ε-Approximation to Data Streams in Sensor Networks

Guohua Li, Jianzhong Li, Hong Gao
School of Computer Science and Technology
Harbin Institute of Technology, China
Email: liguohua.hit@gmail.com, {lijzh, honggao}@hit.edu.cn

Abstract—**The rapid development in processor, memory, and radio technology have contributed to the furtherance of decentralized sensor networks of small, inexpensive nodes that are capable of sensing, computation, and wireless communication. Due to the characteristic of limited communication bandwidth and other resource constraints of sensor networks, an important and practical demand is to compress time series data generated by sensor nodes with precision guarantee in an online manner. Although a large number of data compression algorithms have been proposed to reduce data volume, their offline characteristic or super-linear time complexity prevents them from being applied directly on time series data generated by sensor nodes. To remedy the deficiencies of previous methods, we propose an optimal online algorithm GDPLA for constructing a disconnected piecewise linear approximation representation of a time series which guarantees that the vertical distance between each real data point and the corresponding fit line is less than or equal to ε. GDPLA not only generates the minimum number of segments to approximate a time series with precision guarantee, but also only requires linear time $O(n)$ bounded by a constant coefficient 6, where unit 1 denotes the time complexity of comparing the slopes of two lines. The low cost characteristic of our method makes it the popular choice for resource-constrained sensor networks. Extensive experiments on a real dataset have been conducted to demonstrate the superior compression performance of our approach.**

I. I

Technological advances in sensing and low-power embedded communication technologies have made possible scenarios in which a large number of battery-powered sensor nodes are deployed to sense the physical world and form a self-organized Wireless Sensor Network (WSN). These sensors monitor various parameters continuously, such as temperature, humidity, illuminance, *etc*. For instance, GreenOrbs [1], a collaborative research project to study long-term large-scale WSNs in a forest, is a continuously monitoring system with up to 330 nodes deployed in the wild. To obtain near real-time, accurate and continuous observations, a high sampling frequency is desirable as it provides a detailed and accurate underlying data distribution. It is apparent that this could lead to a data glut, which will impose a serious burden on storing and transmitting the data. Therefore, how to reduce the amount of data while not compromising the mission of applications is of profound practical significance.

Data collection is an important problem in sensor networks [2]–[5], and data compression is a popular way to reduce data volume. A number of compression methods have been developed, such as wavelets [6], [7], discrete fourier transform [8]–[10], and discrete cosine transform [11]. However, due to limited power, memory and communication ability of sensor networks, and the high time and space complexity of these approaches, they cannot be seamlessly tailored for sensor networks.

Recently, some work [12]–[14] have addressed the problem of approximating time series data using piecewise linear approximation (PLA) techniques, which divide the time series data into a few segments, and the data in each segment is then approximated by a line segment. There are two types of PLA, connected PLA and disconnected PLA. Connected (Disconnected) PLA implies that two adjacent line segments share (do not share) the same endpoint. For clarity, we will call the problems, which input a time series and return a piecewise linear representation composed of connected (disconnected) line segments, as segment problems. Generally, the segment problems can be classified into two classes defined as follows.

- **Space-bounded PLA-Construction Problem**: Given a time series X of length $|X|$, an error metric $E(X, \widetilde{X})$ defined between X and $\widetilde{X}$, which is composed of disconnected (connected) line segments as a piecewise linear representation of X, and an upper limit L on the number of segments, find a set of L segments as $\widetilde{X}$, a piecewise linear approximation of X, to minimize $E(X, \widetilde{X})$.
- **Error-bounded PLA-Construction Problem**: Given a time series X of length $|X|$, an error metric $E(X, \widetilde{X})$ defined between X and $\widetilde{X}$, which is composed of disconnected (connected) line segments as a piecewise linear representation of X, and a maximum error tolerance ε (> 0), find $\widetilde{X}$ of the shortest length for which $E(X, \widetilde{X})$ is at most ε.

For ease of presentation, Space-bounded PLA-Construction Problem and Error-bounded PLA-Construction Problem are respectively called as S-PLA problem and E-PLA problem for short. A basic optimal algorithm based on dynamic programming can be proposed to solve the S-PLA problem. However, the high cost in time and space and the offline characteristic of the optimal solution prevent it from being applied directly on data streams, especially in resource-constrained sensor networks. Most solutions of the E-PLA problem, however, are often online and have low time and space complexity, which make them favorably received in several scientific communities.

In this paper, we thoroughly investigate the E-PLA problem [12]–[14], which has attracted much attention in the database community. To the best of our knowledge, the previous solutions on the E-PLA problem have the following shortcoming. For disconnected piecewise linear approximation, there does not exist an optimal algorithm, which generates the minimum number of line segments for approximating a time series with precision guarantee, within linear time. Aimed at overcoming this shortcoming, we propose an optimal online algorithm.

The main idea of GDPLA is briefly stated as follows. Given a single time series S and the maximum allowable error bound ε, GDPLA greedily scans the time series in order, approximating data points in the current segment until a new data point arrives and there does not exist a segment to approximate the data points including the newly arrived data point and the already seen but not compressed data points, with the aforementioned ε-bounded error requirement. Then a new line segment is created starting from the above-mentioned new data point. In summary, the contributions of this paper are shown as follows.

1) We formulate the single-sensor data compression problem as the E-PLA problem, and propose the algorithm GDPLA, an online algorithm for constructing a disconnected piecewise linear approximation version of a time series which guarantees that the distance between each real data point and the corresponding fit line in the vertical direction (*i.e.* the L_∞ norm) is less than or equal to ε. Notable features of GDPLA include (i) GDPLA is an optimal algorithm in the sense that it generates the minimum number of segments approximating a time series with precision guarantee, (ii) GDPLA has linear time complexity $O(n)$ bounded by a constant coefficient 6. To the best of our knowledge, it has the lowest time complexity in comparison with the previous piecewise linear approximation methods on time series data.
2) We present a detailed theoretical analysis for GDPLA including (i) the proof of optimality in the sense that it generates the minimum number of line segments, and (ii) the proof of having linear time complexity $O(n)$ bounded by a constant coefficient 6.
3) Last but not least, we conduct abundant experiments to evaluate GDPLA with a real dataset. Our evaluation demonstrates superior compression performance of GDPLA over the previous piecewise linear approximation approaches.

The rest of this paper is organized as follows. Section II introduces problem formulation. The design and analysis of GDPLA are presented in Section III. In Section IV, the extensive experiment results are shown to evaluate GDPLA. The related work is described in Section V. Finally, Section VI concludes this paper.

II. P F

In this section, we present the preliminaries of our work, which include some notations and problem formulation.

A. Notations

Let $S^{(n)}$ =< $x[t_1], x[t_2], \ldots, x[t_n]$ > be a finite time series, where $t_1, t_2, \cdots, t_n$ are an ordered sequence of time and $X = \{x[t_k], (k = 1, 2, \cdots, n)\}$ is an ordered list of the measurements generated by a sensor node at $t = t_k$ $(k = 1, 2, ..., n)$. Let $S[t_a : t_b]$ denote a subseries of $S^{(n)}$ from time t_a to time t_b $(1 \le a < b \le n)$, *i.e.*, $S[t_a : t_b]$ =< $x[t_a], x[t_{a+1}], \ldots, x[t_b]$ >. A line segment is represented by a tuple $((t^{(a)}, x[t^{(a)}]), (t^{(b)}, x[t^{(b)}]))$ where $t^{(a)} < t^{(b)}$, $(t^{(a)}, x[t^{(a)}])$ and $(t^{(b)}, x[t^{(b)}])$ are two endpoints of the line segment. The line segment $((t^{(a)}, x[t^{(a)}]), (t^{(b)}, x[t^{(b)}]))$ is the approximation representation of $S[t_a : t_b]$, if (1) $t^{(a)} \le t_a$ and $t^{(b)} \ge t_b$; and (2) the maximum vertical distance between the data points $(t_i, x[t_i])$ $(a \le i \le b)$ and the line segment $((t^{(a)}, x[t^{(a)}]), (t^{(b)}, x[t^{(b)}]))$ is less than or equal to some preset value ε. The approximation value of the i-th data points $(t_a \le t_i \le t_b)$ of the time series $S^{(n)}$ is calculated as follows.

$$\widetilde{x}[t_i] = x[t^{(a)}] + \frac{x[t^{(b)}] - x[t^{(a)}]}{t^{(b)} - t^{(a)}}(t_i - t^{(a)})$$

In this paper, we expect to use a set of disconnected line segments to approximate $S^{(n)}$, such that the error of an approximate value over the corresponding real value is bounded by some preset value ε. Therefore, we only need to record two endpoints for each line segment, thereby reducing the overhead of recording and transmitting all the data points.

