# Novel $\varepsilon$-Approximation to Data Streams in Sensor Networks

Jianzhong Li, *Member, IEEE*, Guohua Li, and Hong Gao

**Abstract**—We are gradually moving into a realm where sensors, processors, memory and wireless transceivers would be seamlessly integrated together in the physical world and form a wireless sensor network. Such networks pose new challenges in data processing and transmission due to the characteristic of limited communication bandwidth and other resource constraints of sensor networks. To reduce the cost of storage, transmission and processing of time series data generated by sensor nodes, the need for more compact representations of time series data is compelling. 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. Motivated by these observations, we propose an optimal online algorithm GDPLA for constructing a disconnected piecewise linear approximation 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 $\varepsilon$. 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 a proper choice for resource-constrained WSNs. Extensive experiments on two real data sets have been conducted to demonstrate the superior compression performance of our method.

**Index Terms**—Sensor networks, data streams, $\varepsilon$-approximation, piecewise linear approximation

## 1 Introduction

Technological advances in the development of sensing and low-power embedded communication technologies have made possible scenarios in which a large number of battery-powered sensor nodes are deployed, both technologically and economically, to sense the physical world and form a self-organized distributed Wireless Sensor Network (WSN) [2], [3], [4], [5]. The nodes in a WSN 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 continuous monitoring system (collecting various sensory data including temperature, humidity, illuminance, and carbon dioxide titer) 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 easy to see how this could lead to a data glut: 330 sensor nodes, even recording a few tens of bytes per second would generate several gigabytes *each month*. A key challenge is how to collect these data from a sensor network efficiently and continuously for further processing and analysis. A naive approach is to instruct all nodes in the sensor network to send their readings at regular intervals to a base station. This approach, however, imposes a serious burden on storing and transmitting data and is significantly inconsistent with one of the major concerns for WSNs: energy conservation. Because sensor nodes have limited battery lifetime, and radio transmission is the primary consumer of energy, excessive transmission quickly depletes batteries, rendering the nodes useless. Therefore, how to reduce the amount of data while not compromising the mission of applications is a fundamental research problem.

A popular technique to significantly reduce communication cost in WSNs is to compress data during the process of data collection, which is an important problem in sensor networks. A number of compression methods have been developed, such as wavelets [6], [7], discrete fourier transform [8], [9], [10], and discrete cosine transform [11]. However, due to limited power, memory and communication ability of sensor nodes, and the high time and space complexities of these methods, they cannot be seamlessly tailored for WSNs.

Motivated by these observations, some new techniques [12], [13], [14] have been proposed to address the problem of approximating time series data using piecewise linear approximation (PLA) methods, which divide the time series data into a few segments, and then the data in each segment is approximated by a line segment. To the best of our knowledge, there doesn't have a PLA method which uses the minimum number of segments to represent a time series with precision guarantee in linear time. Aimed at this defect, we propose a new PLA method which uses the minimum number of segments to represent a time series with precision guarantee in linear time.

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 segmentation problems.

• *The authors are with the Department of Computer Science and Technology, Harbin Institute of Technology, PO Box 750, Harbin 150001, P.R. China E-mail: {lijzh, honggao}@hit.edu.cn, lghhit1@126.com.*

*Manuscript received 2 Nov. 2013; revised 25 Mar. 2014; accepted 5 May 2014. Date of publication 11 May 2014; date of current version 8 May 2015. Recommended for acceptance by Y. Liu.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TPDS.2014.2323056*

For the solutions of segment problems, there are two important measures: the number of segments and computational complexity (time and space complexities). Intuitively, for a solution of E-PLA problem, the smaller the number of segments and the lower the computational complexity it has, the better the representation it will be. The most recent methods that are similar to ours are PMC-MR [15], Cache [16] and Linear filter [12]. PMC-MR greedily scans a time series in order, inserting data items in the first bucket until the difference between the maximum and the minimum entries in the bucket exceeds $2\varepsilon$, 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 $\varepsilon$ from the first data point. A new data point is recorded only if it is not within $\varepsilon$ 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 $\varepsilon$ away from the predicted line segment, a new line segment is started. To the best of our knowledge, the above methods have the following shortcoming. they are all not 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 GDPLA.

The main idea of GDPLA is briefly stated as follows. Given a single time series $S$ and the maximum allowable error bound $\varepsilon$, GDPLA greedily scans the tim\e 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 $\varepsilon$-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 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 (DPWL) 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 $L_\infty$ norm) is less than or equal to $\varepsilon$. 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 using two real data sets and make direct comparisons against previously studied approximation techniques like PMC-MR [15], Cache [16] and Linear filter [12]. In both data sets our method presents superior compression performance over the previous piecewise linear approximation methods.

