Indexable PLA for Efficient Similarity Search

Qiuxia Chen Lei Chen Xiang Lian Yunhao Liu
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{chen, leichen, xlian, liu}@cse.ust.hk

Jeffrey Xu Yu
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Hong Kong, China
yu@se.cuhk.edu.hk

ABSTRACT

Similarity-based search over time-series databases has been a hot research topic for a long history, which is widely used in many applications, including multimedia retrieval, data mining, web search and retrieval, and so on. However, due to high dimensionality (i.e. length) of the time series, the similarity search over directly indexed time series usually encounters a serious problem, known as the "dimensionality curse". Thus, many dimensionality reduction techniques are proposed to break such curse by reducing the dimensionality of time series. Among all the proposed methods, only *Piecewise Linear Approximation* (PLA) does not have indexing mechanisms to support similarity queries, which prevents it from efficiently searching over very large time-series databases. Our initial studies on the effectiveness of different reduction methods, however, show that PLA performs no worse than others. Motivated by this, in this paper, we re-investigate PLA for approximating and indexing time series. Specifically, we propose a novel distance function in the reduced PLA-space, and prove that this function indeed results in a lower bound of the Euclidean distance between the original time series, which can lead to no *false dismissals* during the similarity search. As a second step, we develop an effective approach to index these lower bounds to improve the search efficiency. Our extensive experiments over a wide spectrum of real and synthetic data sets have demonstrated the efficiency and effectiveness of PLA together with the newly proposed lower bound distance, in terms of both *pruning power* and *wall clock time*, compared with two state-of-the-art reduction methods, *Adaptive Piecewise Constant Approximation* (APCA) and *Chebyshev Polynomials* (CP).

1. INTRODUCTION

The retrieval of similar time series has been studied ever since early 1990s [1, 9, 2], and this area remains as a hot research topic even today due to its wide usage in many new applications, including network traffic analysis [8], sensor network monitoring [31, 28], moving object tracking [7], and financial data analysis [30, 26]. For example, in a coal mine application [28], sensors are deployed in the mine collecting data such as temperature and density of oxygen, which can be modeled as time series. Since emergency events (e.g. low density of oxygen or fire alarm) usually correspond to some specific patterns (also in the form of time series), the event detection can be considered as the pattern search over time series data, which highly demands fast retrieval to keep the safety of miners. In this paper, we revisit the *similarity search* problem that obtains time series in a time-series database similar to a given query time series, especially in the case where the total number of time series in the database is large and each time series is long (i.e. with high dimensionality). In brief, given a time-series database $\mathcal{D}$ and a query time series Q, a *similarity query* retrieves those time series $S \in \mathcal{D}$ such that $dist(Q, S) \leq \varepsilon$, where $dist(\cdot, \cdot)$ is a distance function between two time series and ε a similarity threshold.

Since the length of time series is usually very long (e.g. ≥ 1024), it becomes infeasible to directly index time series using spatial indexes, such as R-tree [10]. This is because of the serious "dimensionality curse" problem in high dimensional space. Specifically, when the dimensionality becomes very high, the query performance of the similarity search using a multidimensional index can be even worse than that of a *linear scan*. In order to solve this problem, Faloutous et al. [9] presented a general framework, called GEMINI. In particular, GEMINI reduces the original time series into a lower dimensional space (reduced space) using a dimensionality reduction technique, maintains data in the reduced space with a multidimensional index (e.g. R-tree [10]), and ensures that the efficiency of the similarity search can be achieved while not introducing *false dismissals* (actual answers that are however not in the final result). The proposed dimensionality reduction techniques include *Singular Value Decomposition* (SVD) [13, 18], *Discrete Fourier Transform* (DFT) [24], *Discrete Wavelet Transform* (DWT) [5, 23, 12, 27], *Piecewise Linear Approximation* (PLA) [20, 17], *Piecewise Aggregate Approximation* (PAA) [15, 29], *Adaptive Piecewise Constant Approximation* (APCA) [16] and *Chebyshev Polynomials* (CP) [4].

We list seven popular dimensionality reduction techniques in Table 1, in terms of the time complexity, space complexity, and capability to be indexed in the reduced space, where n is the length of each time series, N is the total number of time series in the database, and $(2m)$ is the reduced dimensionality. In terms of the time complexity, CP is more costly than PLA, DWT, PAA and APCA (Table 1); and PLA is

VLDB '07, September 23-28, 2007, Vienna, Austria.

Techniques	Time	Space	Indexable
PLA	$O(n)$	$O(n)$	No
DFT	$O(n \cdot log(n))$	$O(n)$	Yes
DWT	$O(n)$	$O(n)$	Yes
SVD	$O(N \cdot n^2)$	$O(N \cdot n)$	Yes
PAA	$O(n)$	$O(n)$	Yes
APCA	$O(n)$	$O(n)$	Yes
CP	$O((2m) \cdot n)$	$O(n)$	Yes

Table 1: A Comparison Among Dimensionality Reduction Techniques

Techniques	MMD ($2m = 8$)	MMD ($2m = 16$)
PLA	1.72	1.64
DFT	2.07	2.13
DWT	2.21	1.89
APCA	2.20	1.86
CP	1.84	1.76

Table 2: Minimum Maximum Deviation (MMD)

much lower than that of CP. However, in terms of the *pruning power* (or the tightness of lower bound distances), the existing experimental results show that APCA outperforms DFT, DWT, and PAA [29]; and CP is better than APCA [4].

It is interesting to note that PLA is the only one, out of the seven, that does not have an *indexable* lower bound distance function. This non-indexability, for PLA, is mainly due to the difficulties of designing a lower bound distance function in the reduced PLA-space [17], but not due to PLA itself. Therefore, currently, the only feasible way to perform a similarity search based on PLA is the *linear scan*, which incurs the scalability problem in very large databases. Note that, there are reported studies to compare existing indexable approaches [23, 16, 4], in terms of the pruning power, but no study to compare PLA with other methods, due to its non-indexability.

In this paper, we concentrate ourselves on investigating the pruning power of PLA and designing an indexable lower bound distance function for PLA. In the sequel, we briefly justify our investigation by showing the advantages of PLA in terms of the *minimum maximum deviation* (MMD) [4] and *reconstruction accuracy* [22]. Figure 1 plots the opening stock prices of a Fortune500 company, called `ALCOA`, during the period from February 28th, 1978 to October 24th, 2003 (totally 6,480 days). We also draw the approximation curves of this stock series in Figure 1 using three different methods APCA, CP, and PLA, where the reduced dimensionality (denoted as $2m$) is set to 8 and y-axis is normalized to $[-2, 2.5]$. Since it is hard to compare the three methods just by the naked eye, we consider the measure MMD with results presented in Table 2, where m is the number of segments used in PLA or APCA, and $2m$ is the reduced dimensionality (2 dimensions for each segment in PLA or APCA). PLA is superior to DFT, DWT, APCA, and CP, in terms of MMD. Table 3 illustrates the average reconstruction accuracies [22] of PLA, DFT, DWT, APCA, and CP. In particular, we applied each of these dimensionality reduction methods to 24 *benchmark* data sets [30, 32, 6, 7], and reduced the dimensionality of each time series from 256 to 16. We reconstruct the original series from the reduced data, and calculate the reconstruction accuracy. PLA achieves similar results to CP while outperforming others. Based on studies of Table 2 and Table 3, PLA shows high potential to be used as an effective

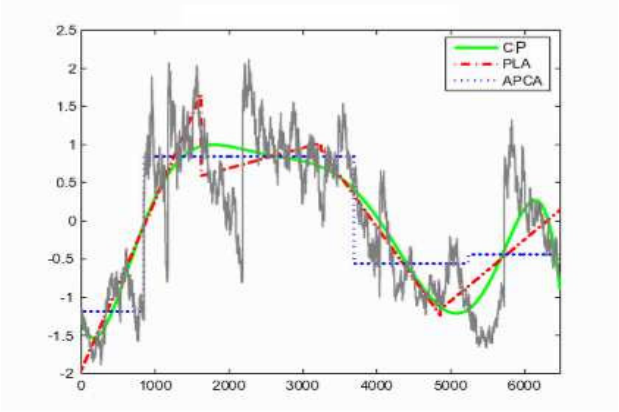


Figure 1: Opening Stock Prices (`ALCOA`)

dimensionality reduction tool, which motivates us to find an *indexable* lower bound distance for PLA to answer similarity queries.

Techniques	PLA	DFT	DWT	APCA	CP
Accuracy	88.4%	38.1%	80.2%	86.4%	89%

Table 3: The Reconstruction Accuracy (24 Benchmarks)

The main contributions of this paper are summarized below.