Assume that H line segments, *i.e.*, $G = \{g^{(1)}, g^{(2)}, \ldots, g^{(H)}\}$ will be generated to approximate $S^{(n)}$, where $g^{(1)}$ approximates the data points $\{\ (t_i, x[t_i]), i = 1, 2, \ldots, j_1\ \}$, and $g^{(h)}$, where $h \in [2, H]$, approximates the data points $\{\ (t_i, x[t_i]), i = j_{h-1} + 1, j_{h-1} + 2, \ldots, j_h\ \}$, and hence $j_H = n$.

Let $\widetilde{S}^{(n)}$ =< $\widetilde{x}[t_1], \widetilde{x}[t_2], \ldots, \widetilde{x}[t_n]$ > be an approximation version of the finite time series $S^{(n)}$, reconstructing by G. We quantify the error using L_∞ norm, defined as:

$$E(S^{(n)}, \widetilde{S}^{(n)}) = \max_{1 \le k \le n} |x[t_k] - \widetilde{x}[t_k]| \tag{1}$$

B. Problem formulation

Based on the above-mentioned notations, our optimization problem can be formulated as follows.

Problem 1. (Error-bounded PLA-Construction Problem) *Given a time series $S^{(n)}$, an error bound ε, and the error function $E(S^{(n)}, \widetilde{S}^{(n)})$, find the set G composed of disconnected line segments, with $E(S^{(n)}, \widetilde{S}^{(n)}) \le \varepsilon$ to minimize $|G|$.*

III. G D P L A

In this section, we first propose an online algorithm GDPLA for Error-bounded PLA-Construction Problem. We then analyze the time and space complexity, and optimality of GDPLA.

A. Method Overview

We consider a data stream generated by any non-sink sensor node. Without loss of generality, we use n_0 to denote a non-sink sensor node. Let S =< $x[t_1], x[t_2], x[t_3], \ldots$ > be the infinite time series generated by the sensor node n_0. When the first data point $(t_1, x[t_1])$ arrives, we store it. When $(t_2, x[t_2])$ arrives, it has no trouble finding a line segment satisfying the error bound requirement, *i.e.*, the maximum vertical distance between $(t_i, x[t_i])$ $(i = 1, 2)$ and the approximation line segment is at most ε. When $(t_3, x[t_3])$ arrives, we check whether $(t_3, x[t_3])$ can be approximated together with $(t_1, x[t_1])$ and $(t_2, x[t_2])$ by a line segment within the preset error bound ε. If so, we store $(t_3, x[t_3])$. Otherwise, we output the line segment, which satisfies (i) it approximates $(t_1, x[t_1])$ and $(t_2, x[t_2])$ within the preset error bound ε, and (ii) it minimizes the mean square error for the data points approximated by the current line segment, *i.e.*, $(t_1, x[t_1])$ and $(t_2, x[t_2])$. And then we start a new line segment starting from a new data point, *i.e.* $(t_3, x[t_3])$. The above process is repeated when a new data point arrives.

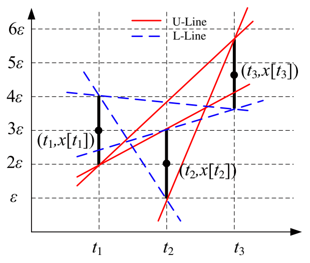


Fig. 1. For $S[t_1:t_3]$, there are three 2ε*-bound line segments* (three vertical black line segments) corresponding to three data points $(t_i, x[t_i])$ $(i = 1, 2, 3)$, three U-Lines represented by red lines and three L-Lines represented by blue lines.

B. Several Lemmas for GDPLA

In the following, we define two types of straight lines, U-Line (the abbreviation of upper line) and L-Line (the abbreviation of lower line) and a special class of line segment 2ε-bound line segment.

U-Line (**L-Line**) *Given a time series* $S[t_a : t_b] =< x[t_a], x[t_{a+1}], \ldots, x[t_b] >$*, a line* u_{pq} (l_{pq})*, passing through the point* $(t_p, x[t_p] - \varepsilon)$ $((t_p, x[t_p] + \varepsilon))$ *and the point* $(t_q, x[t_q] + \varepsilon)$ $((t_q, x[t_q] - \varepsilon))$ *where* $a \le p < q \le b$*, is called a U-line (an L-line) of* $S[t_a:t_b]$.

2ε**-bound line segment** *Given a time series* $S[t_a : t_b] =< x[t_a], x[t_{a+1}], \ldots, x[t_b] >$*, a line segment* $l_{t_k,2\varepsilon}$*, connecting the point* $(t_k, x[t_k]-\varepsilon)$ *and the point* $(t_k, x[t_k]+\varepsilon)$ *where* $a \le k \le b$*, is called a* 2ε*-bound line segment of* $S[t_a:t_b]$.

Based on the above definitions, we then define two special kinds of lines, UI-Line and LI-Line.

UI-Line (**LI-Line**) *Given a time series* $S[t_a : t_b] =< x[t_a], x[t_{a+1}], \ldots, x[t_b] >$*, a U-line* u_{pq} *(an L-line* l_{pq}*) where* $a \le p < q \le b$*,* u_{pq} (l_{pq}) *is a UI-Line (an LI-Line) of* $S[t_a:t_b]$ *if and only if it intersects with all line segments* $l_{t_k,2\varepsilon}$ $(a \le k \le b)$*, i.e.,* $u_{pq} \cap l_{t_k,2\varepsilon} \ne \emptyset (l_{pq} \cap l_{t_k,2\varepsilon} \ne \emptyset)$ $(a \le k \le b)$.

Remark: The aforementioned definitions of four kinds of lines and one kind of line segment are based on the time series $S[t_a:t_b]$. In order to thoroughly understand these definitions, we give an example as shown in Figure 1. For $S[t_1 : t_3]$, there are three data points $(t_i, x[t_i])$ $(i = 1, 2, 3)$ generated by some sensor node, three 2ε-bound line segments $l_{t_k,2\varepsilon}$ $(k = 1, 2, 3)$ depicted by vertical black line segments, three U-Lines represented by red lines and three L-Lines represented by blue lines. While there are only one UI-Line u_{12} and one LI-Line l_{23}. We will demonstrate that there are only one UI-Line and one LI-Line for any time series $S[t_a : t_b]$ in Lemma 2. The main function of the UI-Line and the LI-Line of a time series $S[t_a:t_b]$ is to determine whether a newly arrived data point can be approximated together with the data points observed so far but not compressed yet, and the detail is depicted in Lemma 3. In the following, we present several important lemmas, which are the theoretical foundation of GDPLA.

Lemma 1. *Given a time series* $S[t_a : t_b]$ *containing data points* $(t_k, x[t_k])$ $(a \le k \le b)$*, a U-Line* $u_{p_0q_0}$ *(an L-Line* $l_{p_0q_0}$*) where* $a \le p_0 < q_0 \le b$*. If* $u_{p_0q_0}$ $(l_{p_0q_0})$ *is a UI-Line (an LI-Line), then the following properties hold:*

(P1) All $b-a+1$ *data points* $(t_k, x[t_k])$ $(a \le k \le b)$ *are within* ε *from* $u_{p_0q_0}$ $(l_{p_0q_0})$ *in the vertical direction.*

(P2) The straight line $u_{p_0q_0}$ $(l_{p_0q_0})$ *has the minimum (maximum) slope among all U-Lines (L-Lines) in* $S[t_a : t_b]$*, i.e.,* $k_{u_{p_0q_0}} \le k_{u_{p'q'}}$ $(k_{l_{p_0q_0}} \ge k_{l_{p'q'}})$ *where* $a \le p' < q' \le b$ *and* $k_{u_{p_0q_0}}$ $(k_{l_{p_0q_0}})$ *denotes the slope of the straight line* $u_{p_0q_0}$ $(l_{p_0q_0})$.

(P3) When a new data point $(t_{b+1}, x[t_{b+1}])$ *arrives, if*

$$x[t_{b+1}] - \varepsilon > u_{p_0q_0}\big|_{t=t_{b+1}} \; (x[t_{b+1}] + \varepsilon < l_{p_0q_0}\big|_{t=t_{b+1}}), \tag{2}$$

where $u_{p_0q_0}\big|_{t=t_{b+1}}$ $(l_{p_0q_0}\big|_{t=t_{b+1}})$ *denotes the value of* $u_{p_0q_0}$ $(l_{p_0q_0})$ *when* $t = t_{b+1}$*, then the data points* $\{(t_k, x[t_k]) \mid a \le k \le b+1\}$ *cannot be approximated by a single straight line satisfying the error bound requirement.*

Proof: Based on the definition of UI-Line, we can derive that $u_{p_0q_0} \cap l_{t_k,2\varepsilon} \ne \emptyset$ $(a \le k \le b)$, which means that $u_{p_0q_0}$ has the property (P1).

