IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON RELIABILITY IN SENSOR-CLOUD SYSTEMS AND APPLICATIONS (SCSA)

Received August 17, 2020, accepted September 2, 2020, date of publication September 9, 2020, date of current version September 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3022959

EPLA-DSTree: Extending Piecewise Linear Approximation on a Dynamic Segmentation Tree Index in Sensor-Cloud Systems

BIN XIE[1], QIUHONG LI[2], YANG WANG[3], AND PENG WANG[3]
[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[2]Shanghai Chang Jiang Intelligent Data Technology Company Ltd., Shanghai 201600, China
[3]Computer College of Fudan University, Shanghai 200433, China

Corresponding author: Qiuhong Li (09110240012@fudan.edu.cn)

ABSTRACT In sensor-cloud applications, a huge amount of time series are generated. Efficient similarity search approaches are necessary for processing these sensor time series data. Concerning of time series search, whole matching similarity search and subsequence similarity search are two main research focuses. In this paper, we study the whole matching similarity search problem. We propose EPLA-DSTree, which extends piecewise linear approximation on a dynamic segmentation tree index for whole matching on time series. Compared with DSTree, EPLA-DSTree improves data locality of nodes by a better time series representation. EPLA-DSTree has a tighter lower bound for nodes which leads to a better query performance. Experiments show that it has a less index building time and a better query performance. To meet the requirements of sensor-cloud applications, we present an parallel EPLA-DSTree on MapReduce, which is a popular cloud programming model.

INDEX TERMS Time series search, sensor cloud, index.

I. INTRODUCTION

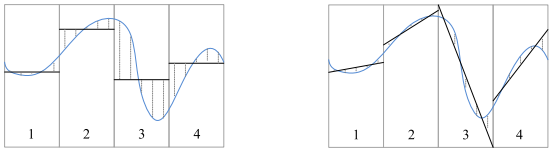
In sensor-cloud applications, a huge amount of data series are generated. Most of the data series are time relevant. For example, UCI [35] project, which collects 18000 time-series recordings from a chemical detection sensor platform at six different locations in a wind tunnel facility in response to ten high-priority chemical gaseous substances. Similarity search on time series is essential in many applications [1]–[5], [9], [15], [38], [39]. Typical application ranges across image processing [26], finance analysis [25] and environment monitoring [29], [29]. Given a set TS of time series, a query time series Q, and a distance threshold ϵ, a *similarity search* retrieves the time series $S \in TS$ such that $D(Q, S) \leq \epsilon$, where $D(\cdot, \cdot)$ is a distance function. When the Euclidean distance is used and the time series in question are assumed of the same length, the problem is called *whole matching* [1], which has been popularly used in various applications. The problem is challenging in practice, since often the set of time series TS to be searched may contain many time series and each time series may be long.

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Zakirul Alam Bhuiyan.

A time series can be regarded as a point in a multidimensional space, one dimension representing a time instant. A fundamental challenge, however, is that the length of time series is often long. A time series often contains readings at hundreds or even thousands of instants. It is highly ineffective to directly index time series using spatial indexes, such as an R-tree [7]. Obviously, the data collecting by sensors can be treated as time series data. In the paper, we study a real use case of time series similarity search problem in a bridge condition monitoring system. In this system, data was collected from about one thousand sensors of more than 20 types, such as thermometers, accelerometers, strain gauges, displacement meters, and fatigue meters. The length of each time series is 256, and one million times series were collected.

To tackle the whole matching problem, many index structures have been proposed [1]–[5], [15], which will be briefly reviewed in Section V, all those indexes are based on two fundamental principles.

To tackle this problem, many existing methods apply dimensionality reduction techniques, such as Singular Value Decomposition (SVD) [8], Discrete Fourier Transform (DFT) [1], Discrete Wavelet Transform (DWT) [4], Piecewise Linear Approximation (PLA) [5], [13], Piecewise Aggregate



(a) Approximation by APCA (b) Approximation by PLA

FIGURE 1. A comparison of APCA and PLA.

Approximation (PAA) [10], Adaptive Piecewise Constant Approximation (APCA) [11], Extended Adaptive Piecewise Constant Approximation (EAPCA) [24] and Chebyshev Polynomials (CP) [2]. After dimensionality reduction, a multidimensional index, such as R-tree [7], can be used as an index in the lower dimensional space.

Accordingly, in the state-of-the-art time series indexing methods, such as the R-tree based methods, SAX [14], iSAX [15], *all time series to be indexed are segmented in the same way*. Thus, they are *global segmentation approaches*. Those methods focus on how to approximate or symbolize segments and construct indexes. The segmentation of time series is not closely integrated with index building. DSTree [24] improves the data locality by adopting a dynamic segmentation method. However, the time series representation can be improved. As Fig. 1 shows, we can acquire a better representation of time series by utilizing Piecewise Linear Approximation (PLA).

In this paper, we study a more accurate time series representation based on DSTree, which improves the bound tightness and search efficiency by utilizing PLA. Note that our work can be easily extended to *subsequence matching* [6] where query time series are allowed to have different lengths.

We propose EPLA-DSTree, which extends piecewise linear approximation on a DSTree index for whole matching on time series. By combining Piecewise Linear Approximation (PLA) with dynamic segmentation technique of DSTree, which not only offers better representation accuracies, but also support upper bound estimations, which enrich the functionalities of index greatly. As Fig. 1 shows, PLA representation is more similar to the real time series compared with EAPCA representation, which is adopted by DSTree. Furthermore, we present an parallel EPLA-DSTree on MapReduce, which is a popular cloud programming model. By this way, EPLA can be used to handle huge amount of sensor cloud data. The main idea of the parallel approach is to utilize a Locality Sensitive Hashing [27] for data partitioning.

The main contributions include:

- We propose a novel time series representation by extending PLA, named as EPLA.
- We propose a novel time series index, named as EPLA-DSTree by combining EPLA and DSTree.
- We implement a similarity search approach utilizing EPLA-DSTree on a real bridge monitoring system, which collects monitoring data by more than one thousand sensors.

TABLE 1. Some frequently used symbols.

Symbol	Meaning
$D(S, S')$	the (Euclidean) distance between time series S and S'
r_j	the right end time instant of segment j
μ_j^S	the mean of segment j in time series S
σ_j^S	the standard deviation of segment j in time series S
SG	a segmentation of a time series
N	a node in a DSTree
$UB(N)$	the upper bound of distances between time series in node N
$LB(N_1, N_2)$, $UB(N_1, N_2)$	the lower and upper bounds of distances between two batches of time series in N_1 and N_2

- We propose a parallel approach based on Locality Sensitive Hashing of EPLA-DSTree on MapReduce to meet the requirements of Sensor Cloud applications.

The rest of the paper is organized as follows. Section II-A presents preliminary knowledge of the work. Section III introduces the EPLA representation and how it is applied to the DSTree index and improve the data locality and query performance. Section IV reports the experiment results. Section V reviews the related work. Section VI concludes the paper. Table 1 summarizes the symbols frequently used in this paper.

II. PRELIMINARY KNOWLEDGE

A. DSTree

A DSTree supports two types of queries. The first one is the traditional similarity search, which returns the time series nearest to the query time series. The second type is to estimate the distance distribution, which returns a histogram of distances between the query time series and all indexed time series.

Before introducing the exact similarity search, we first introduce a *heuristic search* method, which is more efficient and will be used in the exact search method later.

1) A HEURISTIC METHOD

Instead of finding the exact most similar time series by checking all possible nodes in a DSTree, a heuristic search only investigates one leaf node, and tries to find the most similar time series in this node. This method is based on the heuristic that similar time series are often indexed in the same node.

Specifically, given a query Q, we start from the root node. If the root node is not a leaf node, then we find a child node of the root node that can hold Q as if Q ware inserted into the index. This search process is conducted recursively until a leaf node N is met. Then, we calculate the distance $D(S, Q)$ for every time series $S \in \mathcal{TS}_N$, and return the time series of the shortest distance. Please note that the heuristic method, as the name suggests, may not find the most similar time series in the whole data set.

2) THE EXACT SEARCH

To speed up search, we combine the heuristic search method and the lower bounding distance function to prune the search space. The exact search begins with a best-so-far (BSF)

answer returned by the heuristic search method. The intuition is that, by quickly obtaining a time series that is likely similar to the query time series, a large portion of the search space may be pruned effectively.