TABLE 1
Notations Used

| Symbol | Description |
|---|---|
| $t_k$ | The timestamp |
| $x[t_k]$ | The measurement generated by a sensor node at $t_k$ |
| $S^{(n)}$ | The time series $< x[t_1], x[t_2], \ldots, x[t_n] >$ |
| $\widetilde{S}^{(n)}$ | The approximation version of $S^{(n)}$ |
| $S[t_a : t_b]$ | The time series $< x[t_a], x[t_{a+1}], \ldots, x[t_b] >$ |
| $u_{pq}$ | The line passing the points $(t_p, x[t_p] - \varepsilon)$ and $(t_q, x[t_q] + \varepsilon)$ |
| $l_{pq}$ | The line passing the points $(t_p, x[t_p] + \varepsilon)$ and $(t_q, x[t_q] - \varepsilon)$ |
| $l_{t_k, 2\varepsilon}$ | The vertical line segment passing through the point $(t_k, x[t_k])$, and the length is $2\varepsilon$ |
| $k_{u_{pq}}$ ($k_{l_{pq}}$) | The slope of the line $u_{pq}$ ($l_{pq}$) |

The remainder of this paper is organized as follows. Section 2 introduces problem formulation. The design and analysis of GDPLA are presented in Section 3. In Section 4, the extensive experiment results are shown to evaluate GDPLA. The related work is described in Section 5. Finally, Section 6 concludes this paper.

## 2 Problem Formulation

In this section, we present the preliminaries of our work, which include some notations and problem formulation, and the notations used in this paper are shown in Table 1.

### 2.1 Notations

Let $S^{(n)} =< x[t_1], x[t_2], \ldots, x[t_n] >$ be a finite time series, where $t_1, t_2, \ldots, t_n$ are an ordered sequence of time and $X = \{x[t_k], (k = 1, 2, \ldots, n)\}$ is an ordered list of the measurements generated by a sensor node at $t = t_k$ $(k = 1, 2, \ldots, n)$. 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 $\varepsilon$. 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 $\varepsilon$. 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 $\widetilde{S}^{(n)} = lt; \widetilde{x}[t_1], \widetilde{x}[t_2], \ldots, \widetilde{x}[t_n] >$ be an approximation of the finite time series $S^{(n)}$, reconstructed by $G$. We quantify the error using $L_\infty$ norm, defined as

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

### 2.2 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 $\varepsilon$, and the error function $E(S^{(n)}, \widetilde{S}^{(n)})$, find a set $G$ composed of disconnected line segments, with $E(S^{(n)}, \widetilde{S}^{(n)}) \le \varepsilon$ to minimize $|G|$.*

## 3 Greedy Disconnected Piecewise Linear Approximation

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

### 3.1 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 $n_0$. 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 $\varepsilon$. 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 $\varepsilon$. 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 $\varepsilon$, 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 $(t_2, x[t_2])$. 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.

### 3.2 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 $2\varepsilon$-bound line segment.

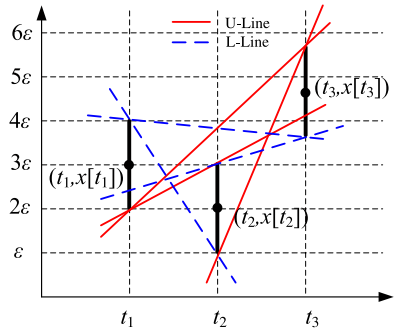



Fig. 1. For $S[t_1 : t_3]$, there are three $2\varepsilon$-*bound line segments* (three vertical black line segments) corresponding to three data points $(t_i, x[t_i])(i = 1, 2, 3)$, three U-Lines (red lines) and three L-Lines (blue lines).

**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 $S[t_a : t_b]$.*

**$2\varepsilon$-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 $2\varepsilon$-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 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 the 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 Fig. 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 $2\varepsilon$-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 at most 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 $u_{p_0 q_0}$ (an L-Line $l_{p_0 q_0}$) where $a \le p_0 < q_0 \le b$. If $u_{p_0 q_0}$ ($l_{p_0 q_0}$) 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 $\varepsilon$ from $u_{p_0 q_0}$ ($l_{p_0 q_0}$) in the vertical direction.*

*(P2) The straight line $u_{p_0 q_0}$ ($l_{p_0 q_0}$) has the minimum (maximum) slope among all U-Lines (L-Lines) in $S[t_a : t_b]$, i.e., $k_{u_{p_0 q_0}} \le k_{u_{p'q'}}$ $(k_{l_{p_0 q_0}} \ge k_{l_{p'q'}})$ where $a \le p' < q' \le b$ and $k_{u_{p_0 q_0}}$ $(k_{l_{p_0 q_0}})$ denotes the slope of the straight line $u_{p_0 q_0}$ ($l_{p_0 q_0}$).*

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

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

*where $u_{p_0q_0}|_{t=t_{b+1}}(l_{p_0q_0}|_{t=t_{b+1}})$ denotes the value of $u_{p_0q_0}$ $(l_{p_0q_0})$ when $t = t_{b+1}$, then there does not exist a UI-Line (an LI-Line) for $S[t_a : t_{b+1}]$.*

**Proof.** Based on the definition of UI-Line, we can derive that $u_{p_0q_0} \cap l_{t_k,2\varepsilon} \neq \emptyset (a \le k \le b)$, 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 $u_{p_0q_0} \cap l_{t_m,2\varepsilon} \neq \emptyset (m = p', q')$, 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 $u_{p_0q_0}|_{t=t_{q'}}[t_{q'}] + \varepsilon$ (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 $\varepsilon$ from $u_{p_1q_1}$ in the vertical direction. Then, $u_{p_1q_1}|_{t=t_{b+1}} \ge x[t_{b+1}] - \varepsilon$. From the known condition $u_{p_0q_0}|_{t=t_{b+1}} < x[t_{b+1}] - \varepsilon$, we then obtain