Given an arbitrary U-Line $u_{p'q'}$ of $S[t_a : t_b]$ where $a \le p' < q' \le b$, since $u_{p_0q_0} \cap l_{t_m,2\varepsilon} \ne \emptyset$ $(m = p', q')$, then $u_{p_0q_0}|_{t=t_{p'}} \ge x[t_{p'}]-\varepsilon$ (*i.e.,* $u_{p_0q_0}|_{t=t_{p'}} \ge u_{p'q'}|_{t=t_{p'}}$), and $u_{p_0q_0}|_{t=t_{q'}} \le x[t_{q'}]+\varepsilon$ (*i.e.,* $u_{p_0q_0}|_{t=t_{q'}} \le u_{p'q'}|_{t=t_{q'}}$). This implies that the slope of $u_{p_0q_0}$ is less than or equal to that of $u_{p'q'}$, *i.e.,* $k_{u_{p_0q_0}} \le k_{u_{p'q'}}$. Thus, $u_{p_0q_0}$ has the property (P2).

Now we assume that $u_{p_0q_0}$ does not have (P3). Let $u_{p_1q_1}$ be the straight line that all $b-a+2$ data points $(t_k, x[t_k])$ $(a \le k \le b+1)$ are within ε from $u_{p_1q_1}$ in the vertical direction. Then, $u_{p_1q_1}\big|_{t=t_{b+1}} \ge x[t_{b+1}] - \varepsilon$. From the known condition $u_{p_0q_0}\big|_{t=t_{b+1}} < x[t_{b+1}] - \varepsilon$, we then obtain

$$u_{p_1q_1}\big|_{t=t_{b+1}} \ge x[t_{b+1}] - \varepsilon > u_{p_0q_0}\big|_{t=t_{b+1}}. \tag{3}$$

Since $u_{p_0q_0}$ is a UI-Line of $S[t_a : t_b]$ and $u_{p_1q_1}$ intersects $l_{t_k,2\varepsilon}$ $(k = p_0, q_0)$, we then obtain that the value of $u_{p_1q_1}$ is greater than or equal to that of $u_{p_0q_0}$ at $t = t_{p_0}$, *i.e.,*

$$u_{p_1q_1}\big|_{t=t_{p_0}} \ge u_{p_0q_0}\big|_{t=t_{p_0}}, \tag{4}$$

and the value of $u_{p_1q_1}$ is less than or equal to that of $u_{p_0q_0}$ at $t = t_{q_0}$, *i.e.,*

$$u_{p_1q_1}\big|_{t=t_{q_0}} \le u_{p_0q_0}\big|_{t=t_{q_0}}. \tag{5}$$

From (4) and (5), we can easily derive that the slope of $u_{p_1q_1}$ is less than or equal to that of $u_{p_0q_0}$, *i.e.,*

$$k_{u_{p_1q_1}} \le k_{u_{p_0q_0}}. \tag{6}$$

However, from (3) and (5), we have

$$k_{u_{p_1q_1}} > k_{u_{p_0q_0}}. \tag{7}$$

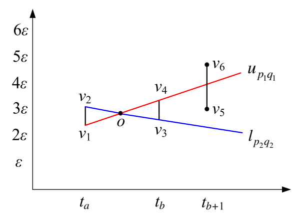
We derive a contradiction from (6) and (7).
Thus, $u_{p_1q_1}$ does not exist and $u_{p_0q_0}$ has the property (P3).

The proof that if $l_{p_0q_0}$ is an LI-Line, then it also has the properties (P1), (P2) and (P3) is quite similar. ■

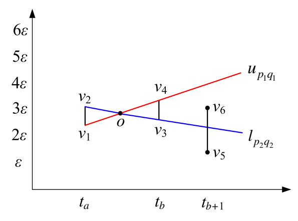
Based on Lemma 1, we have the following lemma.

Lemma 2. *Given a time series* $S[t_a : t_b]$ *containing data points* $(t_k, x[t_k])$ $(a \le k \le b)$*, if there exists one UI-Line* $u_{p_0q_0}$ *(LI-Line* $l_{p_0q_0}$*) in* $S[t_a : t_b]$*, then there does not exist any other UI-Line (LI-Line) in* $S[t_a : t_b]$.

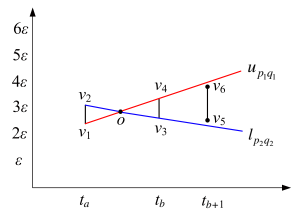
Proof: We prove the lemma by contradiction. Assume that there exists another UI-Line $u_{p_1q_1}$ (in $S[t_a : t_b]$), which is different from $u_{p_0q_0}$. Since $u_{p_0q_0}$ is a UI-Line and $u_{p_1q_1}$ is



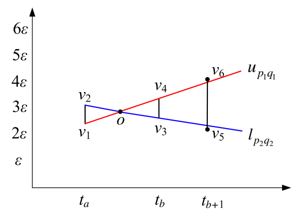
(a) case 1: $u_{p_1q_1} \cap l_{t_{b+1},2\varepsilon} \neq \emptyset$



(b) case 2: $l_{p_2q_2} \cap l_{t_{b+1},2\varepsilon} \neq \emptyset$



(c) case 3: $l_{t_{b+1},2\varepsilon} \subseteq [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}]$



(d) case 4: $[l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] \subseteq l_{t_{b+1},2\varepsilon}$

Fig. 2. When a new data point $(t_{b+1}, x[t_{b+1}])$ arrives, a new line segment $l_{t_{b+1},2\varepsilon}$ (v_5v_6) is created. The relationships among $l_{t_{b+1},2\varepsilon}$, $u_{p_1q_1}$ and $l_{p_2q_2}$ are depicted by four different cases.

also a U-line, then, based on the property (P2) of Lemma 1, we obtain

$$k_{u_{p_0q_0}} \leq k_{u_{p_1q_1}}. \tag{8}$$

Similarly, since $u_{p_1q_1}$ is a UI-Line and $u_{p_0q_0}$ is also a U-line, then, based on the property (P2) of Lemma 1, we obtain

$$k_{u_{p_1q_1}} \leq k_{u_{p_0q_0}}, \tag{9}$$

where the equal sign is true if and only if $u_{p_1q_1}$ is exactly the same as $u_{p_0q_0}$. From Equations (8) and (9), we obtain that $k_{u_{p_1q_1}} = k_{u_{p_0q_0}}$. Then we derive that $u_{p_1q_1}$ is exactly the same as $u_{p_0q_0}$. This is a contradiction. Thus, $u_{p_1q_1}$ does not exist and $u_{p_0q_0}$ is the only UI-Line in $S[t_a : t_b]$.

The proof that if there exists one LI-Line $l_{p_0q_0}$ in $S[t_a : t_b]$, then there does not exist any other LI-Line in $S[t_a : t_b]$ is quite similar. ■

The following lemma indicates how to find the new UI-Line (LI-Line) when a newly arrived data point satisfies a certain condition.

Lemma 3. *Given a time series $S[t_a : t_b]$ of data points $(t_k, x[t_k])$ $(a \leq k \leq b)$, there exist one UI-Line $u_{p_1q_1}$ and one LI-Line $l_{p_2q_2}$ in $S[t_a : t_b]$, where $a \leq p_j < q_j \leq b$ $(j = 1, 2)$. When a new data point $(t_{b+1}, x[t_{b+1}])$ arrives, if the intersection of the vertical line segments $l_{t_{b+1},2\varepsilon} = [x[t_{b+1}] - \varepsilon, x[t_{b+1}] + \varepsilon]$ and $[l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}]$ is non-empty, i.e., $[x[t_{b+1}] - \varepsilon, x[t_{b+1}] + \varepsilon] \cap [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] \neq \emptyset$, then we can calculate the UI-Line and the LI-Line according to the following four different cases.*

(1) When $l_{p_2q_2}|_{t=t_{b+1}} < x[t_{b+1}] - \varepsilon \leq u_{p_1q_1}|_{t=t_{b+1}} \leq x[t_{b+1}] + \varepsilon$ ($l_{p_2q_2}|_{t=t_{b+1}} < u_{p_1q_1}|_{t=t_{b+1}}$ is obvious because the intersection of the UI-Line and the LI-Line (point o) is before $x[t_b + 1]$), the UI-Line of $S[t_a : t_{b+1}]$ is the same as that of $S[t_a : t_b]$, and we only need to calculate the new LI-Line, and its expression is as follows.

$$y = k_l(t - t_{b+1}) + x[t_{b+1}] - \varepsilon$$
$$\text{where} \quad k_l = \max_{a \leq j \leq b} \frac{(x[t_j]+\varepsilon)-(x[t_{b+1}]-\varepsilon)}{t_j - t_{b+1}} \tag{10}$$

(2) When $x[t_{b+1}] - \varepsilon \leq l_{p_2q_2}|_{t=t_{b+1}} \leq x[t_{b+1}] + \varepsilon < u_{p_1q_1}|_{t=t_{b+1}}$, the LI-Line of $S[t_a : t_{b+1}]$ is the same as that of $S[t_a : t_b]$, and we only need to calculate the new UI-Line, and its expression is as follows.

$$y = k_u(t - t_{b+1}) + x[t_{b+1}] + \varepsilon$$
$$\text{where} \quad k_u = \min_{a \leq j \leq b} \frac{(x[t_j]-\varepsilon)-(x[t_{b+1}]+\varepsilon)}{t_j - t_{b+1}} \tag{11}$$

(3) When $l_{p_2q_2}|_{t=t_{b+1}} < x[t_{b+1}] - \varepsilon < x[t_{b+1}] + \varepsilon < u_{p_1q_1}|_{t=t_{b+1}}$, we need to calculate the new UI-Line and LI-Line, and their expressions are shown in Equations (10) and (11) respectively.