- We propose a new indexable lower bound distance function for PLA, denoted as $dist_{PLA}(\cdot, \cdot)$, which calculates a lower bound of the Euclidean distance between any two time series in the reduced PLA-space.
- We give a theorem (Theorem 3.1), and prove, which is not trivial, that our proposed $dist_{PLA}(\cdot, \cdot)$ is a lower bound distance function, which guarantees that it does not introduce any false dismissals during the similarity search through a PLA index.
- We present a new minimum distance function between a PLA query point and a *minimum bounding rectangle* (MBR) containing a set of PLA data points in the R-tree [10], which can be used as a basis to index PLA.
- We conduct extensive experimental studies, and compare the efficiency and effectiveness of PLA with those of two state-of-the-art dimensionality reduction techniques, APCA and CP. Our experimental results confirm that PLA outperforms the other two, in terms of the *pruning power* and *wall clock time*, over all the tested data sets.

The rest of the paper is organized as follows. Section 2 reviews previous works on similarity search. Section 3 gives our new lower bound distance function for PLA with a proof of its correctness. Section 4 illustrates the computational issue of the minimum distance between a PLA query point and an MBR in a PLA index. Section 5 discusses how to support the kNN search over a PLA index. Section 6 demonstrates the experimental results, comparing PLA with APCA and CP. Finally, we conclude and give some future research directions in Section 7.

2. SIMILARITY SEARCH

Many studies on similarity search over time-series databases have been conducted in the past decade. The pioneering work by Agrawal et al. [1] used Euclidean distance as the similarity measure, *Discrete Fourier Transform* (DFT) as the dimensionality reduction tool, and R-tree [10] as the underlying search index. Faloutsos et al. [9] later extended this work to allow the *subsequence matching* and proposed the GEMINI framework for indexing time series. In particular, GEMINI converts each time series into a lower dimensional point by applying any dimensionality reduction technique, and uses a lower bound of the actual Euclidean distance between time series to perform the similarity search, which can guarantee no-false-dismissals while filtering through the index. The subsequent work focused on two major aspects: new dimensionality reduction techniques (assuming that Euclidean distance is the underlying measure) and new approaches to measure the similarity between two time series.

Existing dimensionality reduction techniques include SVD [13, 18], DFT [24], DWT [5, 23, 12, 27], PLA [20, 17], PAA [15, 29], APCA [16], and CP [4]. These methods first reduce the dimensionality of each time series to a lower dimensional space, and then apply a new metric distance function to measure the similarity between any two transformed (reduced) data. Note that, in order to guarantee no-false-dismissals during the similarity search, this metric distance function must satisfy the *lower bounding lemma* [9], that is, the distance between two transformed data in the reduced space should be a lower bound of their actual Euclidean distance in the original space.

Among all the reduction methods, SVD is accurate, however, costly, in terms of both computation and space costs, since eigenvectors are required to be calculated and extra space is needed for the storage of large matrices. Furthermore, APCA and CP are the two state-of-the-art reduction approaches, proposed by Keogh et al. [16] and Cai and Ng [4], respectively. In particular, APCA [16] divides the time series into disjoint segments of different lengths and takes the mean within each segment. Thus, each segment can be represented by two reduced coefficients, the mean value and the length of the segment. CP [4] obtains coefficients of *Chebyshev polynomials* which are used as the reduced data. From the previous study by Keogh et al. [16], APCA outperforms DFT, DWT, and PAA, in terms of the pruning power by orders of magnitude. Moreover, CP is claimed to be better than APCA [16], however, incurs more computation cost than DWT, PAA and APCA.

To the best of our knowledge, previous work on the similarity search by PLA without false dismissals is the L-index [20], which however cannot be indexed and requires a linear scan. Another work by Wu et al. [26] applied PLA to approximate time series, however, they defined their own distance function, specific to the financial application, rather than the Euclidean distance which is the focus of this paper. In fact, for different distance measures (functions), such as L_p-norm [29], *Dynamic Time Warping* (DTW) [3], *Edit distance with Real Penalty* (ERP) [6], and *Edit distance with Real Sequence* (EDR) [7], different lower bounds are designed for the similarity search [6, 4, 7, 26, 25, 19, 21]. Recently, Lee et al. [19] defined a weighted distance measure using three kinds of distances, *perpendicular*, *parallel*, and *angle distances*, between sub-trajectories. Moose and Patel [21] proposed Swale as a similarity measure used in the presence of noise, time shifts, and data scaling. Thus, it is quite interesting to investigate the similarity search with other distance measures and we would leave it as one of our future work.

3. PLA: SIMILARITY SEARCH

In this work, we focus on dimensionality reduction techniques that do not introduce false dismissals while filtering through the index. Moreover, our target query is *k nearest neighbor query* (kNN), which returns k time series in a time-series database that have the smallest distances to a given query time series. Inspired by our initial studies (shown in Section 1), that PLA should behave as well as the state-of-the-art techniques, such as APCA and CP, we re-investigate PLA as a reduction method for an efficient similarity search without false dismissals over time-series databases. Note that, the proposal of PLA is not our contribution, whereas the definition of an indexable lower bound distance function on PLA and the method to index this lower bound distance are our major focus of the work. Furthermore, in this paper, we study the *whole matching* [1] with Euclidean distance where time series in the database and query time series are of the same length. However, our work can be easily extended to the *subsequence matching* [9] where query time series are allowed to have different lengths or other useful distance measures such as L_p-norms ($1 \leq p \leq +\infty$) (by relaxing the search radius [29]).

3.1 Piecewise Linear Approximation (PLA)

In time-series databases, each time series S consists of an ordered sequence of values, formally, $S = \langle s_1, s_2, \ldots, s_n \rangle$, where n is the length of time series S. In this paper, we consider the Euclidean distance (i.e. L_2-norm), which has been widely used in many applications such as the (sub)sequence matching [1, 9]. Specifically, given two time series $S = \langle s_1, \ldots, s_n \rangle$ and $Q = \langle q_1, \ldots, q_n \rangle$ of length n, the Euclidean distance $dist(S, Q)$ between S and Q is given by:

$$dist(S, Q) = \sqrt{\sum_{i=1}^{n} (s_i - q_i)^2}. \quad (1)$$

As an approximation technique, *Piecewise Linear Approximation* (PLA) [20, 17] approximates a time series with line segments. Given a sequence $S = \langle s_1, \ldots, s_n \rangle$ of length n, PLA can use one *line segment*, $s'_t = a \cdot t + b$ ($t \in [1, n]$), to approximate S, where a and b are two coefficients in a linear function such that the *reconstruction error*, $RecErr(S)$, of S is minimized. Formally, $RecErr(S)$ is defined by the Euclidean distance between the approximated and actual time series (Eq. (2)).

$$RecErr(S) = \sqrt{\sum_{t=1}^{n} (s_t - s'_t)^2} = \sqrt{\sum_{t=1}^{n} (s_t - (a \cdot t + b))^2} \quad (2)$$

where two parameters a and b satisfy the following two conditions:

$$\frac{\partial RecErr(S)}{\partial a} = 0 \quad (3)$$

$$\frac{\partial RecErr(S)}{\partial b} = 0 \quad (4)$$

Notations	Descriptions
S	a time series $\langle s_1, \ldots, s_n \rangle$
N	total number of time series
a and b	two coefficients in the function of linear curve
n	the length of time series
m	the number of segments in PLA or APCA
l	the length of each segment ($= \lceil \frac{n}{m} \rceil$)
$dist(\cdot,\cdot)$	Euclidean distance between two time series
$dist_{PLA}(\cdot,\cdot)$	the distance between two reduced PLA series

Table 4: Meanings of Notations

Here, a and b can be obtained by solving both Eq. (3) and Eq. (4). In particular, we have:

$$a = \frac{12\sum_{t=1}^{n}(t - \frac{n+1}{2})s_t}{n(n+1)(n-1)} \quad (5)$$

$$b = \frac{6\sum_{t=1}^{n}(t - \frac{2n+1}{3})s_t}{n(1-n)} \quad (6)$$

where s_t is the actual value at timestamp t in time series S. The line segment $s'_t = a \cdot t + b$ can well approximate time series, S, since the two parameters, a and b, are selected so as to achieve the minimum reconstruction error, $RecErr(S)$.

However, although an approximation (reduction) method with low reconstruction error can closely mimic a time series, it does not necessarily imply an efficient similarity search through the index with data reduced by this method. The key factor that affects the query performance (in terms of the pruning power) is the tightness of the lower bound distance defined in the reduced space, compared to the real Euclidean distance between two time series in the original space. For example, the same dimensionality reduction technique with different definitions of lower bound distance function would result in quite different pruning powers, due to different tightness of the lower bounds. In general, the tighter the lower bound distance is, the higher the pruning power is.

In our work, we consider approximating time series with multiple line segments (instead of one). Specifically, given a time series $S = \langle s_1, \ldots, s_n \rangle$, we divide S into m *disjoint* segments of equal length l (i.e. $n = m \cdot l$), separately approximate each segment with a (best fitted) line segment (as mentioned before) and finally obtain two coefficients a_i and b_i of the linear function from the i-th line segment (for $1 \le i \le m$), based on Eq. (5) and Eq. (6). Therefore, the PLA representation S_{PLA} of a time series S is given as follows:

$$S_{PLA} = \langle a_1, b_1; a_2, b_2; \ldots; a_m, b_m \rangle \quad (7)$$

where the time complexity of computing S_{PLA} is $O(n)$.