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

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.,

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

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.,

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

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.,

$$k_{u_{p_1q_1}} \le k_{u_{p_0q_0}}. \quad (6)$$

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

$$k_{u_{p_1q_1}} > k_{u_{p_0q_0}}. \quad (7)$$

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. □

From Lemma 1, It is easy to obtain that other U-Lines (L-Lines) whose slopes are more (less) than $u_{p_0q_0}$ $(l_{p_0q_0})$ cannot be a UI-Line (an LI-Line) in $S[t_a : t_b]$. Otherwise, it contradicts (P2) of Lemma 1. In addition, if there does not exist a UI-Line (an LI-Line) in $S[t_a : t_b]$, then it implies that the U-Line (L-Line) with the minimum (maximum) slope does not intersect all line segments $l_{t_k,2\varepsilon}$ $(a \le k \le b)$.

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

$$k_{u_{p_0q_0}} \le k_{u_{p_1q_1}}. \quad (8)$$

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

$$k_{u_{p_1q_1}} \le k_{u_{p_0q_0}}, \quad (9)$$

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 gives the answer to the question 'Under what circumstances does UI-Line or LI-Line exist for a time series $S[t_a : t_b]$?'

**Lemma 3.** *Given a time series $S[t_a : t_b]$ containing data points $(t_k, x[t_k])$ $(a \le k \le b)$, the following three statements are equivalent.*

*(C1) There exists a line segment $l_0$ intersects all line segments $l_{t_k,2\varepsilon}$ $(a \le k \le b)$, i.e., $l_0 \cap l_{t_k,2\varepsilon} \neq \emptyset (a \le k \le b)$.*

*(C2) There exists one UI-Line $u_{p_0q_0}$ in $S[t_a : t_b]$, where $a \le p_0 < q_0 \le b$.*

*(C3) There exists one LI-Line $l_{p_1q_1}$ in $S[t_a : t_b]$, where $a \le p_1 < q_1 \le b$.*

**Proof.** We will prove that (C1)⇔(C2) and (C1)⇔(C3), then we obtain (C1)⇔(C2) ⇔(C3). We first prove that C1)⇔(C2). It is evident that if (C2) holds, then (C1) holds. In the following, we prove that if (C1) holds, then (C2) holds. By contradiction, suppose that there does not exist one UI-Line in $S[t_a : t_b]$. It implies that the U-Line of the minimum slope $u_{p_2q_2}$ $(a \le p_2 < q_2 \le b)$ does not intersect all line segments $l_{t_k,2\varepsilon}$ $(a \le k \le b)$. Then, there exists at least one line segment that does not intersect $u_{p_2q_2}$. As shown in Fig. 2, the line segment that does not intersect $u_{p_2q_2}$ must lie in one of the six areas. For ease of description, let $v_2v_3$ be the U-Line with the minimum slope in $S[t_a : t_b]$ and $v_5v_6$ be the line segment that does not intersect $u_{p_2q_2}$ as shown in Fig. 3. When $v_5v_6$ lies in area (I) as shown in Fig. 3a, it is evident that the slope of U-Line $v_6v_3$ is less than that of U-Line $v_2v_3$, and this contradicts the assumption that $v_2v_3$ is the U-Line with the minimum slope in $S[t_a : t_b]$. We can also obtain the same contradiction when $v_5v_6$ lies in areas (II), (V) and (VI), as shown in Fig. 3b, 3c and 3d, respectively. When $v_5v_6$ lies in area (III), it is evident that

$$k_{v_3v_6} > k_{v_2v_3}, \quad (10)$$

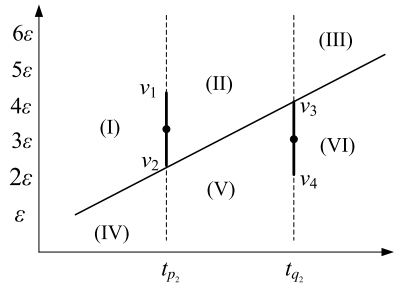



Fig. 2. The vertical line segment that does not intersect $u_{p_2q_2}$ may lie in one of the six areas.

where $k_{v_3v_6}$ is the slope of line $v_3v_6$. Since $l_0$ intersects $v_1v_2$ and $v_3v_4$, we have