(4) When $x[t_{b+1}] - \varepsilon \leq l_{p_2q_2}|_{t=t_{b+1}} < u_{p_1q_1}|_{t=t_{b+1}} \leq x[t_{b+1}] + \varepsilon$, the UI-Line and the LI-Line of $S[t_a : t_{b+1}]$ are the same as that of $S[t_a : t_b]$.

Proof: For case (1), since $u_{p_1q_1}$ is the UI-Line of $S[t_a : t_b]$, then we obtain that $u_{p_1q_1} \cap l_{t_k,2\varepsilon} \neq \emptyset$ $(a \leq k \leq b)$. Figure 2(a) shows that $l_{p_2q_2}|_{t=t_{b+1}} < x[t_{b+1}] - \varepsilon \leq u_{p_1q_1}|_{t=t_{b+1}} \leq x[t_{b+1}] + \varepsilon$, which implies that $u_{p_1q_1} \cap l_{t_{b+1},2\varepsilon} \neq \emptyset$, and then we have $u_{p_1q_1} \cap l_{t_k,2\varepsilon} \neq \emptyset$ $(a \leq k \leq b + 1)$. Thus, $u_{p_1q_1}$ is also the UI-Line of $S[t_a : t_{b+1}]$.

Assume that v_1 (v_4) is the intersection point of $u_{p_1q_1}$ and $l_{t_a,2\varepsilon}$ ($l_{t_b,2\varepsilon}$), and v_2 (v_3) is the intersection point of $l_{p_2q_2}$ and $l_{t_a,2\varepsilon}$ ($l_{t_b,2\varepsilon}$), and the point o is the intersection point of $u_{p_1q_1}$ and $l_{p_2q_2}$ as shown in Figure 2(a). Since $u_{p_1q_1}$ and $l_{p_2q_2}$ are the UI-Line and LI-Line in $S[t_a : t_b]$ respectively, then the points $(t_k, x[t_k] - \varepsilon)$ $(a \leq k \leq b)$ are below or on the fold line v_1ov_3, and the points $(t_k, x[t_k]+\varepsilon)$ $(a \leq k \leq b)$ are above or on the fold line v_2ov_4. When the new data point $(t_{b+1}, x[t_{b+1}])$ arrives, we connect the point v_5 $(t_{b+1}, x[t_{b+1}]-\varepsilon)$ and the point o and then obtain a line l, which can always be rotated around point v_5 counter-clockwise until it touches the first vertex $(t_{p_0}, x[t_{p_0}]+\varepsilon)$ $(a \leq p_0 \leq b)$. It is obvious that the line l is an L-Line $l_{p_0(b+1)}$, and all the points $(t_j, x[t_j] - \varepsilon)$ $(a \leq j \leq b + 1)$ are below or on it, and all the points $(t_j, x[t_j] + \varepsilon)$ $(a \leq j \leq b+1)$ are above or on it. This implies that $l_{p_0(b+1)} \cap l_{t_j,2\varepsilon} \neq \emptyset$ $(a \leq j \leq b + 1)$. Thus $l_{p_0(b+1)}$ is the LI-Line of $S[t_a : t_{b+1}]$. The proof for cases (2) and (3) are quite similar.

From the definitions of UI-Line and LI-Line and Lemma 2, we can easily obtain that the UI-Line and the LI-Line of $S[t_a : t_{b+1}]$ are the same as that of $S[t_a : t_b]$ for case (4). ■

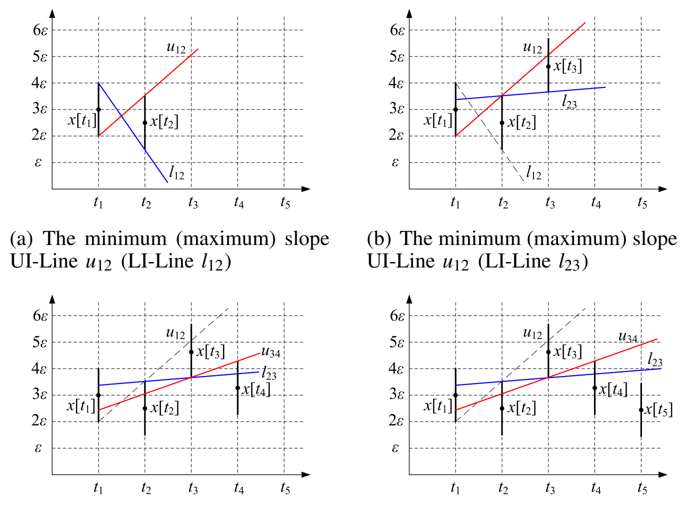
In the following, we present an example to illustrate Lemmas 1, 2 and 3.

Example

After the data points $(t_1, x[t_1])$ and $(t_2, x[t_2])$ arrive (Figure 3(a)), u_{12} (l_{12}) is the UI-Line (LI-Line) of the time series $S[t_1 : t_2]$. When $(t_3, x[t_3])$ arrives, since $(t_3, x[t_3])$ can be represented by u_{12}, then we needn't to calculate the new UI-Line based on Lemma 2. However, l_{12} needs to be update as l_{23} which is the LI-Line of $S[t_1 : t_3]$ according to Lemma 3. Based on Lemma 1, the data points $\{(t_k, x[t_k]) \mid 1 \leq k \leq 3\}$ are all within ε from the line u_{12} (l_{23}) in the vertical direction.

Similarly, when $(t_4, x[t_4])$ arrives, we needn't to calculate the new LI-Line based on Lemma 2, and based on Lemma 3, u_{12} is updated as u_{34}, which is the UI-Line of $S[t_1 : t_4]$.

After the arrival of $(t_5, x[t_5])$, $x[t_5] + \varepsilon < l_{23}|_{t=t_5}$ holds, based on Lemma 1, we need to end the current line segment $g^{(h)}$ with $g^{(h)} = u_{34}|_{t_1 \to t_4}$ ($l_{23}|_{t_1 \to t_4}$), which denotes the line segment connecting $(t_1, u_{34}|_{t=t_1})$ $((t_1, l_{23}|_{t=t_1}))$ and $(t_4, u_{34}|_{t=t_4})$ $((t_4, l_{23}|_{t=t_4}))$, and start the new line segment $g^{(h+1)}$ at $t = t_5$.



(a) The minimum (maximum) slope UI-Line u_{12} (LI-Line l_{12})

(b) The minimum (maximum) slope UI-Line u_{12} (LI-Line l_{23})

(c) The minimum (maximum) slope UI-Line u_{34} (LI-Line l_{23})

(d) The minimum (maximum) slope UI-Line u_{34} (LI-Line l_{23})

Fig. 3. Illustrating Lemmas 1, 2 and 3 with an example

Based on Lemmas 1 and 3, we obtain that when a new data point $(t_{b+1}, x[t_{b+1}])$ arrives, if $[x[t_{b+1}] - \varepsilon, x[t_{b+1}] + \varepsilon] \cap [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] = \emptyset$, then we end the current line segment at $t = t_b$ and start a new line segment from $t = t_{b+1}$. However, if $[x[t_{b+1}] - \varepsilon, x[t_{b+1}] + \varepsilon] \cap [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] \neq \emptyset$, then we can calculate the UI-Line and LI-Line of $S[t_a, t_{b+1}]$ based on Lemma 3. An ordinary approach for calculating the UI-Line and LI-Line of $S[t_a, t_{b+1}]$ is to revisit the points that have been seen but have not been approximated yet. However, it eventually results in the time complexity being quadratic with respect to the length of the time series.