We adopt the PLA representation that divides time series into segments of equal length, and leave the interesting case where time series are partitioned into segments of different lengths as our future work. Table 4 summarizes the commonly-used notations in this paper.

3.2 The PLA Lower Bound Distance

In this subsection, we propose a novel PLA lower bound distance function, $dist_{PLA}(\cdot,\cdot)$, in the reduced PLA-space, whose correctness will be proved in the next subsection. In brief, after we reduce the raw time series into a lower dimensional PLA-space, we identify a lower bound distance function in the reduced space, whose inputs are two reduced data after the PLA reduction and output is a lower bound of the real Euclidean distance between two time series in the original space.

Our main idea behind is to use the distances between pairs of PLA line segments as the output of the lower bound distance function. Figure 2 (a) illustrates an example of PLA lower bound distance, which approximates the top (bottom) time series by two PLA line segments E_1 and E_2 (F_1 and F_2). The dotted lines between E_1 and F_1 (E_2 and F_2) indicate the distances between them at different timestamps, which are the ones we use to define the new lower bound distance. In fact, when we use PLA to approximate segments of the time series, each segment can be represented by a PLA line segment and "virtually" shifted to the origin of the space along the temporal (horizontal) axis. This is because changing the computation order of pairs of matching points would not affect the final value of Euclidean distance. Figure 2 (b) illustrates the second pair of PLA line segments E_2 and F_2 in Figure 2 (a), which is shifted to the origin. The reason for our zooming in the second pair, E_2 and F_2, is to show that, every time we calculate the distance from a pair of PLA line segments, the starting index always begins with "1" (refer to labels along the horizontal axis). In other words, if the PLA line segments E_2 and F_2 are represented by $a_{E,2} \cdot j + b_{E,2}$ and $a_{F,2} \cdot j + b_{F,2}$, respectively, then the distance between line segments E_2 and F_2 is the summation of the squared distances for all the matching points, where j starts from 1 to 16. That is, the resulting distance is $\sum_{j=1}^{16}((a_{E,2} - a_{F,2}) \cdot j + (b_{E,2} - b_{F,2}))^2$. Note that, here j starts from 1, not 17 (the original timestamp in the time series). Actually, this restarting index technique is one of the critical parts that make our new lower bound distance indexable. Another technique we use is that we divide time series into segments of equal length and approximate each segment with PLA. Note that, the previous L-index [20] divides each time series into segments of different lengths, which essentially makes the resulting lower bound distance not indexable. In contrast, our approach is different and can thus lead to an indexable lower bound distance. Now we want to find a lower bound of the Euclidean distance between the two time series shown in Figure 2 (a). In this paper, we use the summation of (squared) distances between pairs of line segments (i.e. lengths of dotted lines) as the lower bound.

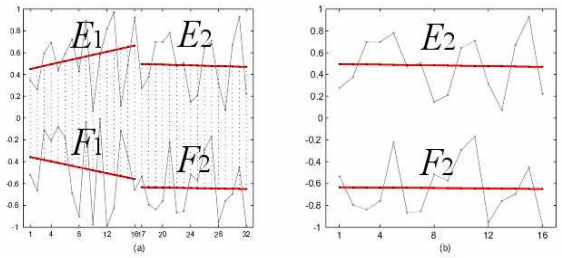


Figure 2: Segmented PLA and Their Lower Bounds

Formally, we define our PLA lower bound distance function $dist_{PLA}(S, Q)$ as follows:

Definition 3.1: Given two time series $S = \langle s_1, s_2, \ldots, s_n \rangle$ and $Q = \langle q_1, q_2, \ldots, q_n \rangle$, we divide each of them into m segments of equal length l ($= \lceil \frac{n}{m} \rceil$). Let $S_{PLA} = \langle a_{11}, b_{11}; \ldots; a_{1m}, b_{1m} \rangle$ and $Q_{PLA} = \langle a_{21}, b_{21}; \ldots; a_{2m}, b_{2m} \rangle$ be the

two PLA representations of S and Q, respectively. The lower bound distance function $dist_{PLA}(S,Q)$ between S_{PLA} and Q_{PLA} is defined by:

$$dist_{PLA}(S,Q) = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{l}(a_{3i}\cdot j + b_{3i})^2}, \quad (8)$$

where $a_{3i} = a_{1i} - a_{2i}$ and $b_{3i} = b_{1i} - b_{2i}$ for $1 \le i \le m$.

As in Eq. (8), $\sum_{j=1}^{l}(a_{3i}\cdot j + b_{3i})^2$ is exactly the summed (squared) distances between the i-th pair of line segments from series S and Q. As in the previous example where $m = 2$ and $l = 16$, the lower bound distance is given by summing up the (squared) lower bound distances for both segments.

By expanding Eq. (8) in Definition 3.1, we have:

$$\begin{aligned} & dist^2_{PLA}(S,Q) \quad (9)\\ = & \sum_{i=1}^{m}\left(\frac{l(l+1)(2l+1)}{6}a_{3i}^2 + l(l+1)a_{3i}b_{3i} + lb_{3i}^2\right). \end{aligned}$$

Furthermore, from Eq. (5) and Eq. (6), a_{1i}, b_{1i}, a_{2i}, and b_{2i} in Definition 3.1 can be calculated by the following formulas:

$$a_{1i} = \frac{12\sum_{j=(i-1)l+1}^{il}(j-(i-1)l-\frac{l+1}{2})s_j}{l(l+1)(l-1)}, \quad (10)$$

$$b_{1i} = \frac{6\sum_{j=(i-1)l+1}^{il}(j-(i-1)l-\frac{2l+1}{3})s_j}{l(1-l)}, \quad (11)$$

$$a_{2i} = \frac{12\sum_{j=(i-1)l+1}^{il}(j-(i-1)l-\frac{l+1}{2})q_j}{l(l+1)(l-1)}, \quad (12)$$

$$b_{2i} = \frac{6\sum_{j=(i-1)l+1}^{il}(j-(i-1)l-\frac{2l+1}{3})q_j}{l(1-l)}. \quad (13)$$

In the next step, we would prove that our proposed distance function $dist_{PLA}(S,Q)$ in Eq. (9) indeed results in a lower bound of the real Euclidean distance between time series S and Q, whose details will be presented in the next subsection.

3.3 Correctness

In order to guarantee no-false-dismissals during the similarity search, the distance $dist_{PLA}(S,Q)$ (defined in Definition 3.1) between any two PLA representations S_{PLA} and Q_{PLA} should satisfy the *lower bounding lemma*, that is:

Theorem 3.1: (Lower Bounding Lemma for PLA) Given two time series S and Q, assume $dist_{PLA}(S,Q)$ is the PLA distance given in Definition 3.1, and $dist(S,Q)$ is the Euclidean distance between two time series S and Q. The similarity search over the indexed PLA data can guarantee no-false-dismissals, if it holds that:

$$dist^2_{PLA}(S,Q) \le dist^2(S,Q). \quad (14)$$

Proof Sketch: In the sequel, we give the proof of the *lower bounding lemma* for PLA. Let $x_i = s_i - q_i$. From Eq. (1), we have $dist(S,Q) = \sqrt{\sum_{i=1}^{n} x_i^2}$. After dividing the time series into m segments, the (squared) Euclidean distance between S and Q can be rewritten as:

$$dist^2(S,Q) = \sum_{i=1}^{m}\sum_{j=(i-1)\cdot l+1}^{i\cdot l} x_j^2. \quad (15)$$

According to Definition 3.1 and Eq. (10) - Eq. (13), the (squared) lower bound distance function $dist^2_{PLA}(S,Q)$ can be rewritten as:

$$\begin{aligned} & dist^2_{PLA}(S,Q)\\ = & \sum_{i=1}^{m}\Big(\frac{l(l+1)(2l+1)}{6}\cdot(a_{1i}-a_{2i})^2 + l\cdot(l+1)\\ & \cdot(a_{1i}-a_{2i})\cdot(b_{1i}-b_{2i}) + l\cdot(b_{1i}-b_{2i})^2\Big)\\ = & \sum_{i=1}^{m}\left(\frac{12}{l(l-1)(l+1)}\left(\sum_{j=(i-1)l+1}^{il}(j-(i-1)\cdot l)\cdot x_j\right)^2\right.\\ & -\frac{12}{l(l-1)}\left(\sum_{j=(i-1)l+1}^{il}(j-(i-1)\cdot l)\cdot x_j\right)\cdot\sum_{j=(i-1)l+1}^{il}(x_j)\\ & \left.+\frac{2(2l+1)}{l(l-1)}\cdot\left(\sum_{j=(i-1)l+1}^{il}(x_j)\right)^2\right). \quad (16) \end{aligned}$$