$$k_{v_2v_3} \geq k_{l_0}. \tag{11}$$

Since $l_0$ intersects $v_3v_4$ and $v_5v_6$, we can obtain that

$$k_{l_0} \geq k_{v_3v_6}. \tag{12}$$

Combining Equations (10), (11) and (12), we have

$$k_{l_0} \geq k_{v_3v_6} > k_{v_2v_3} \geq k_{l_0}. \tag{13}$$

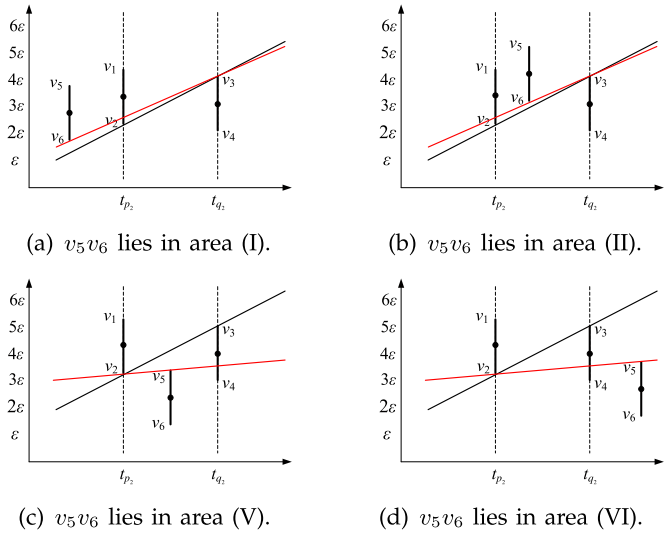
This is a contradiction. When $v_5v_6$ lies in areas (IV), the proof is quite similar. Thus, the assumption does not hold, and there exists one UI-Line $u_{p_0q_0}$ in $S[t_a : t_b]$, where $a \leq p_0 < q_0 \leq b$. Therefore, we have (C1)⇔(C2). The proof for (C1)⇔(C3) is quite similar. □

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

**Lemma 4.** *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 there exists one UI-Line $u_{p_3q_3}$ and one LI-Line $l_{p_4q_4}$ in $S[t_a : t_{b+1}]$, where $a \leq p_3 < q_3 \leq b+1$, $a \leq p_4 < q_4 \leq b+1$. It is evident that $k_{u_{p_1q_1}} \geq k_{l_{p_2q_2}}$ (according to the definitions of UI-Line and LI-Line). (I) When $k_{u_{p_1q_1}} = k_{l_{p_2q_2}}$, we obtain that $u_{p_1q_1}$ is the same as $l_{p_2q_2}$, and the UI-Line (LI-Line) of $S[t_a : t_{b+1}]$ is the same as that of $S[t_a : t_b]$. (II) When $k_{u_{p_1q_1}} > k_{l_{p_2q_2}}$, we can calculate the new UI-Line, or the new LI-Line, or both 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 $(t_b, x[t_b + 1])$), 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:*

$$\begin{aligned} &y = k_l(t - t_{b+1}) + x[t_{b+1}] - \varepsilon \\ &where\ k_l = \max_{a \leq j \leq b} \frac{(x[t_j] + \varepsilon) - (x[t_{b+1}] - \varepsilon)}{t_j - t_{b+1}}. \end{aligned} \tag{14}$$




(a) $v_5v_6$ lies in area (I). (b) $v_5v_6$ lies in area (II).

(c) $v_5v_6$ lies in area (V). (d) $v_5v_6$ lies in area (VI).

Fig. 3. When $v_5v_6$ lies in areas (I), (II), (V) and (VI), it is evident that the slope of U-Line $v_2v_5$ (the red line) is less than that of U-Line $v_2v_3$, contradicting the assumption that $v_2v_3$ is the U-Line with the minimum slope in $S[t_a : t_b]$.

*(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:*

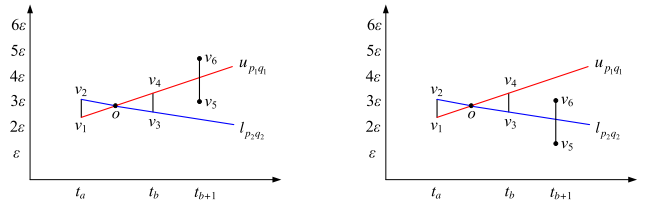
$$\begin{aligned} &y = k_u(t - t_{b+1}) + x[t_{b+1}] + \varepsilon, \\ &where\ k_u = \min_{a \leq j \leq b} \frac{(x[t_j] - \varepsilon) - (x[t_{b+1}] + \varepsilon)}{t_j - t_{b+1}}. \end{aligned} \tag{15}$$

*(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 respectively shown in Equations (14) and (15).*

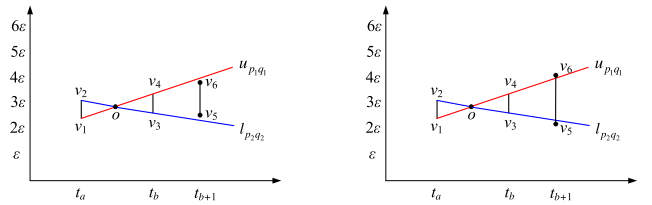
*(4) When $x[t_{b+1}] - \varepsilon \leq l_{p_2q_2}|_{t=t_{b+1}} < u_{p_1q_1}|_{t=t_{b+1}} \leq x[t_{b+1}] + \varepsilon$, the UI-Line and the LI-Line of $S[t_a : t_{b+1}]$ are the same as that of $S[t_a : t_b]$.*

**Proof.** As shown in Fig. 4a. 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 fold line $v_1ov_3$, and the points $(t_k, x[t_k] + \varepsilon)$ $(a \leq k \leq b)$ are above or on fold line $v_2ov_4$, then the lines satisfying the following two conditions will intersect all line segment $l_{t_k,2\varepsilon}$ $(a \leq k \leq b)$. (i) The line passes through point $o$. (ii) The slope of the line is less than or equal to that of $u_{p_1q_1}$ and more than or equal to that of $l_{p_2q_2}$. Since $[x[t_{b+1}] - \varepsilon, x[t_{b+1}] + \varepsilon] \cap [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] \neq \emptyset$, there exists a line that passes through point $o$, and intersects all line segment $l_{t_k,2\varepsilon}$ $(a \leq k \leq b+1)$. From Lemma 3, we obtain that there exists a UI-Line and an LI-Line for $S[t_a : t_{b+1}]$.