To tackle the above issue, two important lemmas are presented for updating the UI-Line or LI-Line, and then we prove that the time complexity (calculation overhead) of our approach is linear with respect to the length of the time series based on the two above-mentioned lemmas.

Lemma 4. *Given a time series $S[t_p : t_q]$, let $u_{min} = u_{p_1q_1}$ ($l_{max} = l_{p_2q_2}$) denote the UI-Line (LI-Line) of $S[t_p : t_q]$ respectively, where $a \leq p_j < q_j \leq b$ ($j = 1, 2$). When $(t_{q+1}, x[t_{q+1}])$ arrives, if $[x[t_{q+1}] - \varepsilon, x[t_{q+1}] + \varepsilon] \cap [l_{max}|_{t=t_{q+1}}, u_{min}|_{t=t_{q+1}}] \neq \emptyset$ and $[l_{max}|_{t=t_{q+1}}, u_{min}|_{t=t_{q+1}}] \nsubseteq [x[t_{q+1}] - \varepsilon, x[t_{q+1}] + \varepsilon]$ hold, then we only need to update k_u (k_l) as follows.*

$$k_u = \min_{p_1 \leq j \leq q} \frac{(x[t_j] - \varepsilon) - (x[t_{q+1}] + \varepsilon)}{t_j - t_{q+1}} \tag{12}$$

$$k_l = \max_{p_2 \leq j \leq q} \frac{(x[t_j] + \varepsilon) - (x[t_{q+1}] - \varepsilon)}{t_j - t_{q+1}} \tag{13}$$

Proof: We prove it by contradiction. Based on Lemma 3, we need to update k_u in cases (2) and (3) where $x[t_{q+1}] + \varepsilon < u_{p_1q_1}|_{t=t_{q+1}}$. Assume that $k_u = k_{u_{p'(q+1)}}$ where $p \leq p' < p_1$ and $u_{p'(q+1)}$ is the UI-Line of $S[t_p : t_{q+1}]$. Since $u_{p'(q+1)}$ is the UI-Line of $S[t_p : t_{q+1}]$, then $u_{p'(q+1)}|_{t=t_{p'}} \leq u_{p_1q_1}|_{t=t_{p'}}$, and since $u_{p_1q_1}$ is the UI-Line of $S[t_p : t_q]$, then $u_{p'(q+1)}|_{t=t_{p_1}} \geq u_{p_1q_1}|_{t=t_{p_1}}$. This implies that

$$k_{u_{p'(q+1)}} \geq k_{u_{p_1q_1}}. \tag{14}$$

In addition, since $u_{p'(q+1)}$ is the UI-Line of $S[t_p : t_{q+1}]$, then $u_{p'(q+1)}$ must be greater than or equal to $u_{p_1q_1}$ at $t = t_{p_1}$, and less than or equal to $u_{p_1q_1}$ at $t = t_{q_1}$. This implies that

$$k_{u_{p'(q+1)}} \leq k_{u_{p_1q_1}}. \tag{15}$$

Based on Equations (14) and (15), we obtain that $k_{u_{p'(q+1)}} = k_{u_{p_1q_1}}$. Since $u_{p'(q+1)}|_{t=t_{q+1}} = x[t_{q+1}] + \varepsilon < u_{p_1q_1}|_{t=t_{q+1}}$ in cases (2) and (3) of Lemma 3, then $u_{p'(q+1)}|_{t=t_{p_1}} < u_{p_1q_1}|_{t=t_{p_1}} = x[t_{p_1}] - \varepsilon$, and then we obtain $u_{p'(q+1)} \cap l_{t_{p_1}, 2\varepsilon} = \emptyset$. This contradicts the fact that $u_{p'(q+1)}$ is the UI-Line of $S[t_p : t_{q+1}]$ based on Lemma 1. Thus, Equation (12) holds. The proof that Equation (13) holds is quite similar. ■

Lemma 4 demonstrates that we can narrow the scan space when we need to update the UI-Line or LI-Line. In the following, we present two special arrays, upper endpoint array and lower endpoint array, which have a significant effect for reducing the scan space even further.

Definition 1 (Upper (Lower) Endpoint Array). *Given a time series $S[t_a : t_b]$, the upper (lower) endpoint array is composed of ordered data points $(t_i, x[t_i] + \varepsilon)$ $((t_i, x[t_i] - \varepsilon))$ where $a \leq i \leq b$.*

For the elements of the Upper (Lower) Endpoint Array, we divide it into two parts according to the following definition.

Definition 2 (Upper (Lower) Prune Set). *Given a time series $S[t_a : t_b]$ and the corresponding Upper (Lower) Endpoint Array $U[t_a : t_b] = \{(t_i, x[t_i] + \varepsilon) | a \leq i \leq b\}$ ($L[t_a : t_b] = \{(t_i, x[t_i] - \varepsilon) | a \leq i \leq b\}$), a point $(t_i, x[t_i] + \varepsilon)$ $((t_i, x[t_i] - \varepsilon))$ is called an upper (a lower) prune point if it satisfies that $\exists p, q \geq 1$, the point $(t_{i+p}, x[t_{i+p}] + \varepsilon)$ $((t_{i+p}, x[t_{i+p}] - \varepsilon))$ is below (above) the line passing through two points $(t_{i-q}, x[t_{i-q}] + \varepsilon)$ and $(t_i, x[t_i] + \varepsilon)$ $((t_{i-q}, x[t_{i-q}] - \varepsilon)$ and $(t_i, x[t_i] - \varepsilon))$ in the vertical direction where $a \leq i - q < i < i + p \leq b$. The set composed of all the upper (lower) prune points in $U[t_a : t_b]$ ($L[t_a : t_b]$) is called the Upper (Lower) Prune Set of $U[t_a : t_b]$ ($L[t_a : t_b]$).*

Lemma 5. *Given a time series $S[t_a : t_b]$ of data points $(t_k, x[t_k])$ ($a \leq k \leq b$), there exist one UI-Line $u_{p_1q_1}$ and one LI-Line $l_{p_2q_2}$ in $S[t_a : t_b]$, where $a \leq p_j < q_j \leq b$ ($j = 1, 2$). When a new data point $(t_{b+1}, x[t_{b+1}])$ arrives, if the condition of case (1) (case (2)) is satisfied, then the newly calculated LI-Line (UI-Line) is impossible to pass through any data point in the Upper (Lower) Prune Set of $S[t_a : t_{b+1}]$.*

Proof: We prove the lemma by contradiction. For case (1), assume that the newly calculated LI-Line L_0 passes through one data point $(t_i, x[t_i] + \varepsilon)$ in the Upper Prune Set of $S[t_a : t_{b+1}]$. Since $(t_i, x[t_i] + \varepsilon)$ is an upper prune point, then according to Definition 2 we have

$$x[t_{i+p}] + \varepsilon < l_1|_{t=t_{i+p}} \tag{16}$$

where l_1 is the line passing through $(t_{i-q}, x[t_{i-q}] + \varepsilon)$ and $(t_i, x[t_i] + \varepsilon)$ ($p, q \geq 1$ and $a \leq i - q < i < i + p \leq b + 1$). In addition, since L_0 is the LI-Line of $S[t_a : t_{b+1}]$, then the point $(t_{i-q}, x[t_{i-q}] + \varepsilon)$ is above L_0, and then we obtain $k_{l_1} < k_{L_0}$, where k_{l_1} (k_{L_0}) is the slope of l_1 (L_0). Since $(t_i, x[t_i] + \varepsilon)$ is the intersection point of l_1 and L_0 and $k_{l_1} < k_{L_0}$, then we have