B. PLA

Given a time series $S = (s_1, \ldots, s_n)$ of length n, PLA divides it into several joint segments $S = (S_1, \ldots, S_m)$, $(m = \frac{n}{w})$, where $S_i = (s_{r_{i-1}+1}, \ldots, s_{r_i})$ $(r_0 = 0, 1 \leq r_1 < \cdots < r_m = n)$. PLA approximates each segment with a linear representation. Concretely, for the i-th segment, we re-denote it as $S_i = (s_{i1}, \ldots, s_{iw_i})$, where $w_i = r_i - r_{i-1}$, PLA uses a linear function $\tilde{s}_j^i = a_i \cdot j + b_i$ $(1 \leq j \leq w_i)$ to approximate S_i, where a_i and b_i are the slope and the intercept respectively such that the reconstruction error (sum of squared error), $sse(S_i)$, is minimized. $sse(S_i)$ is defined as the Euclidean distance between the approximated and actual time series.

$$sse(S_i) = \sum_{j=1}^{w_i} (s_{ij} - \tilde{s}_{ij})^2 = \sum_{j=1}^{w_i} (s_{ij} - (a_i \cdot j + b_i))^2$$

where two parameters a_i and b_i satisfy the following two conditions:

$$\frac{\partial(S_i)}{\partial(a_i)} = 0$$
$$\frac{\partial(S_i)}{\partial(b_i)} = 0$$

Here, a_i and b_i can be obtained by solving the above formula. In particular, we have:

$$a_i = \frac{12 \sum_{j=1}^{w_i} (j - \frac{l+1}{2}) s_{ij}}{w(w-1)(w+1)}$$
$$b_i = \frac{6 \sum_{j=1}^{w} (j - \frac{2w+1}{3}) s_{ij}}{w(w-l)}$$

III. EXTENDING THE PLA REPRESENTATION

Piecewise Linear Approximation (PLA) [5], [13] is a well-known time series representation, which approximates time series using consecutive linear segments. In this section, we extend PLA to EPLA. Similar to EAPCA, both the lower and upper bounds of distances are provided. Before introducing EPLA in detail, we offer a visual demonstration of the advantage of PLA (A detailed review of PLA will be given in Section III-A) over APCA when handling violent fluctuations in time series, which is illustrated in Fig. 1. Clearly, PLA is far less lossier than APCA in this case.

A. PLA PROPERTIES

In fact, PLA has two interesting properties. Let $\varepsilon_{ij} = s_{ij} - a_i \cdot j - b_i$ $(1 \leq j \leq w_i)$. we have

$$\sum_{j=1}^{w_i} j\varepsilon_{ij} = 0 \quad (1)$$

$$\sum_{j=1}^{w_i} \varepsilon_{ij} = 0 \quad (2)$$

For each segment S_j, having obtained the corresponding slope a_j and intercept b_j, the PLA representation of the time series S is $\tilde{S} = ((a_1, b_1, r_1), \ldots, (a_m, b_m, r_m))$. Using the slopes and intercepts, PLA can give a lower bound of the distance between two time series, which is given as follows.

Lemma 1 (PLA Lower Bound): Given two time series X and Y such that $|X| = |Y|$, let $\tilde{X} = ((a_1^X, b_1^X, r_1), \ldots, (a_m^X, b_m^X, r_m))$ and $\tilde{Y} = ((a_1^Y, b_1^Y, r_1), \ldots, (a_m^Y, b_m^Y, r_m))$ be two PLA representations of X and Y, respectively. Then,

$$D(X, Y) \geq \sqrt{\sum_{i=1}^{m} d_i^{PLA}} \quad (3)$$

where

$$d_i^{PLA} = \sum_{j=1}^{w_i} ((a_i^X - a_i^Y)j + (b_i^X - b_i^Y))^2 \quad (4)$$

We omit the proof of Lemma 1 for space usage.

Next, we introduce our new EPLA, which, by combining the sum of squared error, can provide an upper bound and a tighter lower bound on distances.

B. EPLA AND UPPER/LOWER BOUNDS USING SUM OF SQUARED ERROR

Now we give the definition of EPLA. Recall that in EAPCA, the authors introduced the standard deviation to describe the approximation quality. Likewise, here we apply *sse* to describing that of the linear segments. Concretely, for time series S of length n, EPLA approximates it as $\tilde{S} = ((a_1, b_1, sse_1, r_1), \ldots, (a_m, b_m, sse_m, r_m))$, where a is the slope, b is the intercept and *sse* is the sum of squared error. We have the following results.

Theorem 1: Given two time series X and Y such that $|X| = |Y|$, let $\tilde{X} = ((a_1^X, b_1^X, sse_1^X, r_1), \ldots, (a_m^X, b_m^X, sse_m^X, r_m))$ and $\tilde{Y} = ((a_1^Y, b_1^Y, sse_1^Y, r_1), \ldots, (a_m^Y, b_m^Y, sse_m^Y, r_m))$ be two aligned EPLA representations of X and Y, respectively. Then,

$$D(X, Y) \geq \sqrt{\sum_{i=1}^{m} d_i^{PLA} + (\sqrt{sse_i^X} - \sqrt{sse_i^Y})^2} \quad (5)$$

and

$$D(X, Y) \leq \sqrt{\sum_{i=1}^{m} d_i^{PLA} + 2(sse_i^X + sse_i^Y)} \quad (6)$$

Proof:

Using the same definition of X_i, Y_i, ε_{ij}^X and ε_{ij}^Y as those in Lemma 1, and letting $\Delta = \sum_{j=1}^{w_i} (\varepsilon_{ij}^X - \varepsilon_{ij}^Y)^2$, we now begin to deduce

$$\Delta \geq (\sqrt{sse_i^X} - \sqrt{sse_i^Y})^2 = \left(\sqrt{\sum_{j=1}^{w_i} (\varepsilon_{ij}^X)^2} - \sqrt{\sum_{j=1}^{w_i} (\varepsilon_{ij}^Y)^2} \right)^2 \quad (7)$$

and

$$\Delta \leq 2(sse_i^X + sse_i^Y) = 2 \left(\sum_{i=1}^{w_i} (\varepsilon_{ij}^X)^2 + \sum_{i=1}^{w_i} (\varepsilon_{ij}^Y)^2 \right) \quad (8)$$

For Inequation 7

$$\Delta = \sum_{j=1}^{w_i}(\varepsilon_{ij}^X - \varepsilon_{ij}^Y)^2 = \sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2 - 2\sum_{j=1}^{w_i}\varepsilon_{ij}^X\varepsilon_{ij}^Y$$

By the Cauchy-Schwarz inequality

$$(\sum_{j=1}^{w_i}\varepsilon_{ij}^X\varepsilon_{ij}^Y)^2 \le (\sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2)(\sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2)$$

therefore,

$$\begin{aligned}\Delta &\ge \sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2 - 2\sqrt{\sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2\sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2} \\ &= \left(\sqrt{\sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2} - \sqrt{\sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2}\right)^2 \end{aligned} \quad (9)$$

For Inequation 8

$$\begin{aligned}\sum_{j=1}^{w_i}(\varepsilon_{ij}^X - \varepsilon_{ij}^Y)^2 &= \sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2 - 2\sum_{j=1}^{w_i}\varepsilon_{ij}^X\varepsilon_{ij}^Y \\ &= 2\left((\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2\right) \\ &\quad - \sum_{j=1}^{w_i}((\varepsilon_{ij}^X)^2 + (\varepsilon_{ij}^Y)^2 + 2\varepsilon_{ij}^X\varepsilon_{ij}^Y) \\ &= 2\left(\sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2\right) - \sum_{j=1}^{w_i}(\varepsilon_{ij}^X + \varepsilon_{ij}^Y)^2 \\ &\le 2\left(\sum_{j=1}^{w_i}(\varepsilon_{ij}^X)^2 + \sum_{j=1}^{w_i}(\varepsilon_{ij}^Y)^2\right) \end{aligned} \quad (10)$$

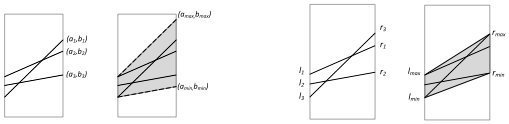
With both Inequations (7) and (8) proved, it is apparent that both the lower and upper bounds hold. ■

Comparing Inequations (3) and (5), the lower bound given by EPLA uses the sums of squared error to achieve a tighter lower bound than PLA. Both the lower and upper bounds given by EPLA are realizable.