From the forms of Eq. (15) and Eq. (16), it is obvious that the two (squared) distances $dist^2(S,Q)$ and $dist^2_{PLA}(S,Q)$ are the summarization of the (squared) distances from all segments. Therefore, it is sufficient to give the proof on one segment. In other words, we only need to prove that the lower bound distance on the first segment in the reduced PLA-space is smaller than or equal to the real Euclidean distance on the same segment. In particular, we define the (squared) Euclidean distance on the first segment as:

$$dist^2(S^{(1)},Q^{(1)}) = \sum_{i=1}^{l} x_i^2. \quad (17)$$

The PLA distance on the first segment is given by:

$$\begin{aligned} & dist^2_{PLA}(S^{(1)},Q^{(1)})\\ = & \frac{12}{l(l-1)(l+1)}\left(\sum_{i=1}^{l} ix_i\right)^2 - \frac{12}{l(l-1)}\left(\sum_{i=1}^{l} ix_i\right)\left(\sum_{i=1}^{l} x_i\right)\\ & +\frac{2(2l+1)}{l(l-1)}\left(\sum_{i=1}^{l} x_i\right)^2. \quad (18) \end{aligned}$$

Thus, we need to prove $dist^2_{PLA}(S^{(1)}, Q^{(1)}) \le dist^2(S^{(1)}, Q^{(1)})$. For simple illustration, we denote terms in Eq. (17) and Eq. (18) using four variables f_l, g_l, c_l, and d_l, with respect to l. That is,

$$\begin{aligned} f_l &= \frac{12}{l(l-1)(l+1)}\left(\sum_{i=1}^{l} ix_i\right)^2\\ g_l &= \frac{12}{l(l-1)}\left(\sum_{i=1}^{l} ix_i\right)\left(\sum_{i=1}^{l} x_i\right)\\ c_l &= \frac{2(2l+1)}{l(l-1)}\left(\sum_{i=1}^{l} x_i\right)^2\\ d_l &= \sum_{i=1}^{l} x_i^2. \quad (19) \end{aligned}$$

Therefore, it is sufficient to prove the following simplified inequality:

$$f_l + c_l \le g_l + d_l. \quad (20)$$

Without loss of generality, let $z_l = g_l + d_l - (f_l + c_l)$. When $l = 2$, $z_2 = 6(x_1 + 2x_2)(x_1 + x_2) + (x_1^2 + x_2^2) - 2(x_1 + 2x_2)^2 - 5(x_1 + x_2)^2 = 0$. It holds that:

$$z_l = \quad z_2 + \sum_{i=3}^{l}(z_i - z_{i-1}) \tag{21}$$

When $l \geq 3$, we denote w_l as follows:

$$w_l = \frac{1}{4}(l+1)l(l-1)(l-2)(z_l - z_{l-1}), \tag{22}$$

where

$$\begin{aligned} w_3 &= 2(g_3 + d_3 - f_3 - c_3 - g_2 - d_2 + f_2 + c_2) \\ &= 4(x_1 + 2x_2 + 3x_3)(x_1 + x_2 + x_3) + 2(x_1^2 + x_2^2 + x_3^2) \\ &\quad -(x_1 + 2x_2 + 3x_3)^2 - \frac{14}{3}(x_1 + x_2 + x_3)^2 \\ &\quad +12(x_1 + 2x_2)(x_1 + x_2) + 2(x_1^2 + x_2^2 + x_3^2) \\ &\quad -4(x_1 + 2x_2)^2 - 10(x_1 + x_2)^2 \\ &= (x_1 - 2x_2 + x_3)^2. \end{aligned} \tag{23}$$

Similarly, we can obtain:

$$\begin{aligned} w_4 &= (2x_1 - x_2 - 4x_3 + 3x_4)^2, \\ w_5 &= (3x_1 - 0x_2 - 3x_3 - 6x_4 + 6x_5)^2, \\ w_6 &= (4x_1 + x_2 - 2x_3 - 5x_4 - 8x_5 + 10x_6)^2, \\ &\ldots \end{aligned}$$

From the equalities above, it is clear that coefficients of x_i ($i \in [1, l]$) for each w_l follow certain rules. For example, each time the coefficient of x_1 is increased by 1 (i.e. $1, 2, 3, 4, 5, 6, \ldots$); that of x_2 is also increased by 1 (i.e. $-2, -1, 0, 1, 2, 3, \ldots$); that of x_3 is increased by 1 except for the first one (i.e. $1, -4, -3, -2, -1, 0, \ldots$); that of x_4 is increased by 1 except for the first two (i.e. $0, 3, -6, -5, -4, -3, \ldots$); that of x_5 is increased by 1 except for the first three (i.e. $0, 0, 6, -8, -7, -6, \ldots$). Therefore, we propose our hypothesis in the following formula:

$$w_l = \left(\frac{1}{2}(l-1)(l-2)x_l + \sum_{i=1}^{l-1}(l + 1 - 3i)x_i\right)^2 \tag{24}$$

Eq. (24) can be proved by the mathematical induction, which is omitted here due to the space limit. The basic idea, however, is to compare the coefficients of terms $x_i x_j$ ($i, j \in [1, l]$) on the right hand side of Eq. (22) with those in Eq. (24). In this way, we can show that every corresponding coefficients of term $x_i x_j$ are equal to each other. As a result, we can prove that $w_l \geq 0$ from Eq. (24). Furthermore, by Eq. (22), we have $z_l \geq z_{l-1}$ ($l \geq 3$). Therefore, based on Eq. (21), we can infer that $z_l \geq 0$ when $l \geq 3$, which completes our proof of Inequality (20).

Since Inequality (20) is equivalent to $dist^2(S^{(1)}, Q^{(1)}) - dist^2_{PLA}(S^{(1)}, Q^{(1)}) \geq 0$, that is, *lower bounding lemma* holds on the first PLA segment, we complete our proof of the *lower bounding lemma* for PLA on the entire time series (since its (squared) lower bound distance is the summation of those on all segments), that is, $dist^2_{PLA}(S, Q) \leq dist^2(S, Q)$. In other words, by using the PLA lower bound distance function $dist_{PLA}(S, Q)$, the similarity search over indexes can guarantee no-false-dismissals. □

4. INDEXING PLA

We have proved in Section 3 that the lower bound distance of PLA follows the *lower bounding lemma*, that is, the distance between any two reduced PLA data is never greater than that between the original time series. This nice property of PLA can guarantee no-false-dismissals when performing the similarity search in the reduced PLA-space, which also confirms the feasibility of our indexing mechanism with PLA (i.e. the GEMINI framework [9]). In the sequel, we discuss indexing the reduced PLA data to speed up the retrieval efficiency of the similarity search.

Recall that, PLA divides each time series S into m disjoint segments of equal size l, and for each (e.g. i-th) segment, approximates it using a (best fitted) line segment with two coefficients a_i and b_i. Thus, each time series S is transformed to totally $2m$ coefficients in the order of $\langle a_1, b_1; a_2, b_2; ...; a_m, b_m\rangle$, which can be treated as a $2m$-dimensional point in the reduced space. Next, we insert each resulting point into a $2m$-dimensional index structure, such as R-tree [10], on which the similarity search can be efficiently processed. Here, since the index construction is the same as the standard R-tree [10], each entry of nodes in the R-tree consists of an MBR containing the reduced PLA data and a pointer pointing to its corresponding subtree.

It is important to note that, different from the traditional similarity search in the R-tree that uses the Euclidean distance as the lower bound distance function in the reduced space [1, 9], the PLA index applies the lower bound distance function given in Eq. (8).

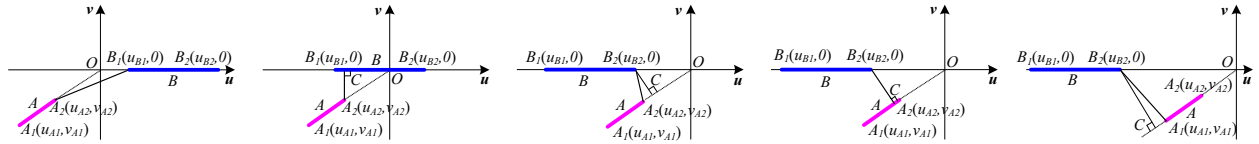
Below, in this subsection, we discuss details of computing the minimum PLA lower bound distance $mindist_{PLA}(q, e)$ from a reduced PLA query point q to an MBR node e (bounding the reduced PLA data). Formally, given a query point q and node MBR e in the PLA-space, the minimum (squared) distance between q and e is given by $mindist^2_{PLA}(q, e) = min_{\forall x \in e}\{dist^2_{PLA}(q, x)\}$, where $dist^2_{PLA}(q, x)$ is the (squared) PLA lower bound distance between q and p.