$$L_0|_{t=t_{i+p}} \geq l_1|_{t=t_{i+p}}. \tag{17}$$

Algorithm 1 GDPCA Algorithm

```
INPUT: Time series $S$ =<$x[t_1], x[t_2], \ldots$ >, error bound $\varepsilon > 0$. OUTPUT: G(S) composed of disconnected line segments within $\varepsilon$ of $S$ in the vertical direction .
1: G(S) ← ⟨⟩;
2: $start \leftarrow t_1$, $end \leftarrow t_2$, $h \leftarrow 1$;
3: InitDLinkList(U), InitDLinkList(L);
4: while the $j$-th data point arrives do
5:     if Equation (2) in Lemma 1 occurs then
6:         $end \leftarrow t_{j-1}$;
7:         append $g^{(h)} \leftarrow (k_0,(t_0, X_0))|_{start \rightarrow end}$ to G(S);
           /* $(k_0,(t_0, X_0))|_{start \rightarrow end}$ denotes the line segment such that (i) its time interval is $[start, end]$, (ii) it passes through the intersection point of UI-Line and LI-Line, (iii) it minimizes the mean square error for the data points observed in $[start, end]$. */
8:         $start \leftarrow t_j$;
9:         $h \leftarrow h + 1$;
10:        InitDLinkList(U), InitDLinkList(L);
11:    else if case (1) of Lemma 3 occurs then
12:        UpdateAndPruneUp(U, $t_j$, $x[t_j]$);
13:        CalculateLILine(U, $t_j$, $x[t_j]$);
14:    else if case (2) of Lemma 3 occurs then
15:        UpdateAndPruneLow(L, $t_j$, $x[t_j]$);
16:        CalculateUILine(L, $t_j$, $x[t_j]$);
17:    else if case (3) of Lemma 3 occurs then
18:        UpdateAndPruneUp(U, $t_j$, $x[t_j]$);
19:        CalculateLILine(U, $t_j$, $x[t_j]$);
20:        UpdateAndPruneLow(L, $t_j$, $x[t_j]$);
21:        CalculateUILine(L, $t_j$, $x[t_j]$);
22:    else if case (4) of Lemma 3 occurs then
23:        UpdateAndPruneUp(U, $t_j$, $x[t_j]$);
24:        UpdateAndPruneLow(L, $t_j$, $x[t_j]$);
25:    end if
26: end while
27: G(S) ← $\langle g^{(1)}, g^{(2)}, \ldots \rangle$
28: return $G(S)$.
```

Since $L_0 \cap l_{t_{i+p}, 2\varepsilon} \neq \emptyset$, then we have

$$L_0|_{t=t_{i+p}} \leq x[t_{i+p}] + \varepsilon. \tag{18}$$

From Equations (17) and (18), we obtain that $x[t_{i+p}] + \varepsilon \geq l_1|_{t=t_{i+p}}$. This is a contradiction to Equation (16). Thus, the newly calculated LI-Line is impossible to pass through any data point in the Upper Prune Set of $S[t_a : t_{b+1}]$. The proof that if the condition of case (2) is satisfied, then the newly calculated UI-Line is impossible to pass through any data point in the Lower Prune Set of $S[t_a : t_{b+1}]$ is quite similar. ∎

Based on the above-mentioned Lemmas, we present GDPCA as shown in Algorithm 1. We employ a doubly linked list U (L) to manage the points with the form of $(t_i, x[t_i] + \varepsilon)$ $((t_i, x[t_i] - \varepsilon))$. The UpdateAndPruneUp(U, t_j, $x[t_j]$) procedure in line 12 is used to delete the upper prune points (based on Lemma 5) from all points with the form of $(t_i, x[t_i] + \varepsilon)$ that have been seen but have not been compressed when a new data point $(t_j, x[t_j])$ arrives. The pseudo code of UpdateAndPruneUp(U, t_j, $x[t_j]$) is given in Algorithm 2. In line 13, CalculateLILine(U, t_j, $x[t_j]$) calculates the new LI-Line, and the details are shown in Algorithm 3, where lines 5-10 are the core procedure.

Algorithm 2 UpdateAndPruneUp(U, t_j, $x[t_j]$)

```
INPUT: Doubly linked list U, $t_j$, $x[t_j]$.
OUTPUT: U satisfying that there does not exist any upper prune point in U.
/* Each node of U has four field: time, data, the prior pointer, the next pointer */
1: Add $(t_j, x[t_j] + \varepsilon)$ as the last node of U;
2: LinkNode *p;
3: p = tail → prior;            /* tail is the tail pointer of U*/
4: while (p.time, p.data) is an upper prune point do
5:     p →prior→next = p →next;
6:     p →next→prior = p →prior;
7:     p = p →prior;
8: end while
9: return U
```

Algorithm 3 CalculateLILine(U, t_j, $x[t_j]$)

```
INPUT: Doubly linked list U, $t_j$, $x[t_j]$.
OUTPUT: The new LI-Line.
1: LinkNode *p1,*p2;
2: p1 = head, p2 = p1 → next;
3: $l_1$ ← The line passing through (p1.time, p1.data) and $(t_j, x[t_j] - \varepsilon)$;
4: $l_2$ ← The line passing through (p2.time, p2.data) and $(t_j, x[t_j] - \varepsilon)$;
5: while $k_{l_1} \leq k_{l_2}$ do
6:     p1 = p2;
7:     p2 = p2 → next;
8:     $l_1$ ← The line passing through (p1.time, p1.data) and $(t_j, x[t_j] - \varepsilon)$;
9:     $l_2$ ← The line passing through (p2.time, p2.data) and $(t_j, x[t_j] - \varepsilon)$;
10: end while
11: Calculate the new LI-Line passing through (p1.time, p1.data) and $(t_j, x[t_j] - \varepsilon)$.
12: return $(k_{l_1}, (t_j, x[t_j] - \varepsilon))$; /*$(k_{l_1}, (t_j, x[t_j] - \varepsilon))$ denotes the new LI-Line passing through $(t_j, x[t_j] - \varepsilon)$ with the slope $k_{l_1}$.*/
```

Suppose that after the *while* condition runs m times, $k_{l_1} \leq k_{l_2}$ does not hold, now we only need to prove that $k_{l_1} \geq k_{l_3}$ where l_3 denotes any line passing through (p3.time,p3.data) and $(t_j, x[t_j] - \varepsilon)$ where p3 is a LinkNode of U and $p2.time < p3.time < t_j$, then the line l_1 is the new LI-Line. By contradiction, if it does not hold, then there exists p4 in U and $p2.time < p4.time < t_j$, satisfying $k_{l_1} < k_{l_4}$ where l_4 denotes the line passing through (p4.time,p4.data) and $(t_j, x[t_j] - \varepsilon)$. Then we obtain that $k_{l_1} > k_{l_2}$ and $k_{l_1} < k_{l_4}$ hold, it is easy to know that (p2.time,p2.data) is an upper prune point according to Definition 2. This contradicts the fact that there does not exist any upper prune point in U in Algorithm 2. Therefore, Algorithm 3 can correctly find out the new LI-Line. The procedures UpdateAndPruneLow(L, t_j, $x[t_j]$) and CalculateUILine(L, t_j, $x[t_j]$) are quite similar with UpdateAndPruneUp(U, t_j, $x[t_j]$) and CalculateLILine(U, t_j, $x[t_j]$), and we omit them due to space limitation.

C. Analysis of GDPCA

1) Space and Time Complexity: The main space and time complexity of GDPCA lies in four subprocedures in Algorithm 1. We focus on the complexity analysis of UpdateAndPruneUp(U, t_j, $x[t_j]$) and CalculateLILine(U, t_j, $x[t_j]$), and that of UpdateAndPruneLow(L, t_j, $x[t_j]$) and CalculateUILine(L, t_j, $x[t_j]$) is quite similar.

Suppose GDPCA outputs H line segments $\{g^{(h)} | 1 \leq h \leq H\}$ to approximate the time series $S[t_1 : t_n]$, and $g^{(h_0)}$ $(1 \leq h_0 \leq H)$ is the line segment that approximates the most data points in $S[t_1 : t_n]$. Let $Num(g^{(h_0)})$ be the number of data points approximated by $g^{(h_0)}$, then the number of the elements of U (L) is at most $Num(g^{(h_0)})$. Thus the space complexity is $O(max_{1 \leq h_0 \leq H}\{Num(g^{(h_0)})\})$. In the worst case, the whole data stream can be approximated by only one line segment. In such situations, the space complexity is $O(n)$. However, the worst case are rare. Usually, $max_{1 \leq h_0 \leq H}\{Num(g^{(h_0)})\}$ is smaller than any finite constant, and therefore the space complexity is $O(1)$ in most cases.

Theorem 1 (**Time Complexity**). *The time complexity of GDPCA for approximating a time series $S[t_1 : t_n]$ of n data points is $O(n)$.*

Proof: For UpdateAndPruneUp(U, t_j, $x[t_j]$), its runtime is mainly dominated by the *while* loop shown in Algorithm 2 from line 4 to line 8. Suppose that the *while* condition runs M times, where M_1 times loop condition is true and M_2 times loop condition is false. Since there only exists one loop condition unsatisfied for each newly arrived data point, and therefore $M_2 = n$, and then we obtain $M = M_1 + M_2 = M_1 + n$. And M_1 times loop condition satisfied implies that a total of M_1 prune operations are performed in UpdateAndPruneUp(U, t_j, $x[t_j]$). Intrinsically, whether a data point is an upper prune point is implemented by comparing the slopes of two lines. For ease of presentation, we claim that the time complexity of one comparison operation is 1, then the total time complexity of UpdateAndPruneUp(U, t_j, $x[t_j]$) is at most $M_1 + n$. Similarly, the total time complexity of UpdateAndPruneLow(L, t_j, $x[t_j]$) is at most $M_1' + n$.