C. BOUNDING DISTANCES TO A SET OF TIME SERIES

If we can acquire the lower bound between a query and a set of time series, we can prune the calculations for the query with the time series in the set with the lower bound greater that the given threshold. The tighter the lower bound is, the better the pruning power we have.

For a time series X and a set of time series $Y_1, \ldots, Y_c$, $(|X| = |Y_1| = \cdots = |Y_c|)$, let $\tilde{X} = ((a_1^X, b_1^X, sse_1^X, r_1), \ldots, (a_m^X, b_m^X, sse_m^X, r_m))$, $\tilde{Y_1} = ((a_1^{Y_1}, b_1^{Y_1}, sse_1^{Y_1}, r_1), \ldots, (a_m^{Y_1}, b_m^{Y_1}, sse_m^{Y_1}, r_m))$, $\ldots$, $\tilde{Y_c} = ((a_1^{Y_c}, b_1^{Y_c}, sse_1^{Y_c}, r_1), \ldots, (a_m^{Y_c}, b_m^{Y_c}, sse_m^{Y_c}, r_m))$ be aligned EACPA representations, respectively. Let the minimal and maximal values of the distance between two lines in the i-th segments of $Y_1, \ldots, Y_c$ be $d\_PLA_i^{min} = \min_{1\le j\le l}\{d\_PLA_i^{Y_j}\}$ and $d\_PLA_i^{max} = \max_{1\le j\le l}\{d\_PLA_i^{Y_j}\}$ respectively. Moreover, let the minimal and maximal values of the sum of squared error in the i-th segments of $Y_1, \ldots, Y_c$, respectively, be $sse_i^{min} = \min_{1\le j\le l}\{sse_i^{Y_j}\}$ and $sse_i^{max} = \max_{1\le j\le l}\{sse_i^{Y_j}\}$. We have the following bounds.



(a) Approximation by a and b (b) Approximation by the end points

FIGURE 2. Set approximations.

$$\min_{1\le j\le c}\{D(X, Y_j)\} \ge \sqrt{\sum_{i=1}^{m}(r_i - r_{i-1})(LB_i^{d\_PLA} + LB_i^{sse})} \quad (11)$$

and

$$\max_{1\le j\le c}\{D(X, Y_j)\} \le \sqrt{\sum_{i=1}^{m}(r_i - r_{i-1})(UB_i^{d\_PLA} + UB_i^{sse})} \quad (12)$$

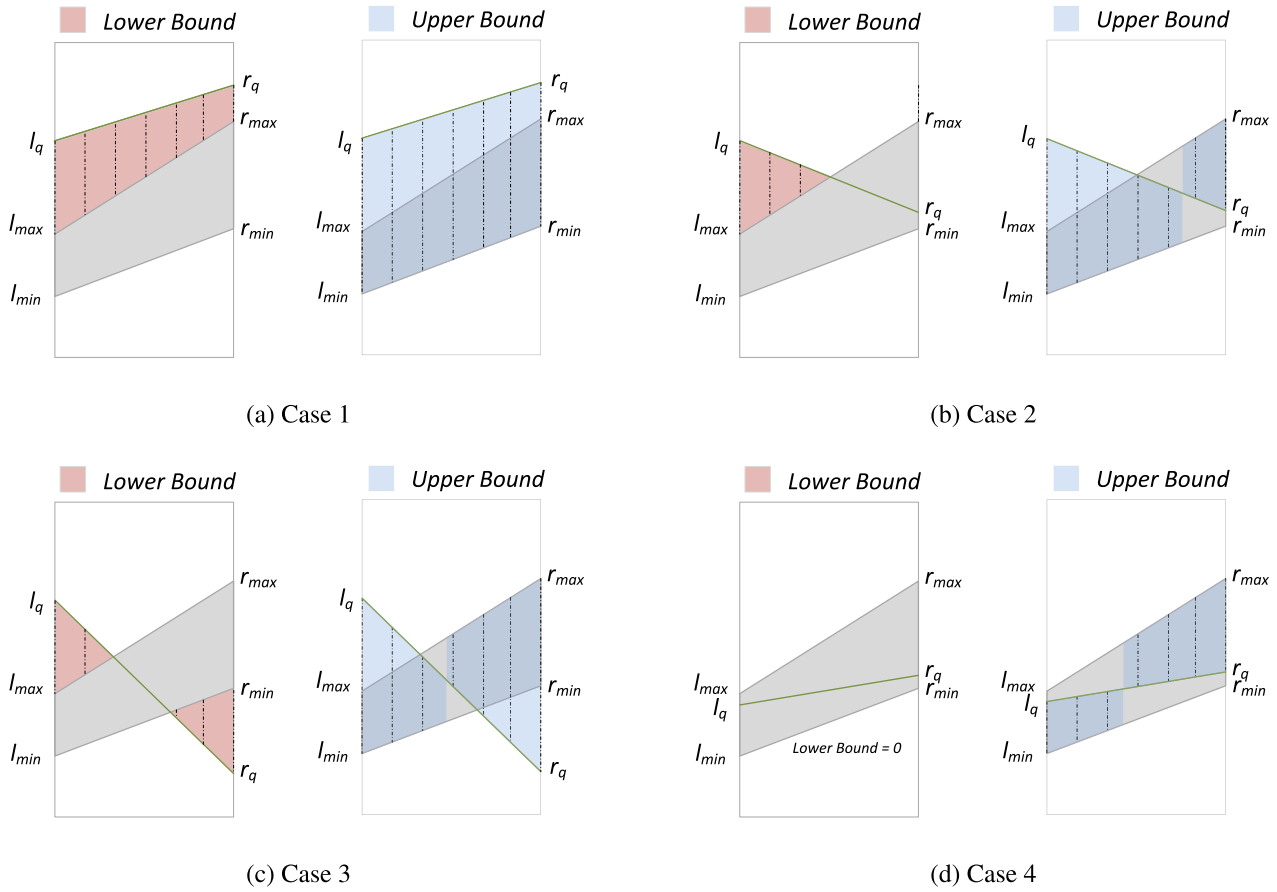
It is apparent that similar to the mean values and standard deviations in EAPCA, EPLA utilizes the distance between to lines d_PLA and the sum of squared error sse to construct the bounds. Before giving an in-depth view of how the bounds of the two parts can be calculated, we first discuss the two representation choices for a time series segment given a linear approximation. To be concrete, given the set of time series $Y_1, \ldots, Y_c$ and their EPLA representations $\tilde{Y_c}$, $\ldots$, $\tilde{Y_c}$ as mentioned above, we have two types of representations for the lines in the i-th segment, as is illustrated in Fig. 2.

In the first type, we use the slope and the intercept to represent a line. Let us illustrate it with the example in Fig. 2(a), where there are three lines, $\tilde{Y}_i^1, \tilde{Y}_i^2, \tilde{Y}_i^3$, where $\tilde{Y}_i^k$ $1 \le k \le 3$ is represented by two parameters, (a_i^k, b_i^k). We combine the largest of each of the two parameters, a_i^{max} and b_i^{max}, to form a line, and a_i^{min} and b_i^{min} to form another. As shown in Fig. 2(a), these two lines can envelop all three lines.

In the second type, we use the left and right ending points to represent the lines. We continue with the example in Fig. 2, where for the same lines as that in Fig. 2(a), $\tilde{Y}_i^1, \tilde{Y}_i^2, \tilde{Y}_i^3$, $\tilde{Y}_i^k$ is thus represented as (l_i^k, r_i^k), where $l_i^k = a_i^k + b_i^k$ and $r_i^k = a_i^k(r_i - r_{i-1}) + b$. Then, by connecting l_i^{min} and r_i^{min}, we can obtain the bottom line in Fig. 2(b), denoted by L_i^{min}; by connecting l_i^{max} and r_i^{max}, we can obtain the top line, denoted by L_i^{max}. These two lines can also envelop all three lines.

It is easy to see that the second type can give a tighter approximation. Therefore, in our work, we use the second type. That is, we use four parameters, l^{max}, l^{max}, r^{min}, r^{min}, to construct the bounds.