Assume that the $2m$-dimensional PLA query point q has coordinates in the order of $\langle q_{a_1}, q_{b_1}; q_{a_2}, q_{b_2}; ...; q_{a_m}, q_{b_m}\rangle$, and any point x in MBR e_i $\langle x_{a_1}, x_{b_1}; x_{a_2}, x_{b_2}; ...; x_{a_m}, x_{b_m}\rangle$. According to Definition 3.1, the (squared) PLA lower bound distance $dist^2_{PLA}(q, x)$ between a query point q and any point x in e is given by:

$$\begin{aligned} dist^2_{PLA}(q, x) &= \sum_{i=1}^{m}\Big(\frac{l(l+1)(2l+1)}{6}(q_{a_i} - x_{a_i})^2 \\ &\quad +l(l+1)(q_{a_i} - x_{a_i})(q_{b_i} - x_{b_i}) \\ &\quad +l(q_{b_i} - x_{b_i})^2\Big). \end{aligned} \tag{25}$$

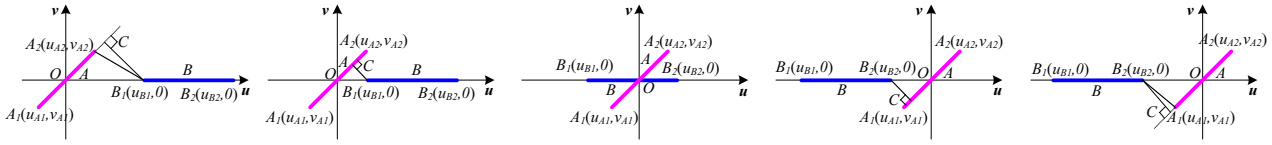
Therefore, our goal is now to find the minimum value of $dist^2_{PLA}(q, x)$ for all points $x \in e$. Note that, both points q and x contain $2m$ coordinates each, which independently come from m disjoint segments. Thus, when computing the minimum distance from query to MBR (with PLA lower bound distance), without loss of generality, we consider each segment individually and then sum up the result from each segment to obtain the overall minimum distance. In particular, our subgoal is to find the minimum possible value of $dist^2_{PLA}(q^{(i)}, x^{(i)})$ for the i-th segment, with points $x \in e$, where $dist^2_{PLA}(q^{(i)}, x^{(i)})$ is given by:

$$\begin{aligned} dist^2_{PLA}(q^{(i)}, x^{(i)}) &= \frac{l(l+1)(2l+1)}{6}(q_{a_i} - x_{a_i})^2 \\ &\quad +l(l+1)(q_{a_i} - x_{a_i})(q_{b_i} - x_{b_i}) \\ &\quad +l(q_{b_i} - x_{b_i})^2. \end{aligned} \tag{26}$$



(a) Case 1.1 (b) Case 1.2 (c) Case 1.3 (d) Case 1.4 (e) Case 1.5

Figure 3: Case 1 (Line Segment A_1A_2 is Completely Contained in the Third Quadrant)



(a) Case 2.1 (b) Case 2.2 (c) Case 2.3 (d) Case 2.4 (e) Case 2.5

Figure 4: Case 2 (Line Segment A_1A_2 is Partially Contained in the First and Third Quadrants)

For simplicity, we ignore the notation i on the RHS of Eq. (26), which can be rewritten as:

$$\begin{aligned} dist^2_{PLA}(q^{(i)}, x^{(i)}) &= \frac{l(l+1)(2l+1)}{6}(q_a - x_a)^2 \\ &\quad + l(l+1)(q_a - x_a)(q_b - x_b) \\ &\quad + l(q_b - x_b)^2 \\ &= \left(\sqrt{l}\frac{l+1}{2}(x_a - q_a) - \sqrt{l}(-x_b + q_b)\right)^2 \\ &\quad + \sqrt{\frac{l^3 - l}{12}}(x_a - q_a)^2 \end{aligned} \quad (27)$$

where $x_a \in [a_{min}, a_{max}]$, $x_b \in [b_{min}, b_{max}]$, and interval $[a_{min}, a_{max}]$ ($[b_{min}, b_{max}]$) is the boundary of MBR e along the $(2i)$-th ($(2i+1)$-st) dimension.

Note that, Eq. (27) is a very complex formula which involves two variables x_a and x_b within intervals $[a_{min}, a_{max}]$ and $[b_{min}, b_{max}]$, respectively. Thus, it is not trivial to calculate the minimum value of $dist^2_{PLA}(q^{(i)}, x^{(i)})$ from Eq. (27). In the sequel, we illustrate the key step to obtain this minimum value.

Interestingly, our problem of obtaining the minimum possible value of $dist^2_{PLA}(q^{(i)}, x^{(i)})$ can be reduced to finding the minimum distance between two line segments in a 2-dimensional space. In particular, we have:

$$dist^2_{PLA}(q^{(i)}, x^{(i)}) = (u_A - u_B)^2 + (v_A - v_B)^2 \quad (28)$$

where

$$u_A = \sqrt{l} \cdot \frac{l+1}{2} \cdot (x_a - q_a), \quad (29)$$

$$u_B = \sqrt{l} \cdot (-x_b + q_b), \quad (30)$$

$$v_A = \sqrt{\frac{l^3 - l}{12}} \cdot (x_a - q_a), \text{ and} \quad (31)$$

$$v_B = 0. \quad (32)$$

Observe that, Eq. (28) is very similar to the Euclidean distance function between two points in a 2-dimensional space. If we denote $A(u_A, v_A)$ and $B(u_B, v_B)$ as two points in a 2-dimensional space, namely *u-v space*, we can find out that our target distance function $dist^2_{PLA}(q^{(i)}, x^{(i)})$ is exactly the squared Euclidean distance between these two points A and B. Furthermore, due to the constraints of points A and B by Eq. (29)-(32), we find that A can be any point within a line segment $v = \left(\left(\frac{l+1}{2}\right)/\sqrt{\frac{l^3-1}{12}}\right) \cdot u$, for u between $\left(\sqrt{l} \cdot \frac{l+1}{2} \cdot (a_{min} - q_a)\right)$ and $\left(\sqrt{l} \cdot \frac{l+1}{2} \cdot (a_{max} - q_a)\right)$, in the *u-v space*, and similarly B can be any point on a line segment lying on the u-axis where $v = 0$ and u is between $\left(\sqrt{l} \cdot (-b_{max} + q_b)\right)$ and $\left(\sqrt{l} \cdot (-b_{min} + q_b)\right)$. Thus, the distance $dist^2_{PLA}(q^{(i)}, x^{(i)})$ can in fact be computed from the minimum distance of two line segments in the *u-v* space! Based on this finding, we transform the original x_a-x_b space into *u-v* space according to Eq. (29)-(32).

Figure 3(a) illustrates a visual example of these two line segments A_1A_2 and B_1B_2 corresponding to points A and B, respectively. For simplicity, in the sequel, we denote $A_1(u_{A1}, v_{A1})$ as the bottom-left vertex of line segment A_1A_2 and $A_2(u_{A2}, v_{A2})$ as the top-right one. Similarly, $B_1(u_{B1}, 0)$ is the left vertex of line segment B_1B_2, whereas $B_2(u_{B2}, 0)$ is the right one. Note that, the minimum value of $dist^2_{PLA}(q^{(i)}, x^{(i)})$ in Eq. (28) is exactly the (squared) minimum distance between two line segments A_1A_2 and B_1B_2 in the *u-v space*.

In order to obtain the minimum distance between line segments A_1A_2 and B_1B_2, we study their relative positions, which can be classified into three major cases by considering the position of A_1A_2. Specifically, since line segment A_1A_2 can only fall into the first and/or third quadrants in the *u-v space*, we give three cases as follows.

- **Case 1:** Line segment A_1A_2 is completely contained in the third quadrant of the *u-v space*.
- **Case 2:** Line segment A_1A_2 is partially contained in the first and third quadrants of the *u-v space*.
- **Case 3:** Line segment A_1A_2 is completely contained

in the first quadrant of the *u-v space*.

Figure 3 illustrates Case 1, where line segment A_1A_2 is completely contained in the third quadrant of the *u-v space*. In particular, by considering the position relationship between B_1B_2 and A_1A_2, we can further divide Case 1 into five subcases, Case 1.1–Case 1.5, as shown in Figures 3(a) - 3(e), respectively, where B_1B_2 moves along the u-axis from right to left. Specifically, in Case 1.1 (Figure 3(a)), line segment B_1B_2 is to the right of point A_2 (i.e. $u_{B1} > u_{A2}$). Obviously, in this case, the minimum possible distance between line segments A_1A_2 and B_1B_2 is $|A_2B_1|$, where $|XY|$ stands for the distance between two points X and Y in the *u-v space*. For Case 1.2 in Figure 3(b), the pedal C is contained in line segment B_1B_2 if we draw a perpendicular line from point A_2 to B_1B_2. Thus, the minimum possible distance between two line segments in this subcase is $|A_2C|$. In Case 1.3 (Figure 3(c)), the pedal is not contained in line segment B_1B_2 (A_1A_2) if we draw a perpendicular line from A_2 (B_2) to B_1B_2 (A_1A_2). The minimum distance between these two line segments is obviously $|A_2B_2|$. In Case 1.4 (Figure 3(d)), when pedal C is contained in A_1A_2 by drawing a perpendicular line from B_2 to A_1A_2, the minimum possible distance is $|B_2C|$. Similarly, in Case 1.5 (Figure 3(e)), when line segment B_1B_2 further moves towards left and this pedal C is not contained in B_1B_2 any more, the minimum value becomes $|A_1B_2|$. The conditions of these five subcases are listed in Table 5.

