrary length correlations. Among all existing techniques, prefix search [12], multi-resolution [17], and reference net [37] fall into this category but they can only support non-normalized correlation metrics. However, as we will discuss in Sec. 2.1, the non-normalized correlation metrics do not reveal the true similarity of the sequences. The necessity of normalizing sequence values for similarity search is also justified by a recent publication [33]. To the best of our knowledge, there is neither an efficient nor a feasible solution to answer the longest-lasting correlated subsequence queries. Therefore, we study a novel index structure that supports efficient similarity computation constrained by a reasonable storage overhead.

**Summary of Contributions and Outline.**

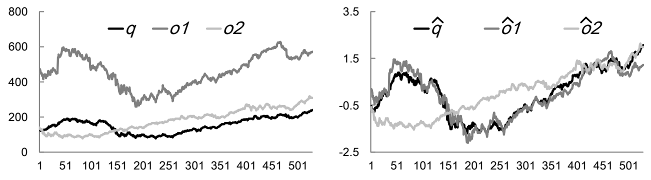
1. We define a new query, *longest-lasting correlated query*, in sequence databases, which is useful in many real-world applications.
2. We propose an $\alpha$-skipping technique to reduce the Pearson computation from $O(\ell)$ to $O(\alpha)$, which significantly shorten the computation time of a similarity calculation.
3. We propose a size-tunable index, *diamond cover index*, to efficiently compute $k$LCS while the index size can be adjusted to a given size $\mathcal{S}$. To the best of our knowledge, this is the first index to support normalized distance metric of arbitrary length subsequences.

The rest of this paper is organized as follows. In Sec. 2, we justify the definitions and challenges of the problem. Sec. 3 presents two non-index methods, SOTA and SKIP, for answering $k$LCS. Subsequently, we discuss our index techniques in Sec. 4. We thoroughly evaluate our proposed methods in Sec. 5. The related work is summarized in Sec. 6 and we conclude our work in Sec. 7.

## 2. JUSTIFICATION

### 2.1 Sequence Normalization

To reasonably compare the correlation of different sequences, the sequence values are necessarily normalized as discussed in [19, 33]. We use the following example to demonstrate the effect of normalization in correlation measurements.



(a) Raw sequences (b) Normalized sequences

| $d_2(q,o_1)$ | $d_2(q,o_2)$ |
|---|---|
| 7118.29 | 1514.83 |

(c) L2 of $q$ and $o$

| $\hat{d}_2(q,o_1)$ | $\hat{d}_2(q,o_2)$ |
|---|---|
| 7.26 | 24.40 |

(d) L2 of $\hat{q}$ and $\hat{o}$

**Figure 4: The effect of normalization**

Fig. 4(a) and 4(b) illustrate three sequences by their raw and normalized values, respectively, where the Euclidean distances of the query $q$ and the objects $o_1$ and $o_2$ are shown in Fig. 4(c) and Fig. 4(d). As clearly shown in Fig. 4(c), $q$ is more similar to $o_2$ if the distances are computed based on their raw values. However, in typical analysis tasks, we may concern more about the trend (i.e., shape) of two sequences instead of their actual distance. The distances reported in Fig. 4(c) are greatly overstates the subjective dissimilarity. Normalizing the values reveals the true similarity of the sequences as shown in Fig. 4(b) and Fig. 4(d). Formally, the value of a subsequence $o(\tau,\ell)$ is normalized by standard score ($Z$-normalization) as follows [33].

$$\hat{o}(\tau,\ell)[i] = \frac{o(\tau,\ell)[i] - \mu_{o(\tau,\ell)}}{\sigma_{o(\tau,\ell)}} \qquad (2)$$

where $\mu_{o(\tau,\ell)}$ and $\sigma_{o(\tau,\ell)}$ denote the mean and standard deviation of $o(\tau,\ell)$, respectively and $\hat{o}(\tau,\ell)$ denotes the normalized form of the subsequence $o(\tau,\ell)$. For the sake of presentation, we also denote $\hat{d}_p(q,o,\tau,\ell)$ as the $p$-norm distance of two $Z$-normalized sequences $\hat{q}(\tau,\ell)$ and $\hat{o}(\tau,\ell)$.

Besides $Z$-normalization, the $p$-norm distance is necessarily normalized by their length in order to produce reasonable $k$LCS result. It is obviously unfair to compare the subsequence correlations of different lengths. As an example, the $p$-norm distance of two longer sequences is more likely larger than the $p$-norm distance of two shorter sequences.

In this work, we focus on Pearson correlation since it not only reveals the true similarity of the sequences by *Z-normalization* but also makes the correlation comparison more fair by *length* *normalization.* The Pearson correlation between two subsequences, $q(\tau, \ell)$ and $o(\tau, \ell)$, actually can be represented by the $Z$-normalized Euclidean distance as follows [32, 38].

$$\rho(q, o, \tau, \ell) = 1 - \frac{(\hat{d}_2(q, o, \tau, \ell))^2}{2\ell} \quad (3)$$

where $\hat{d}_2(q, o, \tau, \ell)$ is normalized by the length $\ell$ in the Pearson correlation.

## 2.2 Indexing Arbitrary Length Queries?

If the length of correlation queries is decided ahead of time, then the computation can be boosted dramatically by plenty of existing indexing techniques [2, 7, 13, 18, 36, 35]. However, in $k$LCS, the length of the correlation queries is varying during the execution. There are two research groups which have suggested techniques to index arbitrary length queries [17, 22]. Their methods require to build multiple indices of various lengths, e.g., building 8 indices of length 8, 16, ..., 1024. A correlation query of length $\ell$ searches the multiple indices and interpolates the results to produce the result of $\ell$. For instance, a correlation query of length $\ell = 55$ can be calculated by searching two indices of length 16 and 32 by the techniques proposed in [17].

However, such multi-indices approaches can only return proper results for non-normalized distance metrics. Suppose that $o(0, 32)$ is the most correlated object to $q(0, 32)$, the $Z$-normalized Euclidean distance $\hat{d}_2(q, o, 0, 32)$ is neither an upper bound nor a lower bound of $\hat{d}_2(q, o, 0, 33)$. This is because the subsequence values must be re-normalized when the length is changed from 32 to 33, i.e., the values of $\mu$ and $\sigma$ may be changed significantly at different length subsequences. To the best of our knowledge, there is no existing indexing framework that can support correlation queries of arbitrary lengths in feasible storage overhead. This is also concluded in a recent publication [33].

## 2.3 Is Computing $k$LCS Time Consuming?

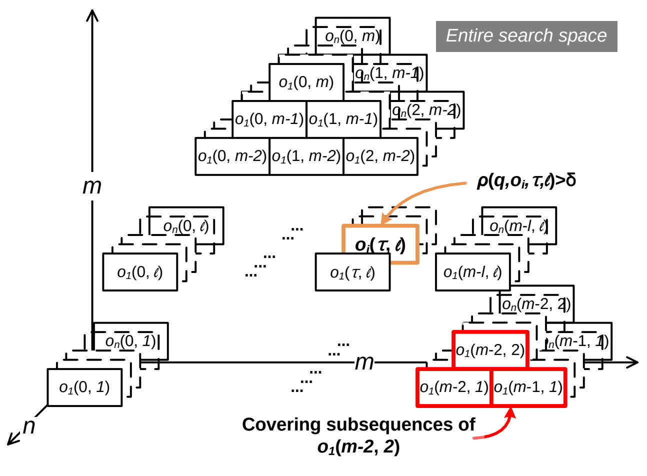


**Figure 5: Entire search space of $k$LCS$(q, O, \delta)$**

We claim that $k$LCS is a challenging problem due to (1) its potentially huge search space and (2) no monotonic property held for normalized distance measures. To find the longest-lasting correlation subsequence, LCS searches every possible subsequence (by varying $\tau$ and $\ell$) until it finds an object subsequence $o_i(\tau, \ell)$ such that $\ell$ is maximized subject to the correlation constraint $\delta$. Fig. 5 illustrates the entire search space of LCS, where the number of possible subsequence combinations is $\frac{m(m+1)}{2}$ and every combination has $n$ objects to be verified. Thereby, the total search space is $O(nm^2)$ and the time complexity is $O(Cnm^2)$, where $C$ is the complexity of a correlation computation.

DEFINITION 4 (COVERING SUBSEQUENCE, $\triangleright$). *$o(\tau, \ell)$ is covering $o(\tau', \ell')$, denoted as $o(\tau, \ell) \triangleright o(\tau', \ell')$, if and only if $\tau' \in [\tau ... \tau + \ell - 1]$ and $\ell' \in [1..\tau + \ell - \tau']$.*

For the sake of the following discussion, we first define the covering relationship between subsequences in Def. 4. Fig. 5 highlights three subsequences covered by $o_1(m - 2, 2)$.