In the following, we first prove that when $k_{u_{p_1q_1}} = k_{l_{p_2q_2}}$, $u_{p_1q_1}$ is the same as $l_{p_2q_2}$. Since $u_{p_1q_1}$ is the UI-Line of $S[t_a : t_b]$, then the slopes of all the L-Lines (including LI-Line $l_{p_2q_2}$) in $S[t_a : t_b]$ is less than or equal to $k_{u_{p_1q_1}}$, and when the equal sign holds, we have $l_{p_2q_2}|_{t=t_{p_1}} = u_{p_1q_1}|_{t=t_{p_1}}$ and $l_{p_2q_2}|_{t=t_{q_1}} = u_{p_1q_1}|_{t=t_{q_1}}$, which means that $l_{p_2q_2}$ is the same as $u_{p_1q_1}$.



(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. 4. 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.

Since $l_{t_{b+1},2\varepsilon} \cap [l_{p_2q_2}|_{t=t_{b+1}}, u_{p_1q_1}|_{t=t_{b+1}}] \neq \emptyset$, we obtain

$$u_{p_1q_1} \cap l_{t_{b+1},2\varepsilon} \neq \emptyset. \tag{16}$$

Since $u_{p_1q_1}$ is the UI-Line of $S[t_a : t_b]$, we have

$$u_{p_1q_1} \cap l_{t_k,2\varepsilon} \neq \emptyset, a \leq k \leq b. \tag{17}$$

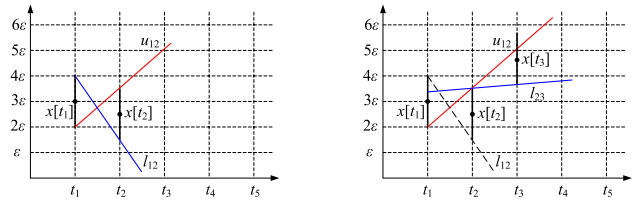
Combining Equations (16) and (17), we obtain

$$u_{p_1q_1} \cap l_{t_k,2\varepsilon} \neq \emptyset, a \leq k \leq b+1. \tag{18}$$

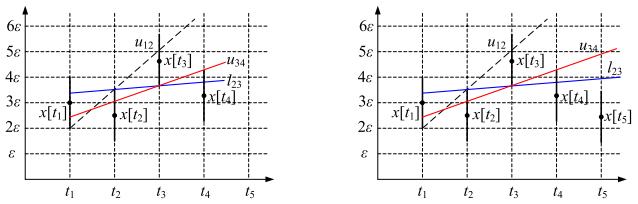
It implies that $u_{p_1q_1}$ is the UI-Line of $S[t_a : t_{b+1}]$. The proof that $l_{p_2q_2}$ is the LI-Line of $S[t_a : t_{b+1}]$ is quite similar. Therefore, when $k_{u_{p_1q_1}} = k_{l_{p_2q_2}}$, the UI-Line (LI-Line) of $S[t_a : t_{b+1}]$ is the same as that of $S[t_a : t_b]$.

When $k_{u_{p_1q_1}} > k_{l_{p_2q_2}}$, for Case (1), since $u_{p_1q_1}$ is the UI-Line of $S[t_a : t_b]$, we obtain that $u_{p_1q_1} \cap l_{t_k,2\varepsilon} \neq \emptyset (a \leq k \leq b)$. Fig. 4a 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 $u_{p_1q_1} \cap l_{t_{b+1},2\varepsilon} \neq \emptyset$, 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 $l_{t_a,2\varepsilon}$ $(l_{t_b,2\varepsilon})$, $v_2$ $(v_3)$ is the intersection point of $l_{p_2q_2}$ and $l_{t_a,2\varepsilon}$ $(l_{t_b,2\varepsilon})$, and point $o$ is the intersection point of $u_{p_1q_1}$ and $l_{p_2q_2}$ as shown in Fig. 4a. 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 fold line $v_1ov_3$, and the points $(t_k, x[t_k] + \varepsilon)$ $(a \leq k \leq b)$ are above or on fold line $v_2ov_4$. When the new data point $(t_{b+1}, x[t_{b+1}])$ arrives, we connect point $v_5(t_{b+1}, x[t_{b+1}] - \varepsilon)$ and point $o$ to obtain line $l$, which can always be rotated around point $v_5$ 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 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 points $(t_j, x[t_j] + \varepsilon)$ $(a \leq j \leq b+1)$ are above or on it. This implies that $l_{p_0(b+1)} \cap l_{t_j,2\varepsilon} \neq \emptyset$ $(a \leq j \leq b+1)$. Therefore, $l_{p_0(b+1)}$ is the LI-Line of $S[t_a : t_{b+1}]$. The proof for Cases (2) and (3) are quite similar.



(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. 5. An example for Lemmas 1, 2 and 4.

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). □

**Remark.** The newly calculated UI-Line (LI-Line) must pass through data point $(t_{b+1}, x[t_{b+1}] + \varepsilon)$ $((t_{b+1}, x[t_{b+1}] - \varepsilon))$.