Furthermore, Figure 4 corresponds to Case 2, where line segment A_1A_2 is partially contained in the first and third quadrants of the *u-v space*. Note that, in this case, A_1A_2 contains origin O of the *u-v space*. Similar to Case 1, based on the positions of B_1B_2 and A_1A_2, we can also divide Case 2 into five subcases (Figures 4(a) – 4(e)), in which B_1B_2 moves from right to left. In particular, in Case 2.1 (Figure 4(a)), line segment B_1B_2 is on the right of origin O and moreover the pedal C is outside line segment A_1A_2 if we draw a perpendicular line from point B_1 to A_1A_2. In this case, the minimum distance between line segments A_1A_2 and B_1B_2 is obviously $|A_2B_1|$. In Case 2.2 (Figure 4(b)) where B_1B_2 is to the right of origin O and this pedal C is contained in line segment A_1A_2, the minimum possible distance between A_1A_2 and B_1B_2 is given by $|B_1C|$. For Case 2.3 in Figure 4(c), when line segment B_1B_2 contains the origin O, the minimum distance between A_1A_2 and B_1B_2 is zero, since these two line segments intersect with each other. Similarly, in Case 2.4 (Figure 4(d)), line segment B_1B_2 is to the left of origin O and the pedal C is contained in line segment A_1A_2 if we draw a perpendicular line from point B_2 to A_1A_2. In this case, the minimum possible distance is $|B_2C|$. Finally, in Case 2.5 (Figure 4(e)) where line segment B_1B_2 moves further left and pedal C is out of line segment A_1A_2, we have the minimum distance $|A_1B_2|$. The detailed switching conditions of different subcases are summarized in Table 5.

Since Case 3, where A_1A_2 is completely contained in the first quadrant of the *u-v space*, is symmetric to Case 1 where A_1A_2 is in the third quadrant, we would not discuss it in detail.

In summary, we have tackled the problem of computing the minimum distance $mindist_{PLA}(q, e)$ between a query point q and an MBR e, under PLA lower bound distance. In particular, we reduce the problem to another one of finding the minimum distance between two line segments in a 2-dimensional space. The proposed method can efficiently compute $mindist_{PLA}(q, e)$, since only a few switching conditions are needed to check before calculating the distance.

5. KNN SEARCH

In this section, we discuss k *nearest neighbor* (kNN), processing on top of our PLA index without introducing false dismissals. PLA index can be easily extended to answer other queries, for example, *range query*.

Figure 5 illustrates the general framework for the kNN search over the PLA index. In particular, assume we have constructed a $2m$-dimensional R-tree [10] $\mathcal{I}$ on the reduced time series by PLA. Given a query time series Q of length n, a kNN query retrieves k time series in the database that are the most similar to Q. Specifically, procedure kNNSearch first transforms query time series Q to a $2m$-dimensional point q using the PLA dimensionality reduction technique (line 1), and then performs a standard kNN search in a *best-first* manner [11] with query point q over index $\mathcal{I}$ (lines 2-15).

```
Procedure kNNSearch (I, Q, k) {
  Input:  a 2m-dimensional index I, a time series query Q,
          and parameter k
  Output: k time series in I that are the most similar to Q
  (1)  obtain reduced representation q of Q and initialize kNN_list;
  (2)  initialize an empty min-heap H accepting entries
       in the form (e, key)
  (3)  insert (root(I),0) into H
  (4)  while H is not empty
  (5)    (e, key) = pop-heap(H)
  (6)    if key ≥ maxdist(Q, kNN_list)&& (|kNN_list| = k), break;
  (7)    if (e is a leaf node)
  (8)      for each point p ∈ e
  (9)        compute the real distance dist(Q, P)
             // P is the original time series of p
  (10)       update list kNN_list with time series P
  (11)   else // e is a non-leaf node
  (12)     for each entry e_i ∈ e
  (13)       if mindist_PLA(q, e_i) < maxdist(Q, kNN_list)
  (14)         insert (e_i, mindist_PLA(q, e_i)) into H
  (15) return kNN_list
}
```

Figure 5: Pseudo Code of the kNN Search

Specifically, the procedure first initializes an empty minimum heap $\mathcal{H}$ with entries in the form (e, key) (line 2), where e is the node of R-tree and key is the sorting key in heap $\mathcal{H}$. Then, it inserts the root of R-tree into heap $\mathcal{H}$ (line 3). Each time we pop out an entry (e, key) with the minimum key in $\mathcal{H}$. If node e is a leaf node, then for each point p in e, we compute the real Euclidean distance between the original time series P and Q (corresponding to p and q, respectively). Furthermore, we update a list, kNN_list, with time series P if necessary (lines 7-10), where kNN_list contains (at most) k time series we have encountered so far, which are the most similar to Q. If node e is a non-leaf node (lines 12-14), for each entry e_i in e, we add it to heap $\mathcal{H}$ only if the minimum distance $mindist_{PLA}(q, e_i)$ from query q to entry e_i is smaller than the distance $maxdist(Q, kNN\_list)$ from q to the farthest time series in list kNN_list. Procedure kNNSearch terminates either the heap is empty or the minimum key in $\mathcal{H}$ is greater than or equal to $maxdist(Q, kNN\_list)$ (in case k time series are obtained). Finally, we return k time series in the list kNN_list as the kNN result.

Cases	Switching Conditions	$mindist^2_{PLA}(q^{(i)}, e^{(i)})$
1.1	$u_{A2} < 0, u_{B1} > u_{A2}$	$\|A_2B_1\|^2$
1.2	$u_{A2} < 0, u_{B1} \leq u_{A2}, u_{B2} > u_{A2}$	$\|A_2C\|^2$
1.3	$u_{A2} < 0, u_{A1} \leq u_{B2} \leq u_{A2}, u_C \geq u_{A2}$	$\|A_2B_2\|^2$
1.4	$u_{A2} < 0, u_{B2} \leq u_{A2}, u_{A1} < u_C < u_{A2}$	$\|B_2C\|^2$
1.5	$u_{A2} < 0, u_{B2} < u_{A1}, u_C \leq u_{A1}$	$\|A_1B_2\|^2$
2.1	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B1} > 0, u_C \geq u_{A2}$	$\|A_2B_1\|^2$
2.2	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B1} > 0, u_C < u_{A2}$	$\|B_1C\|^2$
2.3	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B1} \leq 0, u_{B2} \geq 0$	0
2.4	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B2} < 0, u_C > u_{A1}$	$\|B_2C\|^2$
2.5	$u_{A1} \leq 0, u_{A2} \geq 0, u_{B2} < 0, u_C \leq u_{A1}$	$\|A_1B_2\|^2$

Table 5: Switching Conditions for Different Cases

Note that, there are two distance functions in Figure 5. One is $maxdist(Q, kNN\_list)$, which can be easily computed by the Euclidean distance between two time series. The other one $mindist_{PLA}(q, e_i)$ (underlined in line 13 of Figure 5) was traditionally defined as the minimum possible Euclidean distance between a query point q and an MBR node e_i in the R-tree. However, in the context of our PLA dimensionality reduction method, this distance function must be re-defined as the minimum PLA lower bound distance between query q and any point in e_i in the reduced PLA-space. Therefore, in line 13 of Figure 5, we use the PLA lower bound distance function $mindist_{PLA}(q, e_i)$ discussed in Section 4.

6. EXPERIMENTAL EVALUATION

In this section, we illustrate through extensive experiments the effectiveness of PLA together with our newly proposed lower bound distance function in the reduced PLA-space, in terms of the pruning power. Furthermore, we demonstrate the query performance of our PLA index over a variety of data sets, compared with two state-of-the-art reduction techniques, APCA and CP, in terms of the *wall clock time*. In our experiments of evaluating the effectiveness of PLA, we tested on 11 real data sets and 1 synthetic data set, which are summarized in Table 6, as well as 24 *benchmark* data sets. In particular, the first six data sets in the table, including *Stocks*, *ERP*, *NHL*, *Slips*, *Kungfu* and *Angle*, have been used by Cai and Ng [4], whereas the subsequent 5 data sets are selected from Keogh's CD[1] [14] and the last one, *Generated*, is randomly generated 3D trajectories [4]. Moreover, 24 *benchmark* data sets [30, 32, 6, 7] contain data from a wide spectrum of applications and with different characteristics, where each data set has 200 time series of length 256. In order to test the efficiency of query processing, we use large data sets, *sstock* and *randomwalk*, each of which contains about 50K time series of length 256. All our tested data sets are available at [*http://www.cse.ust.hk/∼leichen/datasets/vldb07/vldb07.zip*].