LEMMA 1 (NON-MONOTONICITY). *Given two sequences $q(\tau, \ell)$ and $o(\tau, \ell)$, their normalized p-norm distance $\hat{d}_p(q, o, \tau, \ell)$ is neither monotone increasing nor monotone decreasing for the covering subsequence distance $\hat{d}_p(q, o, \tau', \ell')$, where $q(\tau, \ell) \triangleright q(\tau', \ell')$ and $o(\tau, \ell) \triangleright o(\tau', \ell')$.*

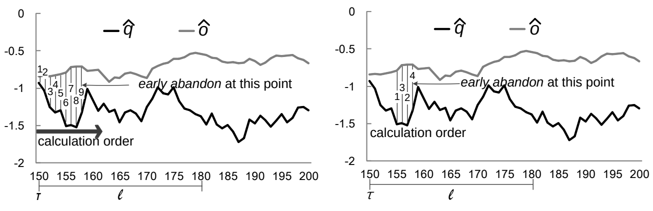
To discover the longest subsequence that satisfies the correlation threshold, a better execution paradigm is to apply binary search if there is any monotone property held for subsequence length $\ell$ or offset $\tau$. Nevertheless, as Lemma 1 shown[6], the normalized $p$-norm distance of a pair sequences does not hold any monotonicity for their covering subsequence distances. For instance, $\rho(q, o_i, \tau, \ell)$ can be either smaller or larger than $\rho(q, o_i, \tau, \ell - 1)$. Thereby, simple search techniques (e.g., binary search and branch-and-bound) are not able to help $k$LCS processing. Thereby, the time complexity remains $O(Cnm^2)$.

## 3. ANSWERING $k$LCS WITH NO-INDEX

In this section, we present two solutions of $k$LCS without using any index structure. The first one is a baseline solution that adapts the state-of-the-art techniques. In the second solution, we exploit the properties of Pearson correlation and reduce the complexity of each Pearson correlation from $O(\ell)$ to $O(\alpha)$.

## 3.1 Adaption of State-of-the-art Techniques

According to the discussion in Sec. 2.3, neither binary search nor branch-and-bound approach is applicable for discovering $k$LCS. To support early termination, we need a top-down execution paradigm which calculates the similarity of the subsequences for every $\tau$ and $\ell$ setting exhaustively from the longest length to the shortest length. We can terminate the 1LCS search immediately once an object subsequence $o_i(\tau, \ell)$ fulfills the correlation threshold $\delta$.



(a) Z-normalization (b) Reordering

**Figure 6: Early abandoning techniques**

However, calculating the correlation of subsequences exhaustively is a time-consuming task, where each Pearson correlation

[6]For the sake of presentation and space limitation, we put the proof of all Lemmas in Appendix and omit the proof of all Corollaries.

takes $O(\ell)$ time to compute. To the best of our knowledge, Rakthanmanon et al. [33] is the first work to consider optimizing the normalization step of the normalized distance computation. In their work, a new technique called *Z-normalization early abandonment* is proposed for early abandoning the calculation of the normalized Euclidean distance, $\hat{d}_2$. Suppose that we are discovering $k$LCS result at $\ell$ length subsequences. To compute the normalized Euclidean distance of $q(\tau,\ell)$ and $o_i(\tau,\ell)$, we can incrementally sum the individual distances to $\hat{d}_2(q,o_i,\tau,\ell)$ until the Pearson correlation becomes lower than the threshold $\delta$ [20, 33]. In Fig. 6(a), we can abandon the calculation of $\hat{d}_2(q,o,\tau,\ell)$ at 9th position if the summation of the individual distances from 1st to 9th position is already larger than $\sqrt{2\ell(1-\delta)}$ (see Eq. 3).

To incrementally sum the individual distances to $\hat{d}_2$, we need to prepare the mean $\mu$ and standard deviation $\sigma$ of $q(\tau,\ell)$ and $o_i(\tau,\ell)$ as shown in Eq. 2, which takes $O(\ell)$ time in total. Inspired by [33], the preparation can be done in $O(1)$ time by keeping two running sums of the subsequence values with a lag of exactly $\ell$ subsequence values. The mathematical representations are given below for clarity.

$$\mu_{\mathcal{X}(\tau,\ell)} = \frac{1}{\ell}\left(\sum_{i=0}^{\tau+\ell-1}\mathcal{X}[i] - \sum_{i=0}^{\tau}\mathcal{X}[i] + \mathcal{X}[\tau]\right) \tag{4}$$

$$\sigma^2_{\mathcal{X}(\tau,\ell)} = \frac{1}{\ell}\left(\sum_{i=0}^{\tau+\ell-1}\mathcal{X}[i]^2 - \sum_{i=0}^{\tau}\mathcal{X}[i]^2 + \mathcal{X}[\tau]^2\right) - \mu^2_{\mathcal{X}(\tau,\ell)} \tag{5}$$

where $\mathcal{X} \in \{q,o\}$. For instance, suppose we have two running sums $\sum_{i=0}^{\tau+\ell-1}\mathcal{X}[i]$ and $\sum_{i=0}^{\tau}\mathcal{X}[i]$, $\mu_{\mathcal{X}(\tau+1,\ell)}$ can be computed by incrementally summing the value of $\mathcal{X}[\tau+\ell]$ and $\mathcal{X}[\tau+1]$ to $\sum_{i=0}^{\tau+\ell-1}\mathcal{X}[i]$ and $\sum_{i=0}^{\tau}\mathcal{X}[i]$, respectively.

To further boost up the performance, we apply another technique called *reordering early abandon* [33], which reorders the calculations as shown in Fig. 6(b). According to [33], the sections of the query $q(\tau,\ell)$ that are farthest from the *mean* will *on average* have the largest contributions to the distance measure. In other words, for each subsequence $o_i(\tau,\ell)$ in $k$LCS computation, the calculation order is based on the mean of $q(\tau,\ell)$.

**Algorithm 1** State-of-the-art (SOTA) algorithm

```
    Algorithm SOTA( Query $q$, Database $O$, Threshold $\delta$, Maximum sequence length $m$, Result size $k$ )
 1: for $\ell := m$ to 1 do
 2:     for $\tau := 0$ to $m-\ell$ do
 3:         update $\sum_{i=0}^{\tau+\ell-1} q[i]$, $\sum_{i=0}^{\tau} q[i]$, $\sum_{i=0}^{\tau+\ell-1} q[i]^2$, $\sum_{i=0}^{\tau} q[i]^2$
 4:         determine the calculation order of $q(\tau,\ell)$ based on $\mu_{q(\tau,\ell)}$
 5:         for each sequence $o$ in $O$ do
 6:             update $\sum_{i=0}^{\tau+\ell-1} o[i]$, $\sum_{i=0}^{\tau} o[i]$, $\sum_{i=0}^{\tau+\ell-1} o[i]^2$, $\sum_{i=0}^{\tau} o[i]^2$
 7:             calculate $\hat{d}_2(q,o,\tau,\ell)$    ▷ early abandon at $\sqrt{2\ell(1-\delta)}$
 8:             if $\hat{d}_2(q,o,\tau,\ell) < \sqrt{2\ell(1-\delta)}$ then    ▷ by Eq. 3
 9:                 add $o_i(\tau,\ell)$ into $R$ if $r \not\triangleright o(\tau,\ell), \forall r \in R$
10:                 goto line 11 when $|R| = k$
11: return $R$
```

Alg. 1 (State-of-the-art, SOTA) applies all the techniques discussed above. SOTA exhaustively searches all $\tau$ and $\ell$ subsequences in a top-down manner. The normalized Euclidean distance $\hat{d}_2$ is computed by *Z-normalization* and *reordering* early abandonment techniques as discussed above. We add a subsequence object $o(\tau,\ell)$ into $R$ if the Pearson correlation $\rho(q,o,\tau,\ell)$ fulfills the correlation threshold $\delta$ and the object subsequence is not *contained* by other results in $R$ (at Line 9). The containment checking takes $O(\log k)$ time if we maintain $R$ by a segment tree. $R$ is returned as the result set when its size reaches $k$.

## 3.2 $\alpha$-Skipping Cumulative Arrays

Even though Alg. 1 (lines 6-7) utilizes the state-of-the-art techniques to calculate $\rho(q,o,\tau,\ell)$, it still takes $O(\ell)$ time to calculate one correlation in the worst case. Inspired by [27], the Pearson correlation can be computed in $O(1)$ time if we first compute five cumulative arrays for $q$ and $o$. More specifically, two of the arrays store the cumulative sum of the subsequence values of $q$ and $o$. Another two store the cumulative sum of square values. The last one stores the cumulative product sum of $q$ and $o$. The arrays are defined mathematically as follows.

$$S_q[u] = \sum_{i=0}^{u} q[i], \quad S_{q^2}[u] = \sum_{i=0}^{u} q[i]^2, \quad S_o[u] = \sum_{i=0}^{u} o[i], \quad S_{o^2}[u] = \sum_{i=0}^{u} o[i]^2, \quad S_{qo}[u] = \sum_{i=0}^{u} q[i]o[i] \tag{6}$$

where $u \in [0..m-1]$. Given these arrays $\{S_q, S_o, S_{q^2}, S_{o^2}, S_{qo}\}$, every component in Eq. 1 (including $\sum q[i]$, $\sum o[i]$, $\sum q[i]^2$, $\sum o[i]^2$, and $\sum q[i]o[i]$) of any $(\tau,\ell)$ setting can be computed in $O(1)$ time by the following equation.

$$\sum_{i=\tau}^{\tau+\ell-1}\mathcal{X}[i] = S_{\mathcal{X}}[\tau+\ell-1] - S_{\mathcal{X}}[\tau] + \mathcal{X}[\tau] \tag{7}$$

where $\mathcal{X} \in \{q,o,q^2,o^2,qo\}$ and $qo[i] = q[i]o[i]$.