In the following, we present an example to illustrate Lemmas 1, 2 and 4.

**Example .** After data points $(t_1, x[t_1])$ and $(t_2, x[t_2])$ arrive (Fig. 5a), $u_{12}$ $(l_{12})$ is the UI-Line (LI-Line) of the time series $S[t_1 : t_2]$. When $(t_3, x[t_3])$ arrives (Fig. 5b), since $(t_3, x[t_3])$ can be represented by $u_{12}$, we do not need to calculate the new UI-Line based on Lemma 2. However, $l_{12}$ needs to be updated as $l_{23}$ which is the LI-Line of $S[t_1 : t_3]$ according to Lemma 4. Based on Lemma 1, data points $\{(t_k, x[t_k]) | \ 1 \leq k \leq 3\}$ are all within $\varepsilon$ from line $u_{12}$ $(l_{23})$ in the vertical direction.

Similarly, when $(t_4, x[t_4])$ arrives (Fig. 5c), we do not need to calculate the new LI-Line based on Lemma 2, and based on Lemma 4, $u_{12}$ is updated as $u_{34}$, which is the UI-Line of $S[t_1 : t_4]$.

After the arrival of $(t_5, x[t_5])$ (Fig. 5d), $x[t_5] + \varepsilon < l_{23}|_{t=t_5}$ holds. Based on Lemma 1, we need to end the current segment and use a line segment to represent the data points seen so far but not compressed. How to select this line segment is presented in Algorithm 1. We then start a new segment at $t = t_5$.

Based on Lemmas 1 and 4, 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 4. 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 as follows, 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 lemmas.

**Lemma 5.** *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 $p \le p_j < q_j \le q$ $(j = 1, 2)$. 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.*

$$k_u = \min_{p_1 \le j \le q} \frac{(x[t_j] - \varepsilon) - (x[t_{q+1}] + \varepsilon)}{t_j - t_{q+1}}, \tag{19}$$

$$k_l = \max_{p_2 \le j \le q} \frac{(x[t_j] + \varepsilon) - (x[t_{q+1}] - \varepsilon)}{t_j - t_{q+1}}. \tag{20}$$

**Proof.** We prove it by contradiction. Based on Lemma 4, 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 $p \le p' < p_1$ 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 $u_{p'(q+1)}|_{t=t_{p'}} \le u_{p_1q_1}|_{t=t_{p'}}$, and since $u_{p_1q_1}$ is the UI-Line of $S[t_p : t_q]$, then $u_{p'(q+1)}|_{t=t_{p_1}} \ge u_{p_1q_1}|_{t=t_{p_1}}$. This implies that

$$k_{u_{p'(q+1)}} \ge k_{u_{p_1q_1}}. \tag{21}$$

In addition, since $u_{p'(q+1)}$ is the UI-Line of $S[t_p : t_{q+1}]$, $u_{p'(q+1)}$ 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

$$k_{u_{p'(q+1)}} \le k_{u_{p_1q_1}}. \tag{22}$$

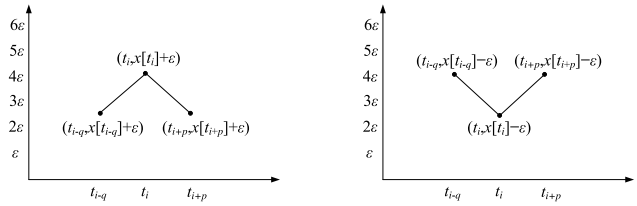
Based on Equations (21) and (22), we obtain that $k_{u_{p'(q+1)}} = k_{u_{p_1q_1}}$. Since $u_{p'(q+1)}|_{t=t_{q+1}} = x[t_{q+1}] + \varepsilon < u_{p_1q_1}|_{t=t_{q+1}}$ in Cases (2) and (3) of Lemma 4, $u_{p'(q+1)}|_{t=t_{p_1}} < u_{p_1q_1}|_{t=t_{p_1}} = x[t_{p_1}] - \varepsilon$, and then we obtain $u_{p'(q+1)} \cap l_{t_{p_1},2\varepsilon} = \emptyset$. 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 (19) holds. The proof for Equation (20) is quite similar. □

Lemma 5 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 $a \le i \le b$.*

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 $U[t_a : t_b] = \{(t_i, x[t_i] + \varepsilon) | a \le i \le b\}$ $(L[t_a : t_b] = \{(t_i, x[t_i] - \varepsilon) | a \le i \le b\})$, 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 $\exists p, q \ge 1$, 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 $a \le i - q < i < i + p \le b$. 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 $U[t_a : t_b]$ $(L[t_a : t_b])$.*




(a) Point $(t_i, x[t_i] + \varepsilon)$ is an upper prune point.

(b) Point $(t_i, x[t_i] - \varepsilon)$ is a lower prune point.

Fig. 6. An example for Definition 2.

Fig. 6 shows an example for Definition 2. In the following, we present a lemma for narrowing the scan space even further based on Definition 2 when we need to update the UI-Line or LI-Line.