We are now finally in a position to specify the bounds of d_PLA and the bounds of sse. The latter can be obtained in a



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

FIGURE 3. An illustration of the naive set bounds.

similar manner to that used to obtain the bounds of standard deviation in EAPCA, that is,

$$LB_i^{sse} = \begin{cases} (\sqrt{sse_i^{min}} - \sqrt{sse_i^X})^2 & \text{if } sse_i^X \le sse_i^{min}; \\ 0 & \text{if } sse_i^{min} < sse_i^X \le sse_i^{max}. \\ (\sqrt{sse_i^{max}} - \sqrt{sse_i^X})^2 & \text{if } sse_i^{max} \le sse_i^X; \end{cases} \tag{13}$$

$$UB_i^{sse} = 2(\sqrt{sse_i^{max}} + \sqrt{sse_i^X}) \tag{14}$$

Next we will focus on obtaining the bounds of the distance between to lines, i.e. the bounds of d_PLA, using l^{max}, l^{max}, r^{min}, r^{min}. We propose two types of bounds, naive bounds and line-based bounds.

1) NAIVE BOUNDS

We now introduce the naive bounds of d_PLA. For simplicity we only focus on one segment, i.e. the i-th segment, the length of which is $w_i = r_i - r_{i-1}$. Recall that in '(4) d_PLA_i is a summation of w_i values. Then we first select the possible minimal value for each term, i.e. each j in (4), and sum them up to form the lower bound. Fig. 3 illustrates the cases according to the position of $\tilde{X}_i = (l_i^X, r_i^X)$. Here we elaborate on case 1.

In case 1, it holds that $l_i^X < l_i^{max}$ and $r_i^X < r_i^{max}$. For each j, the j-th value in $\tilde{X}_i$ be $|l_i^X + (r_i^X - l_i^X) \cdot \frac{j-1}{w_i - 1}|$, denoted by $\tilde{X}_{ij}$. Similarly, for each $\tilde{Y}_i^k$, $1 \le k \le c$, we denote the j-th value in it by $\tilde{Y}_{ij}^k$, that in the top line L_i^{max} by L_{ij}^{max}, and that in the bottom line L_i^{min} by L_{ij}^{min}.

Then for any j, we have

$$(\tilde{X}_{ij} - L_{ij}^{max})^2 \le (\tilde{X}_{ij} - \tilde{Y}_{ij}^k)^2 \le (\tilde{X}_{ij} - L_{ij}^{min})^2$$

By summing all j's, the lower bound is

$$\sum_{j=1}^{n} (\tilde{X}_{ij} - L_{ij}^{max})^2$$

and the upper bound is

$$\sum_{j=1}^{n} (\tilde{X}_{ij} - L_{ij}^{min})^2$$

The lower and upper bounds in other cases are summarized in Fig. 3.

2) LINE-BASED BOUNDS

The native bound is straightforward, and easy to compute. However, it is not optimal. The main reason is that the values used to compute the lower (or upper) bound may not be in the same line. For example, for the upper bound in case 2,

the left part is computed by $\tilde{X}_i$ and L_i^{max}, while the right part is computed by $\tilde{X}_i$ and L_i^{min}. Apparently, this bound is unreachable.

In this section, we introduce the line-based bounds (Fig. 6), which guarantee reachability. The underlying rationale is that for $\tilde{X}_i$, we find a line in the region, the distance between which and $\tilde{X}_i$ is minimal (or maximal). We illustrate this with the lower bound in Fig. 6(b). The red dotted line, denoted by $\tilde{(Y)}_i^{min}$, is the line which minimizes d_PLA_i. That is, it holds that, for the lower bound

$$d\_PLA(\tilde{X}_i, \tilde{Y}_i^{min}) \leq d\_PLA(\tilde{X}_i, \tilde{Y}_i^k), \quad 1 \leq k \leq c$$

For the upper bound, we find a line, denoted by $\tilde{(Y)}_i^{max}$, which holds that

$$d\_PLA(\tilde{X}_i, \tilde{Y}_i^{max}) \geq d\_PLA(\tilde{X}_i, \tilde{Y}_i^k), \quad 1 \leq k \leq c$$

Next, we introduce how to find $\tilde{Y}_i^{min}$ and $\tilde{Y}_i^{max}$ for the lower bound. We formalize this as the following problem. Given the query line $\tilde{X}_i = (l_i^X, r_i^X)$, the j-th value in it being $\tilde{X}_{ij} + \frac{r_i^X - l_i^X}{w_i - 1} \cdot (j-1)$, $(1 \leq j \leq w_i)$. We try to find a line, so that between it and $\tilde{X}_i$, d_PLA is minimal.

Note that any line $\tilde{Y}_i^k$ in the i-th segment, denoted by (l_i^k, r_i^k), can be expressed as,

$$\tilde{Y}_{ij}^k = l_i^k + \frac{r_i^k - l_i^k}{w_i - 1} \cdot (j-1) \; (1 \leq j \leq w_i)$$

where

$$l_i^{min} \leq l_i^k \leq l_i^{max}$$
$$r_i^{min} \leq r_i^x \leq r_i^{max}$$

We have

$$\begin{aligned} d\_PLA(\tilde{X}_i, \tilde{Y}_i^k) &= \sum_{j=1}^{w_i} (l_i^X + \frac{r_i^X - l_i^X}{w_i - 1} \cdot (j-1) \\ &\quad - l_i^k - \frac{r_i^k - l_i^k}{w_i - 1} \cdot (j-1))^2 \\ &= \sum_{j=0}^{w_i - 1} (l_i^X + \frac{r_i^X - l_i^X}{w_i - 1} \cdot j - l_i^k \\ &\quad - \frac{r_i^k - l_i^k}{w_i - 1} \cdot j)^2 \\ &= \sum_{j=0}^{w_i - 1} (l_i^X - l_i^k \\ &\quad + \frac{r_i^X - l_i^X - r_i^k + l_i^k}{n - 1} \cdot i)^2 \end{aligned} \tag{15}$$

Let $\Delta l = l_i^X - l_i^k$, $\Delta r = r_i^X - r_i^k$, as shown in Fig. 4. We then have

$$\begin{aligned} d\_PLA(\tilde{X}_i, \tilde{Y}_i^k) &= \sum_{j=0}^{w_i - 1} (\Delta l + \frac{\Delta r - \Delta l}{w_i - 1}) \cdot j^2 \\ &= \sum_{j=0}^{w_i - 1} (\Delta l^2 + (\frac{\Delta r - \Delta l}{w_i - 1}) \cdot j)^2 \\ &\quad + 2\Delta l \frac{\Delta r - \Delta l}{w_i - 1} \cdot j) \end{aligned}$$

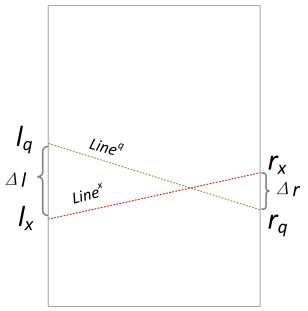


FIGURE 4. Δl and Δr.

$$\begin{aligned} &= \sum_{j=0}^{w_i - 1} \Delta l^2 + \sum_{j=0}^{w_i - 1} (\frac{\Delta r - \Delta l}{w_i - 1})^2 \cdot j^2 \\ &\quad + 2\Delta l \sum_{j=0}^{w_i - 1} \frac{\Delta r - \Delta l}{w_i - 1} \cdot j \\ &= n\Delta l^2 + (\frac{\Delta r - \Delta l}{w_i - 1})^2 \cdot \frac{1}{6}(w_i - 1) w_i (2w_i - 1) \\ &\quad + 2\Delta l \frac{\Delta r - \Delta l}{w_i - 1} \cdot \frac{1}{2}(w_i - 1) w_i \\ &= \frac{w_i}{6(w_i - 1)} ((2w_i - 1)(\Delta r^2 + \Delta l^2) \\ &\quad + (2w_i - 4)\Delta r \Delta l) \end{aligned} \tag{16}$$

We take the partial derivative of Δl and Δr respectively,

$$\frac{\partial d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)}{\partial \Delta l} = 0$$
$$\frac{\partial d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)}{\partial \Delta r} = 0$$

that is,

$$\frac{w_i}{6(w_i - 1)} ((2w_i - 2) \cdot 2\Delta l + (2w_i - 4)\Delta r) = 0$$
$$\frac{w_i}{6(w_i - 1)} ((2w_i - 2) \cdot 2\Delta r + (2w_i - 4)\Delta l) = 0$$

It can be inferred that

$$\Delta l = \frac{2 - w_i}{2w_i - 1} \Delta r \tag{17}$$

$$\Delta r = \frac{2 - w_i}{2w_i - 1} \Delta l \tag{18}$$

Combining (17) and (18), we can obtain the following properties.