However, the space overhead of computing $k$LCS is infeasible since every object $o_i$ is required to construct 3 extra arrays (e.g., $S_{o_i}, S_{o_i^2}, S_{qo_i}$). The total space overhead is $3mn$ that is three times larger than the raw data.[7] In the following, we present a technique called $\alpha$-skipping requires only $\frac{3mn}{\alpha}$ total space overhead ($\alpha$ can be chosen based on memory size). It can compute every Pearson correlation in $O(\alpha)$. We first define the $\alpha$-skipping cumulative array as follows.

DEFINITION 5 ($\alpha$-SKIPPING CUMULATIVE ARRAY, $S^{\alpha}_{\mathcal{X}}$). *Let the skip factor be $\alpha$. A cumulative value $S_{\mathcal{X}}[u]$ is kept into $S^{\alpha}_{\mathcal{X}}$ if and only if $u \mod \alpha = 0$.*

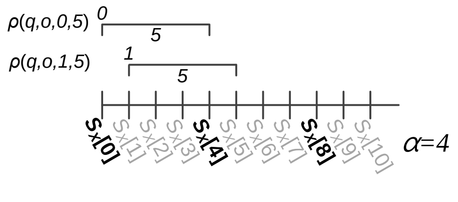


**Figure 7: An $\alpha$-skipping cumulative array**

Fig. 7 shows a 4-skipping cumulative array ($\alpha = 4$). Note that only the cumulative values in bold color are kept in $S^{\alpha}_{\mathcal{X}}$, where the size of $S^{\alpha}_{\mathcal{X}}$ is $1/\alpha$ to the original size. Suppose that $\tau = 1$ and $\ell = 5$, we need the value of $S_{\mathcal{X}}[1]$ and $S_{\mathcal{X}}[5]$ to compute $\sum_{i=1}^{5}\mathcal{X}[i]$ according to Eq. 6; however, neither $S_{\mathcal{X}}[1]$ nor $S_{\mathcal{X}}[5]$ is kept in $S^{\alpha}_{\mathcal{X}}$. In fact, these two values can be derived from $S_{\mathcal{X}}[0]$ and $S_{\mathcal{X}}[4]$ in 1 step. Lemma 2 shows that computing $\rho(q,o,\tau,\ell)$ takes $O(\alpha)$ time.

LEMMA 2. *Given $\alpha$-skipping cumulative arrays and raw data of $q$ and $o$, computing $\rho(q,o,\tau,\ell)$ takes $O(\alpha)$ time.*

During the $k$LCS computation, we can further reduce the computation time of $\rho(q,o,\tau,\ell)$ to $O(1)$ if we keep the corresponding

[7]The space overhead (i.e., $O(2m)$) of $S_q$ and $S_{q^2}$ is negligible.

running sums of $\rho(q, o, \tau - 1, \ell)$. We illustrate this more clearly by Fig. 7. After calculating $\rho(q, o, 0, 5)$, we collect five running sums, including $\sum_{i=0}^{4} q[i]$, $\sum_{i=0}^{4} q^2[i]$, $\sum_{i=0}^{4} o[i]$, $\sum_{i=0}^{4} o^2[i]$, and $\sum_{i=0}^{4} qo[i]$. When calculating the next offset correlation, $\rho(q, o, 1, 5)$, we update the value of each running sum by just subtracting $\mathcal{X}[0]$ from and adding $\mathcal{X}[5]$ into the running sum $\sum_{i=0}^{4} \mathcal{X}[i]$. Obviously, such *incremental* procedures can derive $\rho(q, o, \tau, \ell)$ in $O(1)$ time, which is faster than the $\alpha$-skipping techniques. Thereby, for every subsequence length setting $\ell$, we compute the running sums by the $\alpha$-skipping cumulative arrays at the first offset (i.e., $\tau = 0$); then incrementally update the running sums for the remaining offsets.

**Algorithm 2** SKIP algorithm

```
    Algorithm SKIP( Query $q$, Database $O$, Threshold $\delta$, Maximum sequence length $m$, Result size $k$, Skip factor $\alpha$ )
 1: calculate $S_q^1$ and $S_{q^2}^1$                                              ▷ $O(m)$
 2: calculate $S_{o_i}^\alpha$, $S_{o_i^2}^\alpha$, and $S_{qo_i}^\alpha$ for every $o_i \in O$          ▷ $O(nm)$
 3: for $\ell := m$ to 1 do
 4:     for $\tau := 0$ to $m - \ell$ do
 5:         for $\mathcal{X} \in \{q_i, q_i^2\}$ do
 6:             if $\tau = 0$ then calculate $\sum_{i=\tau}^{\tau+\ell-1} \mathcal{X}[i]$ by $S_{\mathcal{X}}^1$        ▷ $O(1)$
 7:             else update $\sum_{i=\tau}^{\tau+\ell-1} \mathcal{X}[i]$ incrementally           ▷ $O(1)$
 8:         for each sequence $o$ in $O$ do
 9:             for $\mathcal{X} \in \{o, o^2, qo\}$ do
10:                 if $\tau = 0$ then compute $\sum_{i=\tau}^{\tau+\ell-1} \mathcal{X}[i]$ by $S_{\mathcal{X}}^\alpha$   ▷ $O(\alpha)$
11:                 else update $\sum_{i=\tau}^{\tau+\ell-1} \mathcal{X}[i]$ incrementally        ▷ $O(1)$
12:             if $\rho(q, o, \tau, \ell) > \delta$ then                                 ▷ $O(1)$
13:                 add $o(\tau, \ell)$ into $R$ if $r \not\triangleright o(\tau, \ell), \forall r \in R$
14:                 goto line 15 if $|R| = k$
15: return $R$
```

Alg. 2 (SKIP) shows the pseudo code to discover $k$LCS based on the $\alpha$-skipping techniques. First, we calculate five $\alpha$-skipping cumulative arrays which take $O(nm)$ time in total.[8] The SKIP algorithm follows the same top-down execution paradigm of Alg. 1 but it calculates $\rho(q, o, \tau, \ell)$ by the Pearson equation (Eq. 1) based on the cumulative arrays. The running sums required in Eq. 1 are calculated by the $\alpha$-skipping arrays if $\tau = 0$ or calculated incrementally otherwise.

In summary, the SKIP algorithm is superior to the SOTA algorithm since it reduces the computation time for each Pearson correlation from $O(\ell)$ to $O(\alpha)$ with a reasonable space overhead $\frac{3mn}{\alpha}$. For instance, the overhead is just 6% of the object dataset when $\alpha$ is set to 50 and $m$ is 500.

## 4. ANSWERING $K$LCS WITH INDICES

Recall that no known index can support normalized distance queries of arbitrary lengths (see Sec. 2.2). We have illustrated the entire collection of elements that require indexing, as well as the query processing challenges, in Sec. 2.3.

In this section, we introduce a novel space-constrained index for boosting $k$LCS computation. Our index exploits an observation that subsequences of an object $o$ having similar offsets $\tau$ and lengths $\ell$ are likely to have similar correlation with $q$. For instance, if $\rho(q, o, \tau, \ell) < \delta$, then $\rho(q, o, \tau, \ell - 1)$ and $\rho(q, o, \tau + 1, \ell - 1)$ are likely to be smaller than $\delta$. This suggests that we can group similar subsequence together. Specifically, if the correlation upper bound of a subsequence group is below $\delta$, then such a group can be safely pruned.

[8]For clarity, the time complexity of every single step is stated in the pseudo codes.

In this section, we first discuss how to group subsequences of $o$ (with arbitrary lengths) by their Piecewise Aggregate Approximation (PAA) representations into Minimum Bounding Rectangles (MBRs), and derive their correlation upper bounds. Then, we elaborate how to effectively group the subsequences into a space-constrained index such that it offers good pruning power for querying.

### 4.1 Piecewise Aggregate Approximation

PAA is intuitive and simple, yet competitive with other sophisticated dimensionality reduction methods [23]. Specifically, a normalized subsequence $\hat{o}$ of offset $\tau$ and length $\ell$ can be represented in a $\phi$-dimensional PAA element $e_{\hat{o}}$, where the parameter $\phi$ is much smaller than $\ell$ in practice. Given a normalized subsequence $\hat{o}(\tau, \ell)$, the $i$-th element of the $\phi$-dimensional PAA element is defined as:

$$e_{\hat{o}(\tau,\ell)}[i] = \frac{\phi}{\ell} \sum_{j=\frac{\ell}{\phi}\cdot i}^{\frac{\ell}{\phi}(i+1)-1} \hat{o}[\tau + j] \quad (8)$$

Unless explicitly stated, we write $e_{\hat{o}(\tau,\ell)}[i]$ as $e_{\hat{o}}[i]$.

**PAA upper bound.** Zhu et al. [38] shows that the PAA distance $d_{PAA}$ of two PAA representations $e_{\hat{q}}$ and $e_{\hat{o}}$ is a lower bound of their 2-norm distance [9]. Formally, it is stated as:

$$\hat{d}(q, o, \tau, \ell) \geq d_{PAA}(e_{\hat{q}}, e_{\hat{o}}, \phi) = \sqrt{\ell/\phi} \cdot d_2(e_{\hat{q}}, e_{\hat{o}}, \phi) \quad (9)$$

where $d_2(e_{\hat{q}}, e_{\hat{o}}, \phi)$ denotes the 2-norm distance between two elements.