**Lemma 6.** *Given a time series $S[t_a : t_b]$ of data points $(t_k, x[t_k])$ $(a \le k \le b)$, there exist one UI-Line $u_{p_1q_1}$ and one LI-Line $l_{p_2q_2}$ in $S[t_a : t_b]$, where $a \le p_j < q_j \le b$ $(j = 1, 2)$. 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, according to Definition 2 we have

$$x[t_{i+p}] + \varepsilon < l_1|_{t=t_{i+p}}, \tag{23}$$

where $l_1$ is the line passing through $(t_{i-q}, x[t_{i-q}] + \varepsilon)$ and $(t_i, x[t_i] + \varepsilon)$ $(p, q \ge 1$ and $a \le i - q < i < i + p \le b + 1)$. In addition, since $L_0$ is the LI-Line of $S[t_a : t_{b+1}]$, point $(t_{i-q}, x[t_{i-q}] + \varepsilon)$ is above or on $L_0$, i.e.,

$$l_1|_{t=t_{i-q}} \ge L_0|_{t=t_{i-q}}. \tag{24}$$

Combining $l_1|_{t=t_i} = L_0|_{t=t_i} = x[t_i] + \varepsilon$ and Equation (24), we have $k_{l_1} \le k_{L_0}$, 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 $k_{l_1} \le k_{L_0}$, we have

$$L_0|_{t=t_{i+p}} \ge l_1|_{t=t_{i+p}}. \tag{25}$$

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

$$L_0|_{t=t_{i+p}} \le x[t_{i+p}] + \varepsilon. \tag{26}$$

From Equations (25) and (26), we obtain that $x[t_{i+p}] + \varepsilon \ge l_1|_{t=t_{i+p}}$. This is a contradiction to Equation (23). 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. □

**Algorithm 1.** GDPLA

**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) $\leftarrow \langle\rangle$;
2: $start \leftarrow t_1$, $end \leftarrow t_2$, $SegNum \leftarrow 1$, $flag \leftarrow 0$;
3: InitDLinkList($U$), InitDLinkList($L$);
4: **while** the $j$-th data point arrives **do**
5: **if** $flag == 1$ **then**
6: InitDLinkList($U$), InitDLinkList($L$);
7: $flag \leftarrow 0$;
8: continue;
9: **end if**
10: **if** Equation (2) in Lemma 1 occurs **then**
11: $end \leftarrow t_{j-1}$;
12: 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 $[start, end]$. */
13: $start \leftarrow t_j$;
14: $SegNum$++, $flag \leftarrow 1$, InitDLinkList($U$), InitDLinkList($L$);
15: **else if** Case (1) of Lemma 4 occurs **then**
16: UpdateAndPruneUp($U$, $t_j$, $x[t_j]$);
17: CalculateLILine($U$, $t_j$, $x[t_j]$);
18: **else if** Case (2) of Lemma 4 occurs **then**
19: UpdateAndPruneLow($L$, $t_j$, $x[t_j]$);
20: CalculateUILine($L$, $t_j$, $x[t_j]$);
21: **else if** Case (3) of Lemma 4 occurs **then**
22: UpdateAndPruneUp($U$, $t_j$, $x[t_j]$);
23: CalculateLILine($U$, $t_j$, $x[t_j]$);
24: UpdateAndPruneLow($L$, $t_j$, $x[t_j]$);
25: CalculateUILine($L$, $t_j$, $x[t_j]$);
26: **else if** Case (4) of Lemma 4 occurs **then**
27: UpdateAndPruneUp($U$, $t_j$, $x[t_j]$);
28: UpdateAndPruneLow($L$, $t_j$, $x[t_j]$);
29: **end if**
30: $j$++;
31: **end while**
32: $G(S) \leftarrow \langle g^{(1)}, g^{(2)}, \ldots \rangle$
33: return $G(S)$;

Based on the above-mentioned lemmas, we present GDPLA 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 16 is used to delete the upper prune points (based on Lemma 6) from all the 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 17, 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. In the following, we prove that Algorithm 3 can correctly calculate the new LI-Line.

**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, prior pointer, next pointer */
1: Add $(t_j, x[t_j] + \varepsilon)$ as the last node of $U$;
2: LinkNode $*p$;
3: $p = tail \rightarrow prior$ ; /* $tail$ is the tail pointer of $U$*/
4: **while** $(p.time, p.data)$ is an upper prune point **do**
5: $p \rightarrow prior \rightarrow next = p \rightarrow next$;
6: $p \rightarrow next \rightarrow prior = p \rightarrow prior$;
7: $p = p \rightarrow prior$;
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 $*p_1$,$*p_2$;
2: $p_1 = head$, $p_2 = p_1 \rightarrow next$;
3: $l_1 \leftarrow$ The line passing through $(p_1.time, p_1.data)$ and $(t_j, x[t_j] - \varepsilon)$;
4: $l_2 \leftarrow$ The line passing through $(p_2.time, p_2.data)$ and $(t_j, x[t_j] - \varepsilon)$;
5: **while** $k_{l_1} \leq k_{l_2}$ **do**
6: $p_1 = p_2$;
7: $p_2 = p_2 \rightarrow next$;
8: $l_1 \leftarrow$ The line passing through $(p_1.time, p_1.data)$ and $(t_j, x[t_j] - \varepsilon)$;
9: $l_2 \leftarrow$ The line passing through $(p_2.time, p_2.data)$ and $(t_j, x[t_j] - \varepsilon)$;
10: **end while**
11: Calculate the new LI-Line passing through $(p_1.time, p_1.data)$ 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 slope $k_{l_1}$.*/

**Theorem 1 (Correctness of Algorithm 3).** *Algorithm 3 can correctly calculate the new LI-Line for the current segment, including the data points seen so far but not compressed yet.*

**Proof.** Suppose that after the *while* condition runs $m$ times, $k_{l_1} \leq k_{l_2}$ does not hold, then we only need to prove that $k_{l_1} \geq k_{l_3}$ where $l_3$ denotes any line passing through $(p_3.time, p_3.data)$ and $(t_j, x[t_j] - \varepsilon)$ where $p_3$ is a LinkNode of $U$ and $p_2.time < p_3.time < t_j$, so that we can obtain that line $l_1$ is the new LI-Line based on Lemma 4.