- $d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)$ is an upward paraboloid, as shown in Fig. 5.
- When $\Delta l > 0$, to minimize $d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)$, it holds that $\Delta r < 0$.
- When $\Delta l < 0$, to minimize $d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)$, it holds that $\Delta r > 0$.
- To maximize $d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)$, either l_i^k or r_i^k should take one of its corresponding boundary values.

Based on the properties above, we can obtain the line where the lower or upper bound can be reached, as well as the corresponding bound.

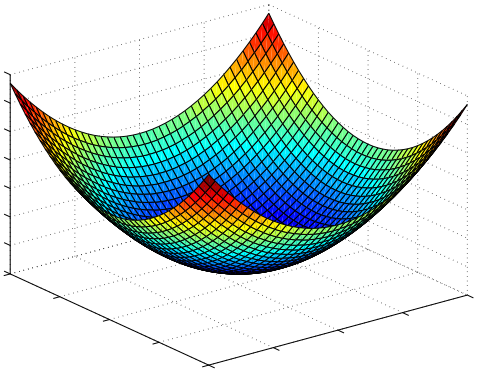


FIGURE 5. $d\_PLA(\tilde{X}_i, \tilde{Y}_i^k)$.

To be specific, there exist the following nine cases. Due to space limitations, only four of them are illustrated in Fig. 6.

Case 1: $l_i^X \geq l_i^{max}, r_i^X \geq r_i^{max}$, as is shown in Fig. 6(a).

In this case, the lower bound is reached when $l_i^k = l_i^{max}$ and $r_i^k = r_i^{max}$, while the upper bound is reached when $l_i^k = l_i^{min}$ and $r_i^k = r_i^{min}$.

Case 2: $l_i^X \geq l_i^{max}, r_i^{max} > r_i^X > r_i^{min}$, as is incompletely shown in Fig.6(b), due to the space limitations.

In this case, the lower bound is reached when $l_i^k = l_i^{max}$ and $r_i^k = min(r_i^{max}, r_i^X + \frac{2-w_i}{2w_i-1}(l_i^{max} - l_i^X))$. One of such occasions is shown in Fig.6(b).

The upper bound is reached on one of the following occasions: $l_i^k = l_i^{min}, r_i^k = r_i^{min}$, or $l_i^k = l_i^{min}, r_i^k = r_i^{max}$, or $l_i^k = l_i^{max}, r_i^k = r_i^{max}$, of which two are shown in Fig.6(b).

Case 3: $l_i^X \geq l_i^{max}, r_i^X \leq r_i^{min}$, as is incompletely shown in Fig. 6(c), due to the space limitations.

To obtain the lower bound in this case, two occasions require considering.

1) $l_i^X - l_i^{max} \geq r_i^{min} - r_i^X$, then $l_i^k = l_i^{max}$, $r_i^k = max(r_i^{min}, min(r_i^{max}, r_i^X + \frac{2-w_i}{2w_i-1}(l_i^{max} - l_i^X)))$.
2) $l_i^X - l_i^{max} < r_i^{min} - r_i^X$, then $l_i^k = max(l_i^{min}, min(l_i^{max}, l_i^X + \frac{2-w_i}{2w_i-1}(r_i^{min} - r_i^X)))$, $r_i^k = r_i^{min}$.

The upper bound can be reached on one of the following occasions. $l_i^k = l_i^{max}, r_i^k = r_i^{max}$ or $l_i^k = l_i^{max}, r_i^k = r_i^{min}$ or $l_i^k = l_i^{min}, r_i^k = r_i^{max}$ or $l_i^k = l_m in, r_i^k = r_i^{min}$.

Case 4: $l_i^{max} > l_i^X > l_i^{min}, r_i^X \geq r_i^{max}$

The lower bound in this case is reached when $l_i^k = min(l_i^{max}, l_i^X + \frac{2-w_i}{2w_i-1}(r_i^{max} - r_i^X)), r_i^k = r_i^{max}$.

The upper bound is reached on one of the following occasions: $l_i^k = l_i^{min}, r_i^k = r_i^{min}$, $l_i^k = l_i^{max}, r_i^k = r_i^{min}$ or $l_i^k = l_i^{max}, r_i^k = r_i^{max}$.

Case 5: $l_i^{max} > l_i^X > l_i^{min}, r_i^{max} > r_i^X > r_i^{min}$, as is shown in Fig. 6(d).

The lower bound in this case is reached when $l_i^k = l_i^X$, $r_i^k = r_i^X$.

The upper bound is reached on one of the following occasions: $l_i^k = l_i^{min}, r_i^k = r_i^{min}$, $l_i^k = l_i^{min}, r_i^k = r_i^{max}$, $l_i^k = l_i^{max}, r_i^k = r_i^{min}$ or $l_i^k = l_i^{max}, r_i^k = r_i^{max}$.

Case 6: $l_i^{max} > l_i^X > l_i^{min}, r_i^X \leq r_i^{min}$.

The lower bound in this case is reached when $l_i^k = max(l_i^{min}, l_i^X + \frac{2-w_i}{2w_i-1}(r_i^{min} - r_i^X)), r_i^k = r_i^{max}$.

The upper bound is reached on one of the following occasions: $l_i^k = l_i^{min}, r_i^k = r_i^{min}$, $l_i^k = l_i^{min}, r_i^k = r_i^{max}$ or $l_i^k = l_i^{max}, r_i^k = r_i^{max}$.

Case 7: $l_i^X \leq l_i^{min}, r_i^X \geq r_i^{max}$.

To obtain the lower bound in this case, two occasions require considering.

1) $l_i^{min} - l_i^X \geq r_i^X - r_i^{min}$, then $l_i^k = l_i^{min}$, $r_i^k = max(r_i^{min}, min(r_i^{max}, r_i^X + \frac{2-w_i}{2w_i-1}(l_i^{min} - l_i^X)))$.
2) $l_i^{min} - l_i^X < r_i^X - r_i^{min}$, then $l_i^k = max(l_i^{min}, min(l_i^{max}, l_i^X + \frac{2-w_i}{2w_i-1}(r_i^{max} - r_i^X)))$, $r_i^k = r_i^{max}$.

Case 8: $l_i^X \leq l_i^{min}, r_i^{max} > r_i^X > r_i^{min}$.

The lower bound in this case is reached when $l_i^k = l_i^{min}$, $r_i^k = max(r_i^{min}, r_i^X + \frac{2-w_i}{2w_i-1}(l_i^{min} - l_i^X))$.

The upper bound is reached on one of the following occasions: $l_i^k = l_i^{min}, r_i^k = r_i^{min}$, $l_i^k = l_i^{max}, r_i^k = r_i^{min}$ or $l_i^k = l_i^{max}, r_i^k = r_i^{max}$.

Case 9: $l_i^X \leq l_i^{min}, r_i^X \leq r_i^{min}$.

The lower bound in this case is reached when $l_i^k = l_i^{min}$, $r_i^k = r_i^{min}$.

The lower bound in this case is reached when $l_i^k = l_i^{max}$, $r_i^k = r_i^{max}$.

In the nine cases above, bounds tighter than the naive ones can be reached in all cases except Case 1 and Case 9, making the line-based bounds a better choice for obtaining $LB_i^{d\_PLA}$ in 11 and $UB_i^{d\_PLA}$ in 12. Combined with 13 and 14, the bounds on set can be easily computed.