For CalculateLILine(U, t_j, $x[t_j]$), its runtime is mainly dominated by the *while* loop shown in Algorithm 3 from line 5 to line 10. Suppose that the *while* condition runs N times, where N_1 times loop condition is true and N_2 times loop condition is false. Since there only exists one loop condition unsatisfied for each newly arrived data point, and therefore $N_2 = n$. In addition, from the global perspective, all the upper prune points are not checked when updating LI-Lines, then we obtain that there are at most $n - M_1$ points checked when updating LI-Lines, and therefore $N_1 \leq n - M_1$. Then the total time complexity of CalculateLILine(U, t_j, $x[t_j]$) is at most $2n - M_1$. Similarly, the total time complexity of CalculateUILine(L, t_j, $x[t_j]$) is at most $2n - M_1'$. Based on the above analysis, we obtain that the time complexity of GDPCA for $S[t_1 : t_n]$ is $O(M_1 + n + M_1' + n + 2n - M_1 + 2n - M_1') = O(6n)$, *i.e.*, $O(n)$. ■

GDPCA not only has the superiority of time complexity $O(n)$, but also has the following strong property that it is optimal in the sense that it generates the minimum number of line segments approximating a time series with the required precision. The above solid facts make GDPCA the popular choice for resource-constrained sensor networks. In what follows, we will present the proof of optimality in details.

Theorem 2 (**Optimality**). *Let $S^{(n)} = S[t_1 : t_n]$ be an arbitrary time series that must be approximated with a disconnected piecewise linear approximation (DPWL) such that for all $k = 1, 2, ..., n$, $|\widetilde{x}[t_k] - x[t_k]| \leq \varepsilon$. If GDPCA produces a DPWL representation with $G(S) = (g^{(1)}, g^{(2)}, ..., g^{(H)})$, then no other valid DPWL representation with $H' < H$ line segments exists.*

Proof: By contradiction, let $G'(S)$ be a valid representation with $G'(S) = (l^{(1)}, l^{(2)}, ..., l^{(H')})$ where $H' < H$. Assume that the end time of $g^{(h)}$ and $l^{(h')}$ are t_{j_h} and $t_{y_{h'}}$ $(1 \leq h \leq H,\ 1 \leq h' \leq H')$, respectively, and then $t_{j_H} = t_{y_{H'}} = t_n$. In the following, we prove that $y_m \leq j_m$ for all $m = 1, 2, ..., H'-1$. By mathematical induction. When $m = 1$, $y_1 \leq j_1$ holds, otherwise, if $y_1 > j_1$ holds, then the subseries $S[t_1 : t_{j_1+1}]$ $(\subseteq S[t_1 : t_{y_1}])$ can be approximated by $l^{(1)}$ within ε. However, from the property (P3) of Lemma 1, we obtain that the subseries $S[t_1 : t_{j_1+1}]$ cannot be approximated by any straight line within ε on the L_∞ norm error metric. This is a contradiction. Thus, $y_1 \leq j_1$ holds.

Now, assume that $y_p \leq j_p$ holds where $0 < p < H' - 1$. We prove that $y_{p+1} \leq j_{p+1}$ holds. By contradiction, if $y_{p+1} > j_{p+1}$ holds, then $y_p \leq j_p < j_{p+1} < y_{p+1}$, which implies that the subseries $S[t_{j_p} : t_{j_{p+1}+1}]$ can be approximated by $l^{(p)}$ within ε. Similarly, from the property (P3) of Lemma 1, we obtain $S[t_{j_p} : t_{j_{p+1}+1}]$ cannot be approximated by any straight line within ε on the L_∞ norm error metric. This is a contradiction. Thus, $y_{p+1} \leq j_{p+1}$ holds. Thus, we prove that $y_m \leq j_m$ for all $m = 1, 2, ..., H'-1$. Specially, $y_{H'-1} \leq j_{H'-1}$ holds. Since $y_{H'-1} \leq j_{H'-1} < j_{H'} < j_H = y_{H'} = n$, then the subseries $S[t_{j_{H'-1}} : t_{j_{H'-1}+1}]$ $(\subseteq S[t_{y_{H'-1}} : t_{y_{H'}}])$ can be approximated by $l^{(H')}$ within ε. However, from the property (P3) of Lemma 1, we obtain that the subseries $S[t_{j_{H'-1}} : t_{j_{H'-1}+1}]$ cannot be approximated by any straight line within ε on the L_∞ norm error metric. This is a contradiction. Thus, no other valid DPWL representation for $S^{(n)} = S[t_1 : t_n]$ with $H' < H$ line segments exists. ■

IV. E

In this section, we further evaluate our approach through extensive experiments.

A. Experimental Setting

We use the real dataset: Intel-Berkeley Temperature dataset, which contains temperature measurements by 54 sensors deployed on a single floor of Intel-Berkeley Research lab over the period of one month [15]. We mainly measure the following three metrics.

1) *Compression ratio.* It is defined as the total number of data points in the original time series divided by the total number of points to represent the approximation version of the original time series.
2) *Average error.* It is defined as $\frac{\sum_{i=1}^{n} |x_i - \widetilde{x_i}|}{n}$, where n is the total number of data points in the original time series, x_i is the original value and $\widetilde{x_i}$ is the approximated value of x_i.

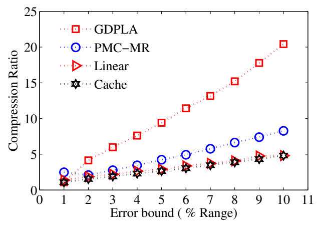


Fig. 4. ε vs. compression ratio

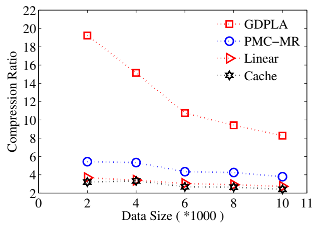


Fig. 5. Data size vs. compression ratio

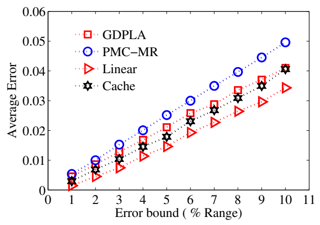


Fig. 6. ε vs. average error

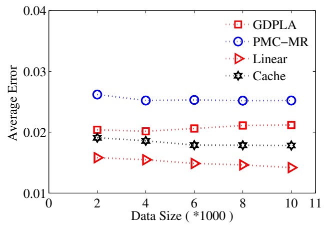


Fig. 7. Data size vs. average error

3) *Average processing time*. It is defined as the total running time divided by the total number of data points in the original time series.

The experiments are conducted on an Intel Core2 Duo machine with a 2.93 GHz processor and 2GB RAM, and the programs are written in Python.

B. Comparison with Alternative Approaches

We mainly compare the performance of GDPLA against PMC-MR [16], Cache [17] and Linear filter [12]. PMC-MR greedily scans the series in order, inserting data items in the first bucket until the difference between the maximum and the minimum entries in the bucket exceeds 2ε, at which point a new bucket is created. All the items in the first bucket is approximated by the average of the max and the min. Cache selects the value of the first data point as the approximate value, and the next incoming data point will be filtered if it is within ε from the first data point. A new data point is recorded only if it is not within ε from the first data point. Linear filter predicts that new data points will fall in the proximity of a line segment whose slope is determined by the first two data points. Whenever a new data point falls more than ε away from the predicted line segment, a new line segment is started.

C. Performance Evaluation

1) Results on compression ratio: Figure 4 depicts that when the original data size is 8000, how the compression ratios of GDPLA, PMC-MR, Linear and Cache change with the increase of error bound ε. As expected, compression ratios increase with the increase of ε for all the approaches, and when $\varepsilon = 0.1$, the compression ratio of our approach GDPLA is 2.5, 4.2 and 5.0 times larger than that of PMC-MR, Linear and Cache respectively. The reason is that GDPLA is the optimal algorithm in the sense that it generates the minimum number of line segments when using the piecewise linear approximation technique to approximate a time series with the required precision. When $\varepsilon = 0.05$, GDPLA only needs to transmit around 10% of the original sample size, and transmit around 5% of the original sample size when $\varepsilon = 0.1$. Figure 5 plots when $\varepsilon = 0.05$, how the compression ratio achieved by GDPLA, PMC-MR, Linear and Cache change when the size of the dataset is varied from 2000 to 10000. Compression ratio drops as the increase of the size of dataset for all the approaches. This may be related to the distribution of data.