COROLLARY 1. *Given two subsequences $q(\tau, \ell)$ and $o(\tau, \ell)$ and set $UB(e_{\hat{q}}, e_{\hat{o}}, \phi) = 1 - d_{PAA}(e_{\hat{q}}, e_{\hat{o}}, \phi)^2/(2\ell) = 1 - d_2(e_{\hat{q}}, e_{\hat{o}}, \phi)^2/(2\phi)$, we have*

$$\rho(q, o, \tau, \ell) \leq UB(e_{\hat{q}}, e_{\hat{o}}, \phi) \quad (10)$$

Based on Eq. 3 and Eq. 9, the PAA distance $d_{PAA}$ provides an upper bound of the Pearson correlation $\rho$ (Corollary 1). In other words, if the upper bound, $UB(e_{\hat{q}}, e_{\hat{o}}, \phi)$, is smaller than the correlation threshold $\delta$, we can say $q(\tau, \ell)$ is not longest-lasting correlated to $o(\tau, \ell)$.

**PAA MBR upper bound.** In the following, we provide the upper bound of the Pearson correlation $\rho$ by two PAA MBRs (whose contained subsequences can have different lengths). Let $M_{\hat{q}}$ and $M_{\hat{o}}$ be the PAA MBRs of a subsequence set of $\hat{q}$ and $\hat{o}$, respectively. Inspired by Eq. 9, the minimum distance between $M_{\hat{q}}$ and $M_{\hat{o}}$ is defined as:

$$d_{PAA}(M_{\hat{q}}, M_{\hat{o}}, \phi) = \sqrt{\ell_{min}/\phi} \cdot d_2^{min}(M_{\hat{q}}, M_{\hat{o}}, \phi) \quad (11)$$

where $\ell_{min}$ is the minimum length of subsequences in $M_{\hat{q}}$ and $M_{\hat{o}}$, and $d_2^{min}(M_{\hat{q}}, M_{\hat{o}}, \phi)$ denotes the minimum 2-norm distance between two MBRs.

LEMMA 3. *The PAA distance between two PAA MBRs, $M_{\hat{q}}$ and $M_{\hat{o}}$, is a lower bound of the PAA distance between any $e_{\hat{q}(\tau,\ell)} \in M_{\hat{q}}$ and any $e_{\hat{o}(\tau,\ell)} \in M_{\hat{o}}$.*

$$d_{PAA}(e_{\hat{q}}, e_{\hat{o}}, \phi) \geq \sqrt{\ell/\ell_{min}} \cdot d_{PAA}(M_{\hat{q}}, M_{\hat{o}}, \phi) \quad (12)$$

*where $\ell$ is the length of $e_{\hat{q}}$ and $e_{\hat{o}}$,*

[9]Our techniques are also applicable to $p$-norm distance. The detail is omitted for simplicity.

Suppose that $e_{\hat{q}(\tau,\ell)}$ and $e_{\hat{o}(\tau,\ell)}$ are the elements in $M_{\hat{q}}$ and $M_{\hat{o}}$, respectively. We can derive the upper bound of the PAA distance of $e_{\hat{q}(\tau,\ell)}$ and $e_{\hat{o}(\tau,\ell)}$ by the PAA distance of their MBRs, which is shown in Lemma 3.

Let $UB(M_{\hat{q}}, M_{\hat{o}}, \phi) = 1 - d_{PAA}(M_{\hat{q}}, M_{\hat{o}}, \phi)^2/(2\ell_{min}) = 1 - d_2^{min}(M_{\hat{q}}, M_{\hat{o}}, \phi)^2/(2\phi)$, we can derive the Pearson correlation upper bound of $\hat{q}(\tau,\ell)$ and $\hat{o}(\tau,\ell)$ based on Corollary 1 and Lemma 3.

$$\rho(q, o, \tau, \ell) \le UB(M_{\hat{q}}, M_{\hat{o}}, \phi) \tag{13}$$

The above discussion shows that the distance of PAA MBRs, $d_{PAA}(M_{\hat{q}}, M_{\hat{o}}, \phi)$, provides an upper bound of the Pearson correlation, $\rho(q, o, \tau, \ell)$, where $e_{\hat{q}} \in M_{\hat{q}}$ and $e_{\hat{o}} \in M_{\hat{o}}$. Thereby, a straightforward indexing approach is to group the elements by a multidimensional index $\mathcal{I}$ (e.g., R*-tree [4], $k$-$d$ tree [5]). To discover all subsequences that satisfy the correlation threshold $\delta$, we traverse $\mathcal{I}$ from root to leaves recursively. For every seen non-leaf MBR $M_{\hat{o}}$, if its upper bound $UB(M_{\hat{q}}, M_{\hat{o}}, \phi)$ is less than the correlation threshold $\delta$, then $M_{\hat{o}}$ cannot contribute any result with respect to $M_{\hat{q}}$ such that it is safely pruned; otherwise, the children MBRs of $M_{\hat{o}}$ are further verified. For every seen leaf MBR $M'_{\hat{o}}$, if $UB(M_{\hat{q}}, M'_{\hat{o}}, \phi) \ge \delta$, then $M'_{\hat{o}}$ is expanded and verified. We report an element $o$ in $M'_{\hat{o}}$ as a result candidate if the length of $e_{\hat{o}}$ is $\ell$ and $\rho(q, o, \tau, \ell) \ge \delta$.

As illustrated in Figure 5, the total number of elements to be indexed is $O(nm^2)$. This huge storage space requirement renders the above indexing approach infeasible.

## 4.2 Diamond Cover Index (DCI)

In this section, we proposed a novel index structure, *Diamond Cover Index* (DCI) which can efficiently answer $k$LCS and reduce the index size to a given storage budget $\mathcal{S}$. DCI is based on two grouping strategies, *intra-object grouping* and *inter-object grouping*, which are thoroughly discussed in this section.

### 4.2.1 Intra-Object Grouping

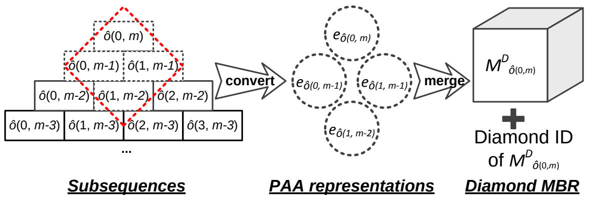
As discussed above, it takes $O(m^2)$ space for indexing the subsequences of an object $o$. In this section, we aim to reduce the index size to a manageable amount $\mathcal{S}$. Our idea is to group the subsequences of an object into a small number of MBRs.

**Diamond Cover Grouping.** A desirable grouping strategy should assign, among the subsequences of $o$, those having similar offsets and lengths into the same group. This would lead to 'tight' groups and maximize their pruning power during query processing. Specifically, we define the concept of *diamond MBR* as follows:

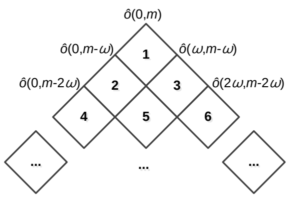
DEFINITION 6 (DIAMOND MBR, $M^D_{\hat{o}(\tau,\ell)}$). *Given the diamond side-length $\omega$, we define the diamond MBR $M^D_{\hat{o}(\tau,\ell)}$ as the MBR of all PAA representations of $e_{\hat{o}(\tau',\ell')}$, for all $\tau'$, $\ell'$ such that $\tau \le \tau' \le \tau + \omega$ and $(\tau + \ell) - (\tau' + \ell') \le \omega$.*

Fig. 8(a) illustrates an example with $\omega = 1$. The diamond MBR $M^D_{\hat{o}(0,m)}$ is the MBR of four PAA representations: $e_{\hat{o}(0,m)}, e_{\hat{o}(0,m-1)}, e_{\hat{o}(1,m-1)}, e_{\hat{o}(1,m-2)}$. As we will discuss later, the parameter $\omega$ provides a trade-off between the pruning power and the storage space.

Next, we present a technique called *diamond cover grouping* that partitions the PAAs (of an object) into a set of diamond MBRs. Fig. 8(b) illustrates an example of the diamond cover grouping. In this grouping, for each diamond MBR $M^D_{\hat{o}(\tau,\ell)}$, $\ell$ is either of the form $m - x \cdot \omega$ or 0, and $\tau$ is either of the form $y \cdot \omega$ or $m - \ell$, where $x, y$ are integers. Lemma 4 shows that the diamond cover grouping significantly reduces the storage overhead, from $O(m^2)$ to $O(m^2/\omega^2)$ per object.