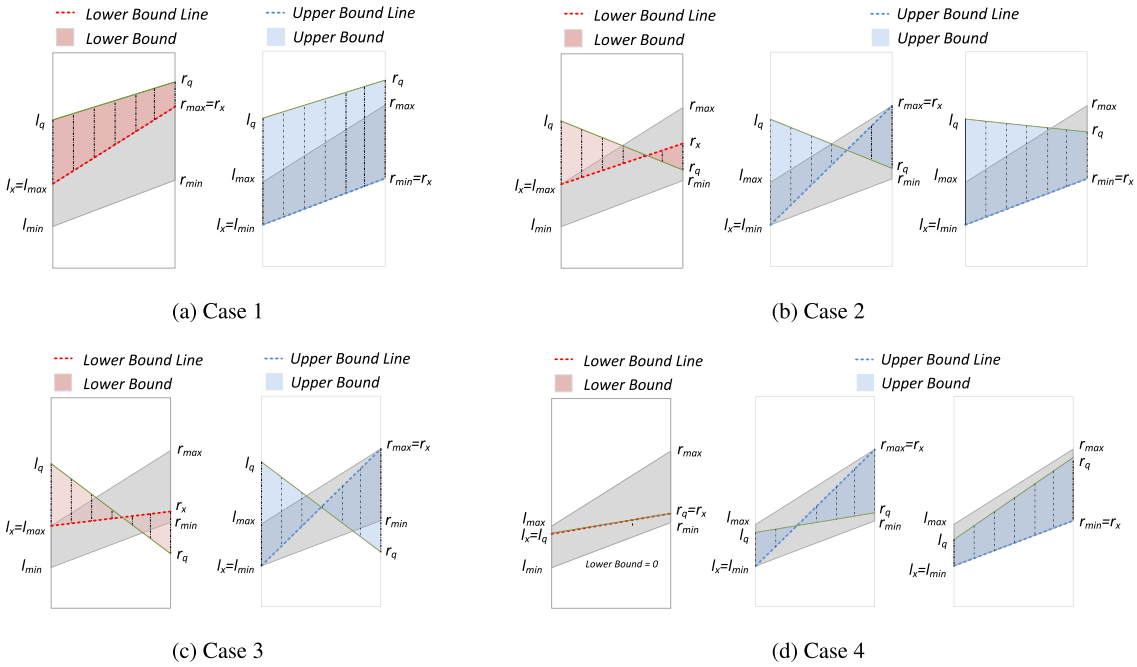
D. INCORPORATING EPLA WITH DSTree

We now demonstrate how EPLA can be fitted into DSTree [24] index. The structure of the index and the contents of each node remains mostly the same as that of EAPCA-DSTree ([24]), except that for the synopsis $Z = (z_1, z_2, \ldots, z_m)$ requires changes in accordance to EPLA, where $z_i = (l_i^{min}, l_i^{max}, r_i^{min}, r_i^{max}, sse_i^{min}, sse_i^{max})$. Also, the splitting strategies require adaptations to EPLA, where in horizontal splitting, we can split by the left point, the right point or *sse*. For the splitting strategy quality measure, let the slope and intercept of the bottom line in the i-th segment be (a_i^{bot}, b_i^{bot}), and the slope and intercept of the top line in the i-th segment be (a_i^{top}, b_i^{top}). We have

$$Qos = \sum_{i=1}^{m} \left[sse_i^{max} + \sum_{j=1}^{r_i - r_{i-1}+1} ((a_i^{top} - a_i^{bot})j + (b_i^{top} - b_i^{bot}))^2 \right] \tag{19}$$

E. EXACT SEARCH USING EPLA-DSTree

The pseudo-code of exact search using EPLA-DSTree is illustrated in Figure.7, which is similar with the exact search algorithm in [24]. The main different is that EPLA-DSTree has a tighter lower bound. The main idea of the algorithm7



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

FIGURE 6. An incomplete illustration of the line-based set bounds.

1: **Input**: A query time series Q
2: **Output**: The nearest time series TS with distance D_{bsf}
3: Select N_{near} by inserting Q into the index tree, N_{near} is the arriving leaf node
4: $(TS, D_{bsf}) = calcMinDist(N_{near}, Q)$;
5: Initialize distance priority queue $queue$;
6: $queue.Add(N_R, D_{LB}(N_R, Q))$;
7: **while** $!queue.isEmpty()$ **do**
8: $(N_{cur}, LB_{cur}) = queue.PopMin()$;
9: **if** $LB_{cur} > D_{bsf}$ **then**
10: break;
11: **end if**
12: **if** N_{cur} is a leaf node **then**
13: $(X, Dist) = calcMinDist(N_{cur}, Q)$;
14: **if** $Dist < D_{bsf}$ **then**
15: $D_{bsf} = Dist; TS = X$;
16: **end if**
17: **else**
18: **for all** children nodes N' of N_{cur} **do**
19: **if** $D_{LB}(N', Q) < D_{near}$ **then**
20: $queue.add(N', D_{LB}(N', Q))$;
21: **end if**
22: **end for**
23: **end if**
24: **end while**
25: return TS,D_{near};

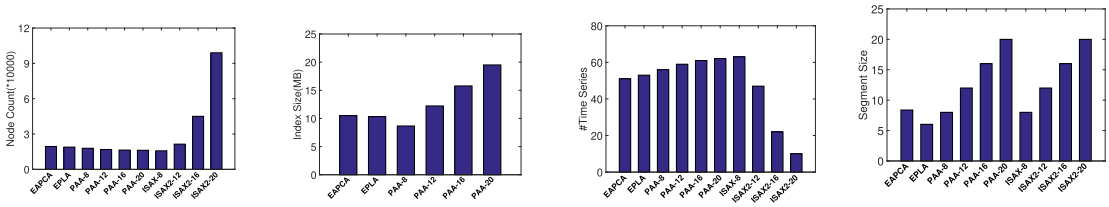
FIGURE 7. *SearchEPLA(Q).*

is that EPLA-DSTree prunes unnecessary comparisons by utilizing a lower bound (Line 9 - Line 11). Only the leaf nodes with a larger distance than the current lower bound are left for comparisons (Line 12 - Line 16). When reaching a inner node, the child node is pushed into the queue if the distance is less than the current lower bound (Line 18 - Line 22).

F. SCALABILITY CONSIDERATION

There are two key factors for the parallelism of EPLA index on cloud platforms. The first one is load balance and the second one is data locality. We aim to find a data partitioning approach to meet both the load balance and the data locality. A simple data splitting method is to utilize hashing technologies. For high-dimensional sensor data, Locality Sensitivity Hashing (LSH) [27] is a good choice for data splitting. The basic idea of locality sensitive hashing is to use hash functions that map similar objects into the same hash buckets with high probability. LSH function families have the property that objects that are close to each other have a higher probability of colliding than objects that are far apart. We consider parallel construction of EPLA tree on MapReduce [28], which is a popular programming model used by cloud platforms. MapReduce was introduced by Dean et. al. in 2004. It is a software architecture proposed by Google. The kernel idea of MapReduce is map and reduce. In map phase, the LSH value for each time series is calculated. The time series with the same LSH values are grouped in a reduce function. In each reduction function, an EPLA tree index is constructed for the time series in an LSH bucket.

Although a parallel EPLA index is built on MapReduce, users need not use MapReduce jobs to search for only one query. Because the trees are stored on HDFS and a serial program is allowed to read the files on HDFS too. If users have a batch of queries, then parallel searching by adopting a MapReduce job is necessary. Given a query, the LSH value is calculated first. The EPLA tree of the LSH value is positioned and an approximate time series is found and the corresponding distance (denoted as *dis*) is calculated. Notice that the lower bound property of LSH, not all EPLA trees are



(a) Number of nodes (b) Index size (c) Average #ts per leaf (d) # segments/node

FIGURE 8. Index size on the real data set.

necessary for the query. Only the ones with the lower bound less than *dis* are needed.

G. AN EXAMPLE OF EPLA ON SENSOR DATA

We used a real data set collected in a bridge condition monitoring system. There are more than 1000 sensors in the system. Each sensor monitors a special condition of the bridge. The data of the sensors are collected and treated as time series. The length of each time series is 256, and one million times series were collected. The total storage space is about 3GB. The health problems of bridge can be reflected by the time series. For example, a bombing event is represented as a special feature in time series. By searching the special feature (treated as a time series too), we can find all the bombing events happening in the given period.

IV. EMPRICAL EVALUATION

In this section, we report extensive experiments to verify the effectiveness of EPLA-DSTree. We compare both PAA-index (using PAA as representation and R-tree as index) and EAPCA-DSTree, iSAX2.0 with EPLA-DSTree in index efficiency, approximate search error rate and pruning power. We also showcase the lower bound tightness. All experiments were executed on a laptop computer with an Intel Core i5 2.7GHz CPU and 64GB main memory. All experimental results were averaged over 50 runs. The source code can be found in [30].