Throughout our experiments, we target at kNN queries and study the efficiency and effectiveness of our proposed approach. Specifically, a kNN query retrieves k time series in the database that have the smallest Euclidean distances from a given query time series, where the query time series is randomly selected from each data set. For the sake of fair comparisons, we always use m segments (2 dimensions per segment) for PLA and APCA, whereas $2m$ reduced dimensionality for CP. We implemented PLA, APCA and CP by C++, and conducted the experiments on a Pentium IV PC 3.2GHz with 512M memory. All the experimental results are averaged over 50 runs.

[1]In fact, not all the data sets in the CD are suitable for testing the pruning power, since either some data sets are very small or the length of time series is too short.

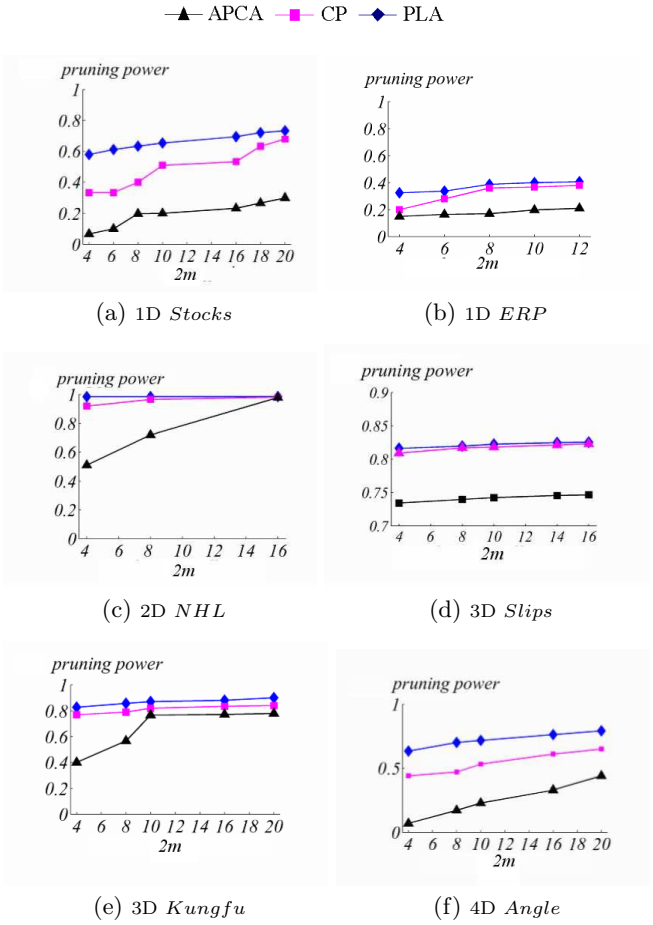
Data Sets	Dim.	Size	Length
Stocks	1	500	6480
ERP	1	496	6396
NHL	2	5000	256
Slips	3	495	400
Kungfu	3	495	640
Angle	4	657	640
Trace	1	200	278
Chlorine Concentration	1	4310	166
Mallat Technometrics	1	2400	1024
Posture	2	200	128
Muscle Activation	1	400	256
Generated	3	10000	720

Table 6: 11 Real Data Sets and 1 Synthetic Data Set

6.1 Effectiveness of PLA vs. APCA and CP

In this subsection, we evaluate the effectiveness of PLA with our proposed PLA lower bound distance over 12 tested data sets in Table 6, compared with two state-of-the-art dimensionality reduction methods, APCA and CP. Specifically, we measure the pruning power of each reduction technique during the kNN search, which is the fraction of time series that can be pruned in the reduced space. Note that, the pruning power can indicate the query performance of a reduction approach, which is free of implementation bias (e.g., page size, thresholds, etc.) [30, 6, 7]. We only report the experimental result with $k = 10$ in the sequel, because the results, when k is varied from 1 to 20, showed similar effects.

Figure 6 illustrates the pruning power of APCA, CP, and PLA over six real data sets (in [4]). Note that, for the sake of fair comparisons, in our experiments, the number, m, of segments for either PLA or APCA is always set to half that of the reduced dimensionality $2m$ in CP. For all the three methods, when we increase the reduced dimensionality, the pruning power also becomes high. Note that, although high pruning power can be achieved with very high reduced dimensionality, the query performance on high dimensional indexes is poor, compared to *linear scan*, due to the "dimensionality curse". Thus, it is the common practice to choose small value as the reduced dimensionality for fast retrieval. From figures, although APCA and CP outperform each other over different data sets, it is clear that PLA is



(a) 1D *Stocks* (b) 1D *ERP*

(c) 2D *NHL* (d) 3D *Slips*

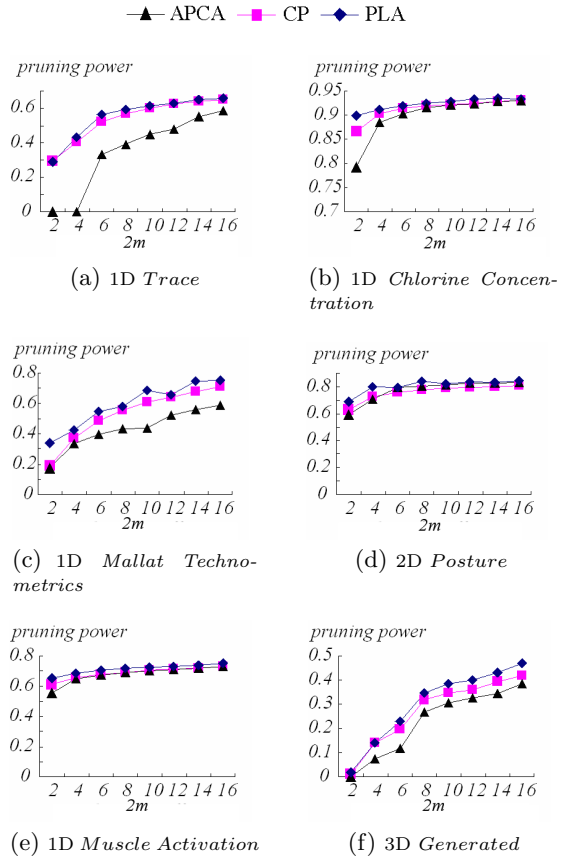
(e) 3D *Kungfu* (f) 4D *Angle*

Figure 6: Effectiveness of APCA, CP, and PLA (6 Real Data sets)

always the best, in terms of the pruning power. As an example, for 1D *Stocks* data sets, when m increases from 4 to 20, the pruning power of PLA is increased from around 57% to 75%; that of CP only from around 34% to 68%; and that of APCA is even worse (i.e. from 7% to 30%). Furthermore, even if CP uses 10 reduced dimensions or APCA 20 dimensions, PLA still has the highest pruning power with only 4 dimensions. In order to show the improvement of PLA, we consider the average pruning power with 8 reduced dimensionality over six real data sets. The average pruning power of PLA is greater than CP by 7% and APCA by 30%, which shows the potentially good query performance of PLA as a dimensionality reduction tool.

Similarly, Figure 7 illustrates the same set of experiments on the other five real data sets from Keogh's CD [14] and one synthetic data set *Generated* [4]. We can see that PLA again outperforms APCA and CP significantly, in terms of the pruning power.

Furthermore, we also conducted comparisons among APCA, CP, and PLA over 24 *benchmark* data sets as well, which covers time series in a wide spectrum of applications [30, 32, 6, 7]. From the experimental results in Figure 8, PLA is still the best among the three, in terms of the pruning power. Note that, the x-axis in figures is the No. of 24 *benchmark* data sets arranged in the alphabetical order.



(a) 1D *Trace* (b) 1D *Chlorine Concentration*

(c) 1D *Mallat Technometrics* (d) 2D *Posture*

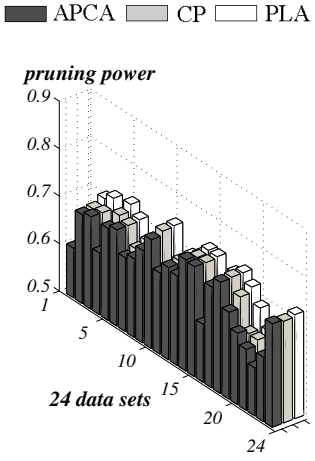
(e) 1D *Muscle Activation* (f) 3D *Generated*

Figure 7: Effectiveness of APCA, CP, and PLA (5 Real Data Sets and 1 Synthetic Data Set)

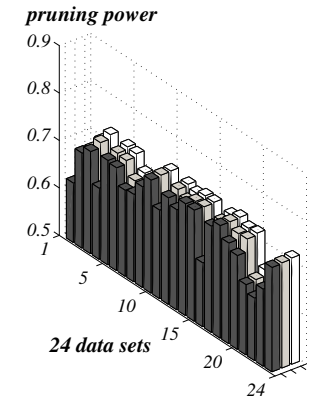
6.2 Efficiency of Query Processing

Up to now, we have compared the effectiveness of APCA, CP, and PLA, in terms of the pruning power, whose results show that PLA outperforms the other two over both real and synthetic data sets. Note, however, that high pruning power does not necessarily result in an efficient search through the index, due to the searching cost and pruning ability of the index. Therefore, as a second step, we demonstrate the query efficiency of the similarity search over the R-tree index [10], comparing APCA and CP with PLA. Specifically, we use the *wall clock time* to measure the query efficiency of the kNN search, which consists of two parts, CPU time and I/O cost, where we incorporate each page access (i.e. I/O) into the *wall clock time* by penalizing $10ms$. Since the previous 12 data sets as well as 24 *benchmark* data sets are a bit small, we use two large (50K) real and synthetic data sets, *sstock* and *randomwalk*, respectively, to evaluate the efficiency of kNN search over the constructed indexes.