(a) Diamond MBR example, with $\omega = 1$



(b) Diamond cover grouping

**Figure 8: Illustration of $\omega$-diamond cover grouping**

LEMMA 4 (NUMBER OF $\omega$-DIAMOND MBRS). *The number of $\omega$-diamond MBRs of an object is $O(m^2/\omega^2)$.*

To further save storage space, we encode the offset $\tau$ and the length $\ell$ of a diamond MBR into a *diamond ID*. These IDs are assigned to diamonds from top to bottom (see Fig. 8(b)). During query time, we can derive the values of $\tau$ and $\ell$ from a diamond ID conveniently. [10]

**Padding Zero Alignment.** We proceed to minimize the volume of a diamond MBR, and thus improve its pruning power.

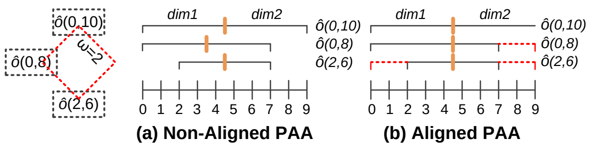


**Figure 9: Example of Padding Zero alignment**

Recall that a diamond MBR $M^D_{\hat{o}(\tau,\ell)}$ is built from the PAA representations $e_{\hat{o}(\tau',\ell')}$ of normalized subsequences $\hat{o}(\tau',\ell')$ of $o(\tau,\ell)$. Assume that $\omega = 2$ and $\phi = 2$ in the example of Fig. 9. The normalized subsequences for building $M^D_{\hat{o}(0,10)}$ are: $\hat{o}(0,10)$, $\hat{o}(0,8)$, $\hat{o}(2,6)$, and $\hat{o}(2,10)$. To construct their PAA representations, each subsequence is decomposed into $\phi = 2$ segments as shown in Fig. 9(a). For instance, $e_{\hat{o}(0,10)}$ and $e_{\hat{o}(0,8)}$ are built by two segments of 5 and 4 values of $o(0,10)$, respectively. However, these elements may not have strong correlation since they are built from subsequences of different lengths and offsets.

To improve the correlation, we propose to *align* the subsequences by padding zeroes such that every subsequence in $M^D_{o(\tau,\ell)}$ has the same length $\ell$ and the same offset $\tau$ in every PAA dimension. For instance, in Fig. 9(b), we pad zeroes to the subsequences $\hat{o}(0,8)$ and $\hat{o}(2,6)$, respectively, such that the aligned subsequences have the same length and offset as $\hat{o}(0,10)$. Note that padding zeroes into sequences does not affect any equation we discussed in Sec. 4.1. On the other hand, it improves the correlation

[10]We remove the detail of this simple conversion due to page limit.

of the elements, minimizes the volume of the diamond MBR, and thus provides better pruning ability during query processing.

**Construction Time.** Constructing a $\omega$-diamond MBR takes $O(\phi\omega^2)$ time, where $\omega^2$ is the number of subsequences in a diamond. For each dimension of an PAA element, it can be calculated in $O(1)$ time by using two cumulative arrays (i.e., $S_o$ and $S_{o^2}$). For the sake of presentation, we put the detail in Appendix.

#### 4.2.2 *Inter-Object Grouping*

In Section 4.2.1, we only group subsequences of *the same object* into $\omega$-diamond MBRs. This section focuses on grouping $\omega$-diamond MBRs of different objects into higher-level MBRs called *compact MBRs*.

For ease of query processing, only diamond MBRs $M^D_{\hat{o}(\tau,\ell)}$ (of different objects) with the same diamond ID are grouped into compact MBRs $M^C_{(\tau,\ell)}$. In order to support efficient query processing, we group similar diamond MBRs together into higher-level MBRs (i.e., compact MBRs) by existing grouping techniques (e.g., R*-tree [4], clustering [24, 15], Hilbert curve [26], *k-d* tree [5]), which aim to minimize the volumes of MBRs.

In terms of storage representation, each compact MBR $M^C_{(\tau,\ell)}$ consists of (i) a MBR, (ii) a diamond ID, for deriving its $\tau$ and $\ell$ values, and (iii) references to the contained diamond MBRs and object IDs. Fig. 10 illustrates an example of the compact MBRs. These compact MBRs serve as filters during query processing. We can still apply Eq. 13 to compute the PAA distance upper bound between the query and a compact MBR, and prune those compact MBRs that cannot contribute to any query result.

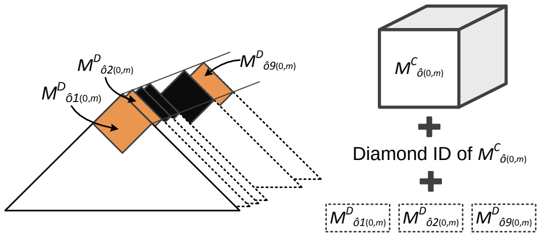


**Figure 10: Inter-object grouping**

#### 4.2.3 *Achieving $\mathcal{S}$ space*

The diamond cover index (DCI) is the collection of the compact MBRs (including containing diamond MBRs) described above. We now elaborate how to build DCI such that its space is bounded by a given space budget $\mathcal{S}$. By using Lemma 4, the total number of $\omega$-diamond MBRs for all $n$ objects is $O(nm^2/\omega^2)$. Thus, we can tune $\omega$ such that the total number of $\omega$-diamond MBRs is within the budget $\mathcal{S}$. By subtracting the above space from $\mathcal{S}$, we obtain the maximum number of compact MBRs to be created in DCI.

### 4.3 Answering $k$LCS by DCI

Alg. 3 is our proposed algorithm that utilizes the diamond cover index $\mathcal{I}^D$ to answer a $k$LCS query $q$. It follows the top-down execution paradigm like in Algorithm 2.

Given a query sequence $q$, we construct the query $\omega$-diamond MBR $M^D_{\hat{q}(\tau,\ell)}$ on-demand. For every compact MBR at $(\tau,\ell)$, we calculate the upper bound of $M^D_{\hat{q}(\tau,\ell)}$ and $M^C_{(\tau,\ell)}$ by Eq. 13. If the upper bound is above threshold $\delta$, then we refine every object in $M^C_{(\tau,\ell)}$ by the function $\omega$-SKIP (see Line 9). $\omega$-SKIP is a variant of SKIP (Alg. 2) which searches only the $\omega$-diamond covering area starting at $(\tau,\ell)$. In addition, $\omega$-SKIP terminates when it discovers the first result or the search length is shorter than the $k$th element in the seen result set $R$.

We add the result $o_r$ into the $R$ if it is not covered by any element in $R$ and is longer than the $k$th result in $R$. We terminate the entire search process and return $R$ as the result when there is no more diamond having subsequences longer than the $k$th element in $R$.

**Algorithm 3** SKIP+DCI algorithm

```
    Algorithm SKIP+DCI( Query $q$, Database $O$, Threshold $\delta$, Maximum
    sequence length $m$, Result size $k$, Skip factor $\alpha$, DCI index $B^c$ )
 1: calculate $S^1_q$ and $S^1_{q^2}$
 2: calculate $S^\alpha_{o_i}$, $S^\alpha_{o_i^2}$, and $S^\alpha_{qo_i}$ for every $o_i \in O$
 3: for all $\ell \in \{m, m-\omega, m-2\omega, ..., 0\}$ do              ▷ from $m$ to 0
 4:     for all $\tau \in \{0, \omega, 2\omega, ..., (m-\ell)\}$ do  ▷ pruning phase at level $\ell$
 5:         construct $M^D_{\hat{q}(\tau,\ell)}$ by the method discussed in Sec. 4.2.1
 6:         for all $M^C_{\hat{o}(\tau,\ell)} \in \mathcal{I}^D_{(\tau,\ell)}$ do
 7:             if $UB(M^D_{\hat{q}(\tau,\ell)}, M^C_{\hat{o}(\tau,\ell)}, \phi) \geq \delta$ then          ▷ see Eq. 13
 8:                 for all $o \in M^C_{\hat{o}(\tau,\ell)}$ do     ▷ refinement phase at level $\ell$
 9:                     $o_r := \omega$-SKIP($q, o, \delta, \alpha, \tau, \ell, R$)
10:                     if $r \not\triangleright o_r, \forall r \in R$ and $r.\ell < o_r.\ell, \exists r \in R$ then
11:                         insert $o_r$ into $R$ or replace $k$th element by $o_r$
12:     if $\ell \leq k$th result length in $R$ then
13:         return $R$ as result
```

**Discussion.** Answering $k$LCS by DCI is efficient since it applies diamond batch pruning to save subsequence distance calculations. Whenever the pruning step works (i.e., $UB(\cdots) < \delta$ at line 7), it saves $|M^C| \cdot \omega^2$ distance calculations.

### 4.4 Extensions of DCI

**Subsequence similarity queries.** Besides $k$LCS, DCI also supports subsequence matching queries [12] of arbitrary lengths. Given a sequence query $q$ of length $\ell$ and a correlation threshold, we verify and prune corresponding compact MBRs based on Eq. 13. As a remark, only those elements in unpruned compact MBRs need to be refined by typical distance calculations [33]. Our index is effective to process this kind of queries since each query diamond MBR $M_{\hat{q}}$ includes subsequences having the same length $\ell$ (instead of having different lengths) such that it gives a tight upper bound to Eq. 13.

**Parallel programming.** Our framework easily supports parallel programming. At index construction, we partition the objects into a number of pieces based on the available processor cores of the query execution machine. An diamond cover index (DCI) is constructed for each piece of objects. At query processing, we simply execute SKIP+DCI on these diamond cover indices in parallel. Note that we can terminate the search of an index when the $k$th result (being checked by message passing or by shared memory) is already longer than the search length.

## 5. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the performance of our proposed methods. All methods are implemented in C++, and evaluated on a machine running Ubuntu 10.10 with Intel Xeon 4-Cores (8-Threads) E5620 2.4GHz, 4GBytes of main memory, and 4TBytes hard drive. We use the following datasets to evaluate our proposed methods.

**RAND.** This synthetic dataset is generated by a random walk technique as suggested in [1]. For each sequence $o$, we randomly generate the first value (i.e., $o[0]$) in $[-1.0, 1.0]$. Each subsequent value of $o$ is generated by $o[i+1] = o[i]*(1+N(\mu,\sigma))$, where $N(\mu,\sigma)$ is a normal distribution function. By default, we set the mean $\mu = 0$ and the standard deviation $\sigma = 0.2$.