A. DATA SETS AND DEFAULT SETTING

The time series in both synthetic and real data sets were normalized with Z-normalization.

1) SYNTHETIC DATA SETS

Each of our synthetic data set is a combination of four types of time series as follows.

- Random walk times series.
- One-segment Gaussian time series.
- Multi-segment Gaussian time series.
- A mixed sine time series.

To generate a time series, the synthetic data generator first randomly chooses a type, and then picks the corresponding parameters randomly to generate the time series. We generated four synthetic data sets of time series lengths 64, 128, 256 and 512, respectively. Each data set contains one million time series by default. We also use synthetic data sets of up to 200 million time series in the scalability test.

2) REAL DATA SETS

We used a real data set collected in a bridge condition monitoring system. The length of each time series is 256, and one million times series were collected. The total storage space is about 3GB.

3) PARAMETERS

In an attempt to verify the effectiveness of data-adaptive and dynamic segmentation versus global segmentation, we compare EAPCA-DSTree and EPLA-DSTree with PAA-index (implemented by ourselves) and iSAX2.0 (source code provided by the authors). Both PAA-index and iSAX2.0 use fixed, global segmentations. To test the performance extensively, we built PAA-index and iSAX2.0 with segment sizes of 8, 12, 16 and 20 respectively. The leaf capacity threshold, ψ, was set to 100. The FBL size for iSAX2.0 was set to 200,000. The fill factor of R-tree in PAA-index was set to 0.5.

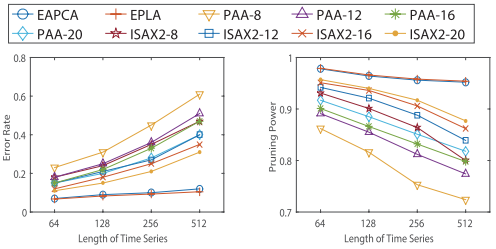
B. INDEX SIZE

We did not implement iSAX2.0 by ourselves. Instead, we used the implementation provided by the authors. We realize that the the implementation details, particularly the storage methods, in iSAX2.0 and the two types of DSTrees may be different. To avoid any confusion, we report the absolute index size for the methods we implemented but not for iSAX2.0.

The first group of experiments compare the index space cost of EAPCA-DSTree, EPLA-DSTree, PAA-index and iSAX2.0 with respect to the length of time series. Specifically, we report three measurements, namely the number of nodes in the tree, the physical index size, and the average number of time series contained by a leaf node. The number of nodes includes both internal and leaf nodes. Considering the difference on data representations in the four approaches, we also compare the physical index size for the two types of DSTree and PAA. We use the average number of time series in leaf nodes to evaluate the balance of the index nodes.

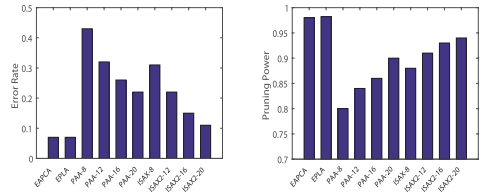
Since the two DSTrees use dynamic segmentation strategies, the segment size varies in different nodes. We report the average segment size with respect to the length of time series, that is, the ratio of the total number of segments in all nodes against the number of nodes.

Figure 8 shows the results on the real data set. The trends are similar to those on the synthetic data sets. The number of nodes of either DSTree is similar to those of iSAX2-12 and smaller than those of iSAX2-16 and iSAX2-20. The average



(a) Heuristic search error rate (b) Exact search pruning power

FIGURE 9. Error rate and pruning power on the synthetic data sets.



(a) Heuristic search error rate (b) Exact search pruning power

FIGURE 10. Error rate and pruning power on the real data set.

number of segments per node of EAPCA-DSTree is 8.38, that of EPLA-DSTree being 6.03. Although the time series in the real data set are more diverse, EPLA-DSTree can still represent the time series with a small number of segments, which verifies the effectiveness of the dynamic splitting strategy in EPLA-DSTree.

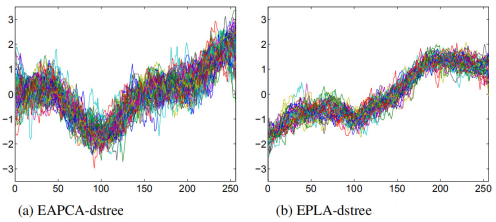
C. ACCURACY

We tested the effectiveness of the indexes in similarity search, including both heuristic search and exact search. The accuracy of heuristic search is measured by the error rate $E = \frac{|\overline{D}-D|}{D}$, where D and $\overline{D}$ are the distance between the query time series and the exact nearest neighbor and the heuristic search result, respectively.

For exact search, we compare the *pruning power*, which is the ratio of the number of time series pruned against the total number of time series. For both heuristic and exact search, 100 time series were used as the queries, half of them picked randomly from the data set, and the rest generated randomly. Figures 9 and 10, respectively, show the results on the synthetic and real data sets.

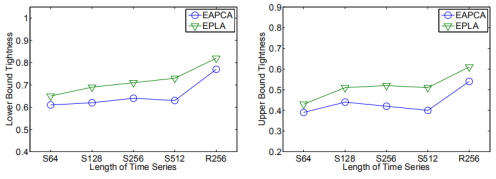
In Figure 9(a), although the error rate increases as the length of time series increases for all three methods, both DSTrees outperform the others clearly, with EPLA-DSTree superior to EAPCA-DSTree.

The advantages of EPLA-DSTree are from two factors. First, a tighter lower bound helps to prune more nodes. Second and more importantly, the less split segmentations can reduce the index build time and improve query efficiency. Consequently, the heuristic search in EAPCA-DSTree is more accurate, which gives DSTree a good starting point in exact search. Moreover, fewer data files are visited since similar time series are clustered better into fewer nodes.



(a) EAPCA-dstree (b) EPLA-dstree

FIGURE 11. Time series similarity comparison in nodes.



(a) Lower bound tightness (b) Upper bound tightness

FIGURE 12. Bound tightness.

We use random walk synthetic data to evaluate the time series similarity in the leaf nodes. The results is represented in Fig. 11. From Fig. 11, we can conclude that the time series in EPLA-DSTree leaf nodes is more similar that the ones in EAPCA-DSTree leaf nodes.

D. LOWER BOUND TIGHTNESS

We test the tightness of the proposed lower bound estimation approach. We measure the *lower bound tightness* by the ratio of the estimated lower bound distance against the minimum distance from a query to all time series indexed in a node. This ratio is between 0 and 1, the larger, the better. We collected this information during the processing of exact search.

E. SCALABILITY

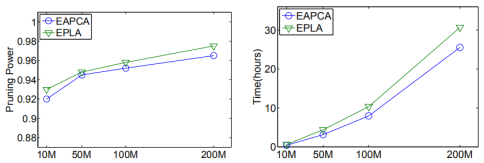
To compare the scalability of EPLA-DSTree with EAPCA-DSTree, we use the data generator provided by iSAX2.0 to generate 4 data sets containing 10 million, 50 million, 100 million and 200 million time series, respectively. Each time series is of length 256. In all experiments, the leaf capacity ψ is set to 5,000. The results of pruning power and index building time are shown in Figure 12.

Figure 12(a) shows that EPLA-DSTree has a higher pruning power than EAPCA-DSTree. Moreover, in all the methods, the pruning power is larger on bigger data sets. When a data set has more time series, the time series in each leaf node are more similar. Consequently, more irrelevant time series can be pruned, and fewer data files are needed to visit to find the most similar time series.

Figure 12(b) shows that the building time of all indexes are roughly linear to the data set size. The building time of EPLA-DSTree is a little longer than that of EAPCA-DSTree

F. SEARCH EFFICIENCY

To compare the search efficiency of EPLA-DSTree with EAPCA-DSTree, we used the synthetic data set of 200 million time series of length 256 each, which is used in the scalability test. The leaf capacity was set to 5,000. 100 time



(a) Pruning power (b) Building time

FIGURE 13. Search efficiency.

series were used as the queries, half of which were picked randomly from the data set, and the rest were generated randomly. The results are shown in Figure 13. EPLA-DSTree has a better pruning power than EAPCA-DSTree. Since the index size is not large (less than 100 MB), we hold the whole index in the main memory during searching.