Specifically, for each time series, we reduce it to a lower dimensional point with three different techniques PLA, APCA, and CP, and insert the reduced data into an R-tree [10]. Next, we randomly select time series in the database as our query time series and issue a kNN query. Here, we set the page size to 4KB. Figure 9 illustrates the *wall clock time* of the kNN search using three methods over both *sstock* and *randomwalk* data sets, by varying the value of k from 6 to 14, where the data size N is 30K and the reduced dimensionality $2m$ is 12. Note that, the vertical axis in figures

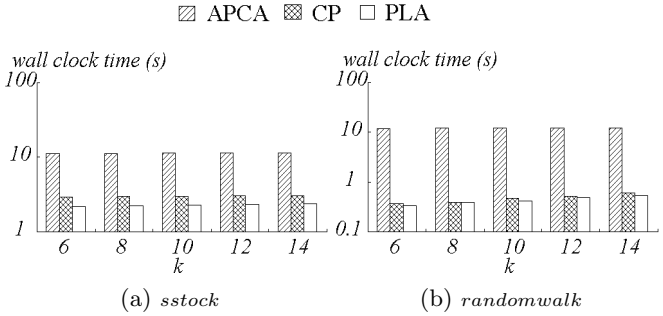


(a) *pruning power* with 4 coefficients



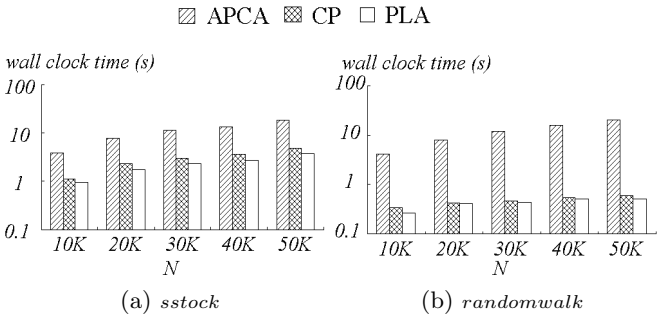
(b) *pruning power* with 8 coefficients

Figure 8: Effectiveness of APCA, CP, and PLA (24 *Benchmark* Data Sets)



(a) *sstock* (b) *randomwalk*

Figure 9: Efficiency of Query Processing (*wall clock time* vs. k)



(a) *sstock* (b) *randomwalk*

Figure 10: Scalability Test of APCA, CP, and PLA (*wall clock time* vs. N)

is in log-scale. In particular, when k increases, the *wall clock time* also increases, for all the three reduction methods APCA, CP, and PLA. From figures, PLA has the smallest *wall clock time* among all the three techniques, followed by CP. For APCA, since it has to bound segments of different lengths in MBRs of R-tree, the resulting minimum distance between a query point and an MBR node is loose. Thus, APCA incurs the worst performance. Therefore, PLA with our proposed lower bound distance function as well as the indexing approach shows much better performance than APCA and CP.

Finally, we test the scalability of PLA over both *sstock* and *randomwalk* data sets, with respect to the data size N, compared with APCA and CP, where $k = 10$ and $2m = 12$. For both data sets, when the data size N increases from 10K to 50K, the *wall clock time* of three approaches also becomes higher. However, PLA always performs better than either APCA or CP, which confirms the scalability of our proposed PLA lower bound distance and PLA indexing approach.

In summary, we have demonstrated through extensive experiments of PLA reduction technique together with our proposed lower bound distance function and indexing method, in terms of both pruning power and wall clock time for query processing, compared with two state-of-the-art techniques APCA and CP.

7. CONCLUSIONS AND FUTURE WORK

Similarity search in the time-series database encounters a serious problem in high dimensional space, known as the "curse of dimensionality". Many dimensionality reduction techniques have been proposed to break such curse and speed up the search efficiency, including two state-of-the-art reduction methods APCA and CP. Based on the initial study of comparing PLA with other indexable dimensionality reduction techniques such as DFT, DWT, PAA, APCA, and CP, PLA shows good query performance in terms of MMD and reconstruction accuracy. Motivated by this, in this paper, we re-investigate the PLA reduction technique which was previously used to approximate time series with linear segments, however, considered as "non-indexable" due to the absence of tight lower bound distance function in the reduced space. In this paper, we propose a novel distance function between any two reduced data by PLA, and prove that it is indeed a lower bound of the true Euclidean distance between two time series. Therefore, we can insert the reduced PLA data into a traditional R-tree index to facilitate the similarity search. As a second step, we propose an efficient search procedure on the resulting PLA index to answer similarity queries without introducing any false dismissals. Extensive experiments have demonstrated that PLA outperforms two state-of-the-art reduction techniques, APCA and CP, in terms of both pruning power and wall clock time.

While considering Euclidean distance as the underlying similarity measure in this paper, it would be interesting to

apply our PLA to other distance functions in the future. For example, under L_p-norm, we can use our PLA index to retrieve candidate series by increasing the search radius [29]. Moreover, this work focuses on the *whole matching* [1]. One interesting future direction is to explore the *subsequence matching* [9] with query series of variable lengths. In particular, similar to [9], we can extract sliding windows of small size, w, from time series, reduce the dimensionality of each sliding window using PLA, and construct a PLA index over the reduced data for similarity search. For any range query with a query series and a similarity threshold ε, we can divide the query into m disjoint windows of size w, reduce the dimensionality of each window to a query point with PLA, and issue m range queries over PLA index centered at m query points, respectively, with smaller radius $\frac{\varepsilon}{\sqrt{m}}$. Thus, the retrieved series are candidates of the query results. Finally, although we only discuss similarity search with PLA over *static* time-series databases, another possible future extension is to apply our proposed PLA lower bound to the search problem in streaming environment.

Acknowledgement

We thank Prof. Raymond Ng from the University of British Columbia and Prof. Eamonn Keogh from the University of California, Riverside, for providing us with data sets used in this paper. Funding for this work was provided by Hong Kong RGC Grant No. 611907, National Grand Fundamental Research 973 Program of China under Grant No. 2006CB 303000, and the NSFC Key Project Grant No. 60533110.

8. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[2] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *VLDB*, 1995.

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

[4] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD*, 2004.

[5] K. P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, 1999.

[6] L. Chen and R. Ng. On the marriage of edit distance and L_p norms. In *VLDB*, 2004.

[7] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, 2005.

[8] C. D. Cranor, T. Johnson, and O. Spatscheck. Gigascope: A stream database for network applications. In *SIGMOD*, 2003.

[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.

[10] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[11] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *TODS*, 24(2), 1999.

[12] T. Kahveci and A. Singh. Variable length queries for time series data. In *ICDE*, 2001.

[13] K. V. R. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. In *SIGMOD*, 1998.

[14] E. Keogh. The UCR time series data mining archive. riverside CA [http://www.cs.ucr.edu/∼eamonn/ tsdma/index.html]. University of California - Computer Science & Engineering Department, 2006.

[15] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *KAIS*, 3(3):263–286, 2000.

[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, 2001.

[17] E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, 1998.

[18] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, 1997.

[19] J. G. Lee, J. Han, and K. Y. Whang. Trajectory clustering: A partition-and-group framework. In *SIGMOD*, 2007.

[20] Y. Morinaka, M. Yoshikawa, T. Amagasa, and S. Uemura. The L-index: An indexing structure for efficient subsequence matching in time sequence databases. In *PAKDD*, 2001.

[21] M. Morse and J. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, 2007.

[22] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *ICDE*, 2004.

[23] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *ICDE*, 2002.

[24] D. Rafiei and A. O. Mendelzon. Efficient retrieval of similar time sequences using DFT. In *FODO*, 1998.

[25] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. J. Keogh. Indexing multidimensional time-series. *VLDBJ*, 15(1):1–20, 2006.

[26] H. Wu, B. Salzberg, and D. Zhang. Online event-driven subsequence matching over financial data streams. In *SIGMOD*, 2004.

[27] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *CIKM*, 2000.

[28] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *SIGMOD*, 2006.

[29] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. In *VLDB*, 2000.

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

[31] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD*, 2003.

[32] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD*, 2003.