**STOCK.** The stock dataset is collected from Yahoo Finance [11]. We extract daily closing prices for 2187 quoted companies in NYSE from 2008 to 2012, where each sequence is of length 1,037.

**TAO.** The Tropocal Atmosphere Ocean (TAO) dataset is collected from the international Tropical Ocean Global Atmosphere (TOGA) program [12]. It contains sea surface temperatures of 55 array sensors over several years. We extract weekly subsequences of each array. The dataset consists of 28,399 sequences in total, where each sequence contains 1,008 values.

Table 1 shows the ranges of the investigated parameters and their default values (in bold). In each experiment, we vary a single parameter, while setting the others to their default values.

**Table 1: Range of parameter values**

| Type | Parameter | Dataset | Values |
|---|---|---|---|
| Index | Stop length ratio, $\ell_{stop}/m$ | - | 5%, **10%**, 15%, 20% |
| | PAA dimensionality, $\phi$ | - | 6, 8, **10**, 12, 14 |
| | Diamond side length, $\omega$ (ID cost ratio) | - | 45(∼5%), **33(∼10%)**, 23(∼20%), 19(∼30%), 17(∼40%) |
| | Index size ratio, $\mathcal{S}/(mn)$ | - | 25%, 50%, **100%**, 200% |
| | Alignment | - | **aligned**, non-aligned |
| Data | Cardinality, $n$ | RAND | 25k, 50k, **100k**, 200k |
| | | STOCK | **2k**, 4k, 6k, 8k, 10k |
| | | TAO | **28399** |
| | Sequence length, $m$ | RAND | 250, **500**, 750, 1000 |
| | | STOCK | **1037** |
| | | TAO | **1008** |
| Query | Correlation threshold, $\delta$ | - | 0.91, 0.93, **0.95**, 0.97 |
| | Skip ratio, $\alpha/m$ | - | 0.2%, 5%, **10%**, 15%, 20% |
| | $k$ | - | **1**, 2, 4, 8, 16 |

In all experiments, a workload of 100 queries is used to evaluate the performance, where the queries are generated based on the distribution of the (synthetic) dataset or randomly picked from the (real) dataset. In this work, we focus on the average response time of the query workload. The raw data and indices (if applicable) are pre-loaded in main memory for handling plenty of queries in the workload. To measure the exact response time, we assume only one single core available in the system unless explicitly stated.

## 5.1 Robustness of Diamond Cover Index

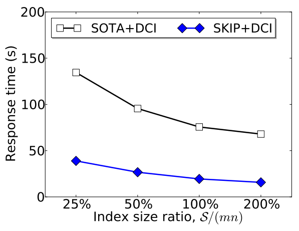
**Index Construction Performance.** DCI construction includes two steps: intra-object grouping (Sec. 4.2.1) and inter-object grouping (Sec. 4.2.2). Under the default setting, the diamond side length $\omega$ is set to 33. It takes 1.5 hours to construct all diamond MBRs at intra-object grouping. The construction speed can be easily boosted up multiple times by using parallel programming (e.g., OpenMP).

We have evaluated different grouping methods for implementing inter-object grouping. We pick Hilbert curve as our default grouping method since it offers fast computation and reasonable grouping performance. For instance, the inter-grouping just takes 66.9 seconds using Hilbert curve under the default settings.

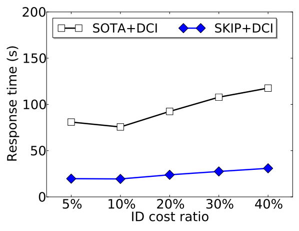
**Effect of Index Parameters.** We proceed to study the effect of various parameters on the response time of our index DCI. First, we show the effect of index size constraint $\mathcal{S}$ in Fig. 11(a). Not surprisingly, the response time reduces as $\mathcal{S}$ becomes larger. Here, we set the space constraint to $mn$ (i.e., $\mathcal{S}/(mn) = 100\%$) since it offers a good tradeoff between the storage size and query performance. When $\mathcal{S} = 50\%$, the response time of SOTA+DCI and SKIP+DCI increases by only 26.3% and 37.3%, respectively. This indicates that the index is still effective when the systems has limited resources.

[11] http://finance.yahoo.com/

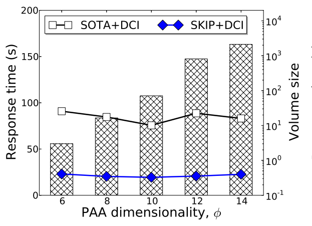
[12] http://www.pmel.noaa.gov/tao/



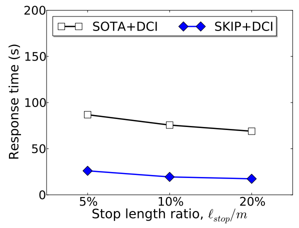
(a) Effect of index size ratio



(b) Effect of side length, $\omega$



(c) Effect of PAA dim, $\phi$



(d) Effect of stop length ratio

**Figure 11: Index tuning experiments**

Next, we study the effect of diamond side length $\omega$ that controls the number of diamond MBRs being constructed. Obviously, the more diamond MBRs are constructed, the higher storage space is needed to store their IDs in compact MBRs. Fig. 11(b) shows the effect of $\omega$. When $\omega$ is set to 33, it constructs 105 diamond MBRs for each object under default settings ($\ell_{stop} = 50$). Given the space constraint $\mathcal{S} = mn$, the cost of storing the diamond MBR IDs (ID Cost) is around 10% of $\mathcal{S}$; in other words, it leaves 90% space for constructing compact MBRs. We pick $\omega = 33$ (ID cost ∼10%) as our default setting since it gives the best overall performance.

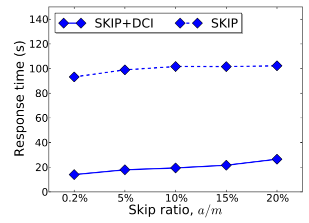
Fig. 11(c) shows the response time (by lines) and average MBR volume (by bars) of PAA dimensionality $\phi$. The value of $\phi$ gives a tradeoff between the number and the volume of the compact MBRs. For instance, when $\phi$ is small, there are more compact MBRs being constructed under the same index size constraint $\mathcal{S}$. However, the volume of each compact MBR becomes looser since every element is represented by fewer dimensions. Typically, the range of $\phi$ is from 8 to 12. According to our experiments, we pick $\phi = 10$ as our default setting as it achieved the best overall performance.

According to our study, only a few queries return short length result. For instance, there is no result being shorter than 20% of $m$ under the default settings. In addition, analysts may not be interested in those short result due to lack of representativeness. Thereby, we omit those diamond MBRs whose subsequence length is shorter than a stop length, $\ell_{stop}$ (an index parameter). Note that this does not affect the correctness of our algorithms since they can still execute non-index methods below the length $\ell_{stop}$. Fig. 11(d) shows the effect of $\ell_{stop}$. We pick 10% of the total length as our default setting since it gives a good tradeoff between the efficiency and the applicability of the index.

In summary, while $\ell_{stop}$ is more likely specified by domain application experts, $\mathcal{S}$ is set based on available resources, and the value of $\phi$ can be tuned according to previous work [18, 35]. The only tunable parameter is the diamond side length $\omega$, which controls between the intra and inter grouping. We recommend to use 10% of the space overhead to store IDs of diamond MBRs.

**Effect of $\alpha$.** Fig. 12(a) shows the effect of $\alpha$, where the response time of both SKIP and SKIP+DCI grows slowly with $\alpha$. This offers room for tuning $\alpha$ under different resource available in the system.

**Effectiveness of alignment.** In Sec. 4.2.1, we discuss a padding


(a) Effect of skip ratio, $\alpha/m$

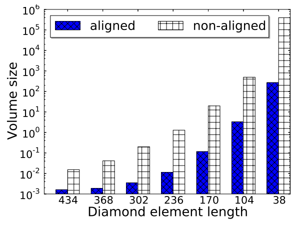
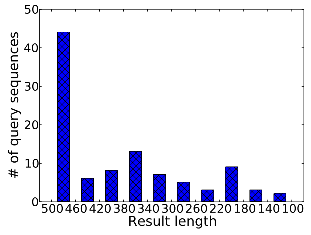

(b) Aligned vs Non-aligned

**Figure 12: The effect of proposed techniques**

zero technique that aligns the elements in a $\omega$-diamond MBR. Fig. 12(b) demonstrates the effectiveness of the alignment. The average volume size of non-aligned MBRs is 1 to 2 orders of magnitude larger than the aligned MBRs at different element lengths.