V. RELATED WORK

With the rapid development of Internet of Things (IoS), sensor data and sensor applications attract more and more attention [20]–[22], [33]. There are many research directions concerning of IoS systems. Researchers focus on sensor data quality, reliability of IoS systems [20], [23], [36], [37], sensor data prediction [33] and so on. 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 [7] as the underlying search index. Faloutsos *et al.* [6] later allowed subsequence matching and proposed the FRM framework for indexing time series.

The subsequent work focused on two major aspects: new dimensionality reduction techniques (assuming that Euclidean distance is the underlying measure) and index building techniques based on dimensionality reduction techniques. Existing dimensionality reduction techniques include SVD [8], DFT [1], DWT [4], PLA [13], PAA [10], APCA [11] and CP [2]. 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) time series. In order to guarantee no-false-dismissals during the similarity search, the metric distance function must satisfy the lower bounding lemma [6].

Among all the reduction methods, SVD is accurate, but, at the same time, costly in both time and space, since SVD needs to calculate eigenvectors and store large matrices using extra space. Furthermore, APCA [11] and CP [2] are the two state-of-the-art reduction approaches. Keogh *et al.* [11] indicated that APCA outperforms DFT, DWT, and PAA in terms of the pruning power by orders of magnitude.

Approaches to building indexes can be categorized into two types. First, the traditional multi-dimensional index approaches, like R-tree, are used without modification [1], [6], [10]. Second, the R-tree method is modified according to the representation of time series [5], [11]. Since APCA contains both mean values and right ending points in the approximate representation, Keogh *et al.* [11] redefined the MBR (Minimal Bound Rectangle) according to APCA. PLA represents time series by disjoint lines. Each line is represented by two parameters: slope and intercept. Chen *et al.* [5] proposed a new MBR definition accordingly.

Most of the previous index approaches can be split into two phases: dimension reduction first and then building the index. Representing time series approximately is independent from building the index. Most recently, a new family of index aRpproaches, iSAX [15] and iSAX 2.0 [3], were proposed based on the representation technique, named SAX, which is a symbolic representation for time series that allows dimensionality reduction and indexing with a lower bounding distance measure.

In recent years, some scalable similarity search approaches are proposed. In [16], the authors proposed coconut, which used bulk-loading techniques that rely on sorting the underlying time series. In [31], the authors proposed a distributed approach for iSAX. In [17], [32], [34], interactive index techniques were proposed.

VI. CONCLUSION

In sensor-cloud systems, a huge amount of data series are collected by sensors. To process and analyze the sensor data efficiently, we propose EPLA-DSTree, which extends piecewise linear approximation on a DSTree index for whole matching on time series. Compared with other time series indexing approaches, EPLA-DSTree improves data locality of index and acquires a better lower bound of nodes. Furthermore, we proposed tighter bounds for PLA-DSTree nodes, which is critical for similarity search efficiency. We presented an extensive performance study on synthetic and real data sets to verify the effectiveness and efficiency of our new approach.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. Int. Conf. Found. Data Org. (FODO)*, 1993, pp. 69–84.
[2] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with Chebyshev polynomials," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 599–610.
[3] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "ISAX 2.0: Indexing and mining one billion time series," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 58–67.
[4] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proc. 15th Int. Conf. Data Eng.*, Mar. 1999, pp. 126–133.
[5] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. Yu, "Indexable PLA for efficient similarity search," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2007, pp. 435–446.
[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1994, pp. 419–429.
[7] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.
[8] K. Kanth, D. Agrawal, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1998, pp. 166–176.
[9] E. Keogh, "A decade of progress in indexing and mining large time series databases," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2006, p. 1268.
[10] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, Aug. 2001.

[11] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2001, pp. 151–162.
[12] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 102–111.
[13] E. Keogh and M. J. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1998, pp. 239–243.
[14] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery (DMKD)*, 2003, pp. 2–11.
[15] J. Shieh and E. Keogh, "ISAX: Indexing and mining terabyte sized time series," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 623–631.
[16] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas, "Coconut: A scalable bottom-up approach for building data series indexes," in *Proc. VLDB Endowment*, 2018, vol. 11, no. 6, pp. 677–690.
[17] K. Zoumpatianos, S. Idreos, and T. Palpanas, "ADS: The adaptive data series index," *VLDB J.*, vol. 25, no. 6, pp. 843–866, Dec. 2016.
[18] E. Keogh and L. Wei, "Semi-supervised time series classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2006, pp. 748–753.
[19] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 1033–1040.
[20] H. Gao, C. Liu, Y. Li, and X. Yang, "V2 VR: Reliable hybrid-network-oriented V2 V data transmission and routing considering RSUs and connectivity probability," *IEEE Trans. Intell. Transp. Syst.*, early access, Apr. 13, 2020, doi: 10.1109/TITS.2020.2983835.
[21] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *Mobile Netw. Appl.*, vol. 25, no. 4, pp. 1233–1248, Aug. 2020, doi: 10.1007/s11036-020-01535-1.
[22] X. Yang, S. Zhou, and M. Cao, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: The product-attribute perspective from user reviews," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 376–390, Apr. 2020.
[23] S. Deng, Z. Xiang, P. Zhao, J. Taheri, H. Gao, J. Yin, and A. Y. Zomaya, "Dynamical resource allocation in edge for trustable Internet-of-Things systems: A reinforcement learning method," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6103–6113, Sep. 2020, doi: 10.1109/TII.2020.2974875.
[24] Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang, "A data-adaptive and dynamic segmentation index for whole matching on time series," *Proc. VLDB Endow.*, vol. 6, no. 10, pp. 793–804, 2013.
[25] D. Shasha, "Tuning time series queries in finance: Case studies and recommendations," *IEEE Data Eng. Bull.*, vol. 22, no. 2, pp. 40–46, 1999.
[26] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 947–956.
[27] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2006, pp. 459–468.
[28] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, pp. 1–13.
[29] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical data prediction for real-world wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 8, pp. 2231–2244, Aug. 2015.
[30] *Source Codes*. [Online]. Available: https://github.com/jennielili/EPLA-DSTree
[31] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas, "DPiSAX: Massively distributed partitioned iSAX," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 1135–1140.
[32] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Indexing for interactive exploration of big data series," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2014, pp. 1555–1566.
[33] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4532–4542, May 2020.
[34] K. Zoumpatianos, S. Idreos, and T. Palpanas, "RINSE: Interactive data series exploration with ADS+," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1912–1915, Aug. 2015.
[35] *Datasets*. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/
[36] T. Wang, G. Zhang, M. Z. A. Bhuiyan, A. Liu, W. Jia, and M. Xie, "A novel trust mechanism based on fog computing in sensor–cloud system," *Future Gener. Comput. Syst.*, vol. 109, pp. 573–582, Aug. 2020.
[37] Y.Liang, T. Wang, Z. A. Bhuiyan, and A. Liu, "Research on coupling reliability problem in sensor-cloud system," in *Proc. SpaCCS Workshops*, 2017, pp. 468–478.
[38] P. Paraskevopoulos, T.-C. Dinh, Z. Dashdorj, T. Palpanas, and L. Serafini, "Identification and characterization of human behavior patterns from mobile phone data," in *Proc. D4D Challenge Session, NetMob*, 2013, pp. 1–13.
[39] K. Zoumpatianos and T. Palpanas, "Data series management: Fulfilling the need for big sequence analytics," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 1677–1678.

BIN XIE received the B.S. degree in computing technology from the National University of Defense Technology, Changsha, China, in 2004, and the M.S. degree in computing science from the Huadong Institution of Computing Technology, Shanghai, China, in 2007. He is currently pursuing the Ph.D. degree with the Nanjing University of Science and Technology. His research interests include big data processing, trajectory trace computing, and data mining.

QIUHONG LI received the Ph.D. degree from Fudan University, in 2014. She has published eight articles in refereed international journals and conference proceedings. Her research interests include database, data mining, and astronomical data processing.

YANG WANG received the Ph.D. degree from Fudan University, in 2015. His research interests include database, data mining, and astronomical data processing.

PENG WANG received the B.S. degree in mathematics from Nankai University, in 2001, and the Ph.D. degree from Fudan University, in 2007. He is currently an Associate Professor with the School of Computer Science, Fudan University. He has published more than 30 articles in refereed international journals and conference proceedings. His research interests include database, data mining, and series data processing.