2) Results on average error: Figure 6 shows the average errors of GDPLA, PMC-MR, Linear and Cache with the increase of error bound ε when the original data size is 8000. We can observe that the average errors are roughly proportional to ε for all the approaches. Figure 7 shows that

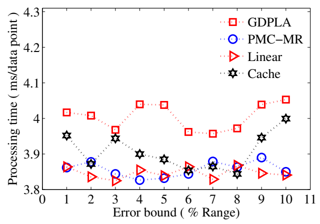


Fig. 8. ε vs. Processing time

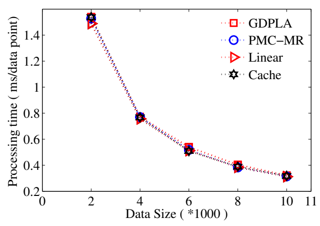


Fig. 9. Data Size vs. Processing time

the average errors of GDPLA, PMC-MR, Linear and Cache change when the size of dataset is varied from 2000 to 10000, where $\varepsilon = 0.05$. It is easy to observe that the average errors of the four approaches are less than 0.03, and the average errors do not change much when the size of dataset is varied from 2000 to 10000. The average error of GDPLA is slightly more than that of Cache and less than that of PMC-MR.

3) Results on average processing time: Figure 8 compares the processing time per data point of GDPLA, PMC-MR, Linear and Cache when ε is varied from 0.01 to 0.1, where the data size is 8000. The processing time per data point of GDPLA is slightly greater than that of other approaches. Figure 9 plots the processing time per data point of GDPLA, PMC-MR, Linear and Cache when the data size is varied from 2000 to 10000, where $\varepsilon = 0.05$. We can find that there are no obvious difference in the processing time per data point of GDPLA, PMC-MR, Linear and Cache, and the processing time per data point of all approaches drop with the increase of data size. The reason for this may be that the system spends much time to transform python code to other code, which leads to the situation that there is not much difference in the total running time among GDPLA, PMC-MR, Linear and Cache.

Figure 10 plots the slope comparisons required by GDPLA when ε is varied from 0.01 to 0.05, data size is 8000. We can easily observe that the number of slope comparisons is less than 48000 (6*8000) for varying ε from 0.01 to 0.05. Figure 11 shows the slope comparisons required by GDPLA when data size is varied from 2000 to 10000, ε is equal to 0.05.

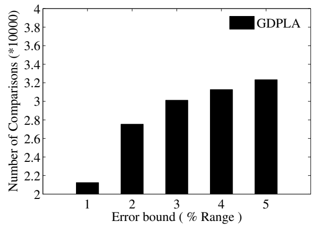


Fig. 10. ε vs. Number of Comparisons

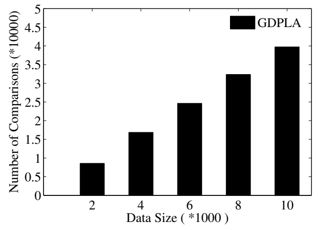


Fig. 11. Data size vs. Number of Comparisons

The results demonstrate that the number of slope comparisons is less than data size multiplied by 6, which validates our theoretical analysis.

V. R

There is a large body of research on time series segmentation in the database community. Due to space limitation, we are able to mention only those that are closely related to our work. Recently, there has been increasing interest in piecewise linear approximation (PLA) techniques in [12], [18]. The idea behind PLA is that a sequence of line segments can be used to represent a time series while maintaining a low approximation error. In general, the PLA techniques for time series can be categorized into two classes: offline algorithms and online algorithms. The offline algorithms are given the whole time series from the beginning and are required to output an answer to solve the problem. Jagadish *et al.* [19] present a dynamic programming algorithm (offline algorithm) to approximate a time series minimizing the L_2 error for a given amount of space (number of buckets) and the time complexity of dynamic programming algorithm is $O(n^2)$ where n is the length of the time series. Top-Down algorithms [12] work by considering every possible partitioning of the time series and splitting it at the best location with precision guarantee, and this process is repeated until some stopping criterion is met. Bottom-Up algorithms [12] start with $n/2$ segments to approximate the n-length time series, and then the algorithms begin to iteratively merge the lowest cost pair until a stop criteria is met.

Online algorithms handle the time series based on the data seen so far not the whole time series. Lazaridis *et al.* [16] propose an optimal online algorithm PMC-MR for constructing a piecewise constant approximation for a time series with the required precision guarantee. Liu *et al.* [18] use the PLA method to approximate the original time series, and the end points of each line segment must be the points in the original time series. On the whole, their algorithm can run in $O(n^2 logn)$ time and takes $O(n)$ memory space. Olston *et al.* [20] use a cache filter to eliminate the data points whose values lie inside $[x_0 - \varepsilon, x_0 + \varepsilon]$, where (t_0, x_0) is the first data point of the already observed but not compressed data points. When an incoming data point violates the error constraint, a new filtering interval is created. The idea of Linear filter is presented in [12], [21]. Linear filter approximates a time series by line segments whose slopes are defined by the first two data points they represent. Whenever a new data point falls more than ε away from the current line segment, a new line segment is started from the new data point.

The above online algorithms for approximating time series are not optimal for constructing a piecewise linear approximation of a time series which guarantees that the vertical distances between data points and the corresponding line segments are less than or equal to ε. To remedy this deficiency, we present an optimal online algorithm GDPLA, which uses piecewise linear approximation techniques to approximate a time series with the error constraint, and GDPLA only has a running time of $O(n)$.

VI. C

In this paper, we present a new data compression algorithm GDPLA designed for sensor networks. GDPLA uses a set of line segments to approximate the original time series, and guarantees that the vertical distances of line segments and the corresponding real data points are less than an error bound ε. More importantly, GDPLA is optimal in the sense that it generates the minimum number of line segments for approximating a time series with precision guarantee, and it only has a linear running time. Extensive experiments using a real dataset demonstrate that the compression performance of GDPLA is far superior to other approaches.

VII. A

This work is supported in part by the Key Program of the National Natural Science Foundation of China under grant No.61033015, the National Basic Research Program of China (973 Program) under Grant No. 2012CB316200, the National Natural Science Foundation of China (NSFC) under Grant No. 61190115, 60933001, 61173022, 61100030, and National Science Foundation (NSF) under grants No. CNS-1152001 and CNS-1252292.

R

[1] L. Mo, Y. He, Y. Liu. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. In *Sensys*, pages 99-112 , 2009.
[2] Z. Cai, S. Ji, J. He, and A. G. Bourgeois. Optimal Distributed Data Collection for Asynchronous Cognitive Radio Networks. In *ICDCS*, pages 245-254,2012.
[3] S. Ji and Z. Cai. Distributed Data Collection and Its Capacity in Asynchronous Wireless Sensor Networks. In *INFOCOM*, pages 2113-2121, 2012.
[4] Z. Cai, S. Ji, and J. Li. Data Caching Based Query Processing in Multi-Sink Wireless Networks. In *International Journal of Sensor Networks*, 11(2): 109-125, 2012.
[5] S. Cheng, J. Li and Z. Cai. $O(\epsilon)$-Approximation to Physical World by Sensor Networks, In *INFOCOM* , 2013
[6] K.P. Chan and A. W. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126-133, Washington, DC, 1999.
[7] M. N. Garofalakis and A. Kumar. Wavelet synopses for general error metrics. In *ACM Transactions on Database Systems*, 30(4):888-928, 2005.
[8] L. Keselbrener, S. Akselrod. Selective discrete Fourier transform algorithm for time-frequency analysis: method and application on simulated and cardiovascular signals. In *IEEE Trans Biomed Eng*, 43(8):789-802, 1996.
[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419-429, 1994.
[10] Y. Wu, D. Agrawal, and A. Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *CIKM*, pages 488-495, New York, 2000.
[11] V. Britanak and K. R. Rao. A new fast algorithm for the unified forward and inverse MDCT/MDST computation. In *Signal Processing*, Volume 82, Issue 3, pages 433-459, 2002.
[12] E. Keogh, S. Chu, D. Hart and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM*, pages 289-296, 2001.
[13] X. Liu, Z. Lin and H. Wang. Novel Online Methods for Time Series Segmentation. In *TKDE* 20(12): 1616-1626, 2008.
[14] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, W. Zwaenepoel. Online Piece-wise Linear Approximation of Numerical Streams with Precision Guarantees. In *PVLDB* 2(1): 145-156, 2009.
[15] http://db.csail.mit.edu/labdata/labdata.html.
[16] I. Lazaridis, and S.Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE* 2003.
[17] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD* 2003.
[18] C. Liu, K. Wu, and J. Pei. An energy efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. In *IEEE Transactions on Parallel and Distributed Systems*, 18:1010-1023, July 2007.
[19] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, 1998.
[20] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD* 2003.
[21] A. Jain, E. Y. Chang, Y. Wang. Adaptive Stream Resource Management Using Kalman Filters. In *SIGMOD* 2004.