## 5.2 Performance Studies


(a) Result distribution

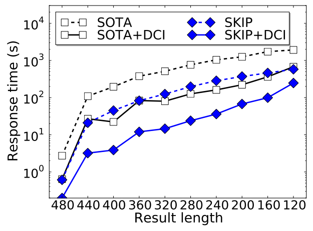

(b) Average response time

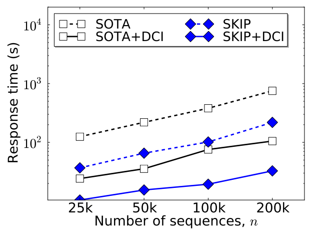
**Figure 13: Performance at different result lengths on RAND**

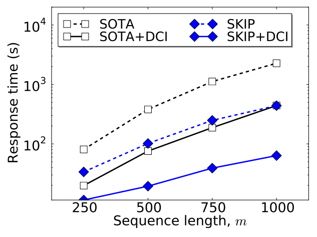
**Performance Overview.** Before evaluating the effect of various scalability parameters, we give a performance overview of four proposed methods, SOTA (Sec. 3.1), SKIP (Sec. 3.2), SOTA+DCI (Sec. 4.3), and SKIP+DCI (Sec. 4.3). Fig. 13 shows the distribution of result lengths and the average response time for each result length, under default parameter settings on RAND dataset. In Fig. 13(a), the total sum of all bars equals 100 (query workload size), and the width of each bar is 40. SOTA and SKIP are 5.01 and 5.23 times on average slower than their index versions, SOTA+DCI and SKIP+DCI, respectively. Especially for the most difficult queries (i.e., shortest result lengths), SKIP+DCI is faster than the state-of-the-art adaption (SOTA) by 19.58 times. As a remark, the state-of-the-art techniques [33] (applied in SOTA) were already 16.24 times faster than previous distance calculation on ECG data [6].

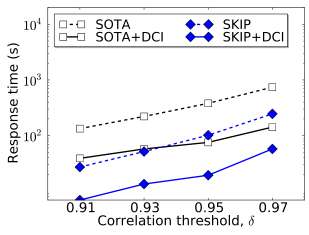
Regarding the memory usage, SOTA and SKIP consume 417.65 MBytes ($\sim$the raw data) and 451.54 MBytes ($\sim$raw data plus $\alpha$-skipping arrays), respectively. All DCI methods require 381.47 MBytes (=the raw data) extra memory to store the index.

**Scalability Experiments.** Fig. 14(a) shows the response time of the methods as a function of cardinality $n$, after setting all other parameters to their default values. Cost grows linearly with $n$ for all methods. SKIP is 3.48 times faster than SOTA on average. The index version of SKIP is 4.65 times faster than the non-index version on average. Our best method, SKIP+DCI, remains superior to other methods at difficult cases (largest cardinality).

Fig. 14(b) shows the response time of the methods versus sequence length $m$. Cost grows exponentially with $m$ for all methods. SKIP (SKIP+DCI) is at least 2.39 (1.75) times faster than SOTA (SOTA+DCI). Note that SKIP is almost as efficient as the index


(a) Effect of cardinality, $n$


(b) Effect of sequence length, $m$


(c) Effect of $\delta$

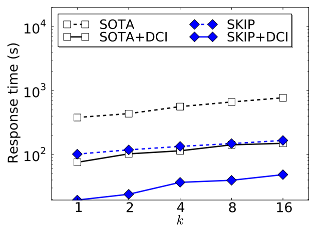
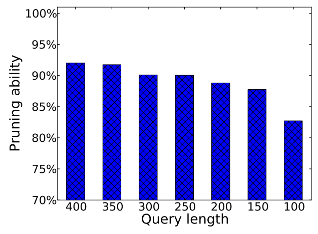

(d) Effect of $k$

**Figure 14: Scalability experiments on RAND**

version of SOTA when $m$ is large. The superiority of SKIP+DCI becomes more obvious when $m$ increases.

The response time of different correlation thresholds $\delta$ is shown in Fig. 14(c). As we aim to find longest-lasting highly correlated subsequences, some weak correlation thresholds are discarded such as $0.8, 0.85$. And one can expect to discover the longest-lasting correlated subsequences in its whole length using these weak correlation thresholds among massive sequence database. All methods are sensitive to $\delta$ since $k$LCS may search shorter subsequences as $\delta$ increases. The performance gap between SOTA+DCI and SKIP+DCI becomes closer at $\delta = 0.97$ since the reordering early abandon technique may work well when $\delta$ is high. As a remark, increasing $\delta$ does not necessarily reveal better result as it may return a set of short and non-representative LCS result.

Fig. 14(d) shows the response time versus $k$. The response time increases linearly with $k$, as more results are discovered. Our SKIP+DCI still performs the best, followed by SKIP/SOTA+DCI, and SOTA.


(a) Sequence matching queries

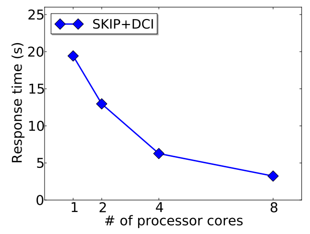
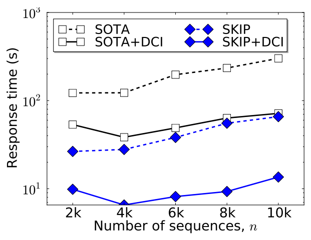

(b) $k$LCS in parallel

**Figure 15: Additional experiments on RAND**

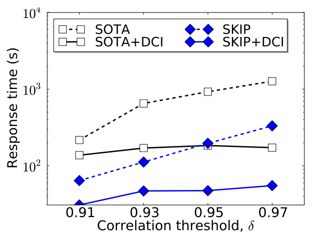
**Additional Experiments.** Fig. 15(a) demonstrates the *pruning ability* of subsequence matching queries on DCI (discussed in Sec. 4.4), where it reports the ratio of pruned objects at threshold 0.95. The result is very impressive since DCI can prune 88.9% of objects on average among different query lengths. This also reveals the potential of applying DCI for various similarity queries.

Fig. 15(b) shows the response time of SKIP+DCI versus the number of processor cores. Based on the number of cores available in the system, we partition the data into the same number of pieces and build an DCI for each piece of data. The result shows that the response time is boosted up proportional to the number of cores used. When we process $k$LCS by 8 cores, the response time is reduced to 3.23 seconds.

**Real dataset experiments.** To evaluate the stock data in a more meaningful way, we simulate more stocks by merging several real stock segments together. In Fig. 16(a), it shows the response time of the methods as a function of the stock data in different cardinalities. Cost grows linearly with $n$ for all methods when $n \geq 4k$. Regarding the case of $n = 2k$, the index methods are slower than the cases, $n = 4k$ and $n = 6k$, since the inter-object grouping is not effective at very low cardinalities.



(a) Effect of $n$ on STOCK



(b) Effect of $\delta$ on TAO

**Figure 16: Real data experiments**

Fig. 16(b) shows the response time of the methods as a function of threshold, $\delta$, on TAO dataset. The trend is similar to that of Fig. 14(c). SOTA+DCI is faster at $\delta = 0.97$ than $\delta = 0.95$ since the reordering early abandon technique works well. However, it is still 2.93 times slower than SKIP+DCI. We add a note that the effect of different scalability parameters on the real datasets is similar to those of Fig. 14 in our internal testings. We omit the experiments due to the space limitation.

## 6. RELATED WORK

Indexing and mining sequence data has received plenty of attention in database and data mining community in the last two decades, such as similarity search and subsequence matching [2, 3, 4, 7, 12, 13, 17, 30, 33, 35], classification and clustering [21, 27], motif discovery [28], prominent streak and anomaly detection [8, 16], etc. However, to the best of our knowledge, there is no existing technique on efficiently discovering the longest-lasting correlated subsequence (LCS) in sequence databases.

In order to discover $k$LCS efficiently, our method requires a multi-length index to support batch pruning for subsequence queries of different lengths. A related work called Multi-Resolution Index (MRI) [17] deals with arbitrary length queries more efficiently than I-adaptive [12] which employs prefix search to support longer query. MRI is a collection of I-adaptive indexes at different based-2 resolutions. At query time, hierarchical prefix search can be applied on MRI. However, as we mentioned in Section 2.2, prefix search can only work for non-normalized distance measures. Rakthanmannon et al. [33] proposed an non-index method to boost up the normalized arbitrary length subsequence search for both Euclidean distance and Dynamic Time Warping (DTW) using the *Z-normalization early abandonment* and *reordering early abandon*. A main drawback of the non-index solutions is that they do not support batch pruning. If a query $q$ is not similar to a subsequence, then $q$ is unlikely similar to its neighbor subsequences.

Other related works include StatStream [38], Mueen et al. [29] and BRAID [34]. StatStream focuses on efficiently finding highest correlated sequence pairs within a finite window by using DFT to maintain a grid data structure. Mueen et al. consider a similar problem to StatStream, but solve it by approximation. It used the graph partitioning to optimize I/O cost and the DFT approximation to optimize the CPU cost while providing error guarantees. BRAID solved the problem of finding lag correlation among multiple streams. Given $k$ co-evolving sequences, at any point of time, BRAID attempts to report all sequence pairs with a lag correlation as well as their lags.

Most of the above methods either only support fixed length query or only work under non-normalized distance metrics. In summary, there is no known technique to support longest-lasting correlated queries efficiently.

## 7. CONCLUSION

In this paper, we propose a novel technique which can efficiently discover longest-lasting correlated subsequences in sequence database. Our basic idea is to derive a tight correlation bound for subsequences of similar length and offset. The correlation upper bound can be calculated prior to accessing the actual subsequence, thus achieving significant reduction of the candidate subsequences. Two core techniques, *$\alpha$-skipping technique* and *diamond cover index*, are proposed for this purpose and they are evaluated thoroughly in this paper. Given reasonable space overhead, our best approach is up to one order of magnitude faster than the state-of-the-art adaption. As future work, we plan to extend our methods to support different kinds of correlation queries.

## Acknowledgement

This project was supported by grant MYRG109(Y1-L3)-FST12-ULH from University of Macau Research Committee and grant PolyU 5302/12E from Hong Kong RGC.

## 8. REFERENCES

[1] Monte Carlo simulated stock price generator. `http://25yearsofprogramming.com/blog/20070412c-montecarlostockprices.htm`.

[2] I. Assent, R. Krieger, F. Afschari, and T. Seidl. The ts-tree: efficient time series search and retrieval. In *EDBT*, pages 252–263, 2008.

[3] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos. Approximate embedding-based subsequence matching of time series. In *SIGMOD*, pages 365–378, 2008.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD*, pages 322–331. ACM Press, 1990.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[6] T. Bragge, M. Tarvainen, and P. A. Karjalainen. High-resolution qrs detection algorithm for sparsely sampled ecg recordings. univ. of kuopio, dept. of applied physics report., 2004.

[7] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh. isax 2.0: Indexing and mining one billion time series. In *ICDM*, pages 58–67, 2010.

[8] V. Chandola, V. Mithal, and V. Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *ICDM*, pages 743–748, 2008.

[9] C.-I. Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Plenum Publishing Co., 2003.

[10] R. Cole, D. Shasha, and X. Zhao. Fast window correlations over uncooperative time series. In *KDD*, pages 743–749, 2005.

[11] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.

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

[13] R. F. S. Filho, A. J. M. Traina, C. T. Jr., and C. Faloutsos. Similarity search without tears: The omni family of all-purpose access methods. In *ICDE*, pages 623–630, 2001.

[14] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[16] X. Jiang, C. Li, P. Luo, M. Wang, and Y. Yu. Prominent streak discovery in sequence data. In *KDD*, pages 1280–1288, 2011.

[17] T. Kahveci and A. K. Singh. Optimizing similarity search for arbitrary length time series queries. *IEEE TKDE*, 16(4):418–433, 2004.

[18] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.

[19] E. J. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov.*, 7(4):349–371, 2003.

[20] E. J. Keogh, L. Wei, X. Xi, M. Vlachos, S.-H. Lee, and P. Protopapas. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *VLDB J.*, 18(3):611–630, 2009.

[21] T. W. Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[22] S.-H. Lim, H. Park, and S.-W. Kim. Using multiple indexes for efficient subsequence matching in time-series databases. *Inf. Sci.*, 177(24):5691–5706, 2007.

[23] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, 2007.

[24] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297. U. of Calif. Press, 1967.

[25] C. Meek, J. M. Patel, and S. Kasetty. Oasis: An online and accurate technique for local-alignment searches on biological sequences. In *VLDB*, pages 910–921, 2003.

[26] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE TKDE*, 13(1):124–141, 2001.

[27] A. Mueen, E. J. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In *KDD*, pages 1154–1162, 2011.

[28] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover. Exact discovery of time series motifs. In *SDM*, pages 473–484, 2009.

[29] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time-series data. In *SIGMOD*, pages 171–182, 2010.

[30] P. Papapetrou, V. Athitsos, M. Potamias, G. Kollios, and D. Gunopulos. Embedding-based subsequence matching in time-series databases. *ACM TODS*, 36(3):17, 2011.

[31] P. Patel, E. J. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *ICDM*, pages 370–377, 2002.

[32] D. Rafiei. On similarity-based queries for time series data. In *ICDE*, pages 410–417, 1999.

[33] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012.

[34] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *SIGMOD*, pages 599–610, 2005.

[35] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB*, pages 385–394. Morgan Kaufmann, 2000.

[36] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, 1993.

[37] H. Zhu, G. Kollios, and V. Athitsos. A generic framework for efficient and effective subsequence retrieval. *PVLDB*, 5(11):1579–1590, 2012.

[38] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002.

## APPENDIX

**Lemma 1. Proof.** For the sake of brevity, we only prove that the normalized $p$-norm subsequence distance $\hat{d}_p(q, o, \tau, \ell)$ is not monotone decreasing for $\ell$. The other cases (including monotone increasing for $\ell$ or $\tau$ and monotone decreasing for $\tau$) can be proved similarly. Suppose the normalized $p$-norm subsequence distance is monotone decreasing for $\ell$, then it must hold

$$(\sum_{i=1}^{\ell} |\hat{q}(\tau,\ell)[i] - \hat{o}(\tau,\ell)[i]|^p)^{1/p} \geq (\sum_{i=1}^{\ell'} |\hat{q}(\tau,\ell')[i] - \hat{o}(\tau,\ell')[i]|^p)^{1/p}$$

where $\ell > \ell'$. Suppose that $q(\tau, \ell\text{-}1)$ is a strict monotone increasing sequence, i.e., $\forall i, q(\tau,\ell)[i-1] < q(\tau,\ell)[i]$, and $o(\tau, \ell\text{-}1)$ is a strict monotone decreasing sequence, i.e., $\forall i, o(\tau,\ell)[i-1] > o(\tau,\ell)[i]$. We must have

$$(\sum_{i=1}^{\ell-1} |\hat{q}(\tau,\ell-1)[i] - \hat{o}(\tau,\ell-1)[i]|^p)^{1/p} > 0 \qquad (14)$$

Next, we set the $\ell$-th value of two subsequence, $q(\tau,\ell)[\ell]$ and $o(\tau,\ell)[\ell]$, to infinite. This makes the difference of their normalized form at every position be close to the limit of 0.

$$\lim_{q(\tau,\ell)[\ell]=o(\tau,\ell)[\ell]=\infty} \hat{q}(\tau,\ell)[i] - \hat{o}(\tau,\ell)[i] = 0$$

where $1 \leq i \leq \ell$. Accordingly, we have

$$\lim_{q(\tau,\ell)[\ell]=o(\tau,\ell)[\ell]=\infty} (\sum_{i=1}^{\ell} |\hat{q}(\tau,\ell)[i] - \hat{o}(\tau,\ell)[i]|^p)^{1/p} = 0 \qquad (15)$$

This is in clear contradiction to our assumption (i.e., Eq. 15$\not\geq$ Eq. 14). Therefore, the normalized p-norm subsequence distance is not monotone decreasing for $\ell$.

**Lemma 2. Proof.** First, we prove that computing $S_{\mathcal{X}}[u+i]$ or $S_{\mathcal{X}}[u-i]$ takes $O(i)$ time by giving $S_{\mathcal{X}}[u]$ and the raw data of $\mathcal{X}$. According to Eq. 6, $S_{\mathcal{X}}[u+i] = S_{\mathcal{X}}[u] + \sum_{j=u+1}^{u+i} \mathcal{X}[j]$ and $S_{\mathcal{X}}[u-i] = S_{\mathcal{X}}[u] - \sum_{j=u-i}^{u-1} \mathcal{X}[j]$. It is clear that computing $\sum_{j=u+1}^{u+i} \mathcal{X}[j]$ or $\sum_{j=u-i}^{u-1} \mathcal{X}[j]$ takes $O(i)$ time. Thereby, we can compute $S_{\mathcal{X}}[u+i]$ or $S_{\mathcal{X}}[u-i]$ in $O(i)$ time. In $\alpha$-skipping arrays, the maximum value of $i$ is $\alpha$. Accordingly, computing $\rho(q, o, \tau, \ell)$ can be done in $O(\alpha)$ time by Eq. 1.

**Lemma 3. Proof.** Based on Eq. 9, we have

$$\begin{aligned} d_{PAA}(e_{\hat{q}}, e_{\hat{o}}, \phi) &= \sqrt{\ell/\phi} \cdot d_2(e_{\hat{q}}, e_{\hat{o}}, \phi) \\ &\geq \sqrt{\ell/\ell_{min}} \sqrt{\ell_{min}/\phi} \cdot d_2^{min}(M_{\hat{q}}, M_{\hat{o}}, \phi) \\ &\geq \sqrt{\ell/\ell_{min}} \cdot d_{PAA}(M_{\hat{q}}, M_{\hat{o}}, \phi) \text{ (by Eq. 11)} \end{aligned}$$

**Lemma 4. Proof.** The number of diamond MBRs per object sequence $o$ is

$$\begin{aligned} 1 + 2 + ... + m/\omega = \frac{(m/\omega)(m/\omega+1)}{2} &= \frac{1}{\omega^2}\frac{m(m+1/\omega)}{2} \\ \leq \frac{1}{\omega^2}\frac{m(m+1)}{2} & \qquad \text{(assuming } \omega \geq 1) \end{aligned}$$

**$\omega$-Diamond Construction.** Given two cumulative arrays (i.e., $S_o$ and $S_{o^2}$), $e_{\hat{o}(\tau,\ell)}[i]$ can be computed in $O(1)$ time by combining Eq. 2 and the following equation.

$$e_{\hat{o}(\tau,\ell)}[i] = \frac{\phi}{\ell} \frac{(S_o[\frac{\ell}{\phi}(i+1)-1] - S_o[\frac{\ell}{\phi} \cdot i] + o[\frac{\ell}{\phi} \cdot i]) - \frac{\ell}{\phi}\mu_{o(\tau,\ell)}}{\sigma_{o(\tau,\ell)}}$$

where $\sigma_{\mathcal{X}(\tau,\ell)}$ is calculated by Eq. 5 and $\mu_{\mathcal{X}(\tau,\ell)}$ is calculated by Eq. 4 in $O(1)$ time.