By contradiction, if it does not hold, then there exists a LinkNode $p_4$ of $U$, and $p_2.time < p_4.time < t_j$, satisfying $k_{l_1} < k_{l_4}$ where $l_4$ denotes the line passing through $(p_4.time, p_4.data)$ and $(t_j, x[t_j] - \varepsilon)$. Then we obtain that $k_{l_1} > k_{l_2}$ (the *while* condition does not hold at the $(m+1)$th loop) and $k_{l_1} < k_{l_4}$ hold, as shown in Fig. 7. It is easy to obtain that

$$k_{p_1 p_2} > k_{l_1} \tag{27}$$

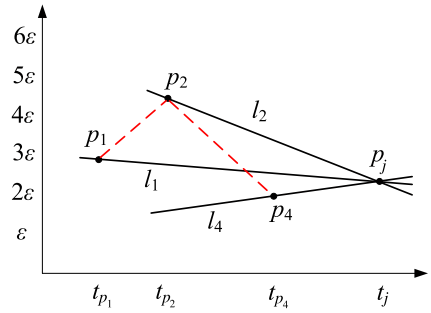



Fig. 7. Data point $p_2$ is an upper prune point.

and

$$k_{l_2} > k_{p_2 p_4}. \tag{28}$$

Combining Equations (27), (28) and $k_{l_1} > k_{l_2}$, we have

$$k_{p_1 p_2} > k_{p_2 p_4}.$$

Then we know that $(p_2.time, p_2.data)$ 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 calculate 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.

### 3.3 Analysis of GDPLA

The main space and time complexities of GDPLA lie 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 GDPLA 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 is 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 2 (Time complexity).** *The time complexity of GDPLA 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 lines 4 to 8. Suppose that the *while* condition runs $M$ times, where for $M_1$ times the loop condition is true and for $M_2$ times the 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 $M = M_1 + M_2 = M_1 + n$. For $M_1$ times the loop condition is satisfied, which implies that a total number 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 determined by comparing the slopes of two lines. For ease of presentation, we claim that the time complexity of one comparison operation is 1, which denotes the time complexity of one comparison between two float numbers, 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 lines 5 to 10. Suppose that the *while* condition runs $N$ times, where for $N_1$ times the loop condition is true and for $N_2$ times the 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 time complexity of CalculateLILine($U$, $t_j$, $x[t_j]$) is at most $2n - M_1$. Similarly, the 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 GDPLA for $S[t_1 : t_n]$ is $O(M_1 + n + M_1' + n + 2n - M_1 + 2n - M_1') = O(6n)$, i.e., $O(n)$. □

GDPLA 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 GDPLA a proper choice for resource-constrained WSNs. In what follows, we present the proof of optimality in details.

**Theorem 3 (Optimality).** *Let $S^{(n)} = S[t_1 : t_n]$ be an arbitrary time series that must be approximated with a Disconnected PieceWise Linear approximation such that for all $k = 1, 2, \ldots, n$, $|\widetilde{x}[t_k] - x[t_k]| \leq \varepsilon$. If GDPLA produces a DPWL representation with $G(S) = (g^{(1)}, g^{(2)}, \ldots, g^{(H)})$, then no other valid DPWL representation with $H' < H$ line segments exists.*

**Proof.** By contradiction, let $G'(S)$ be a valid representation with $G'(S) = (l^{(1)}, l^{(2)}, \ldots, l^{(H')})$ 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 $m = 1, 2, \ldots, H' - 1$ 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 $\varepsilon$. 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 $\varepsilon$ on the $L_\infty$ 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 $\varepsilon$. Similarly, from the property (P3) of Lemma 1, we obtain $S[t_{j_p} : t_{j_{p+1}+1}]$ cannot be approximated

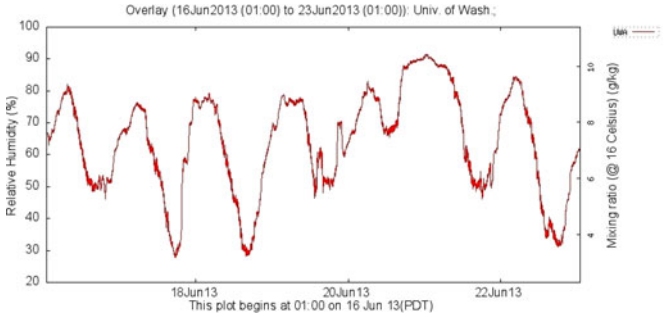



Fig. 8. The humidity measurements sampled in a minute basis from June 16 to June 23 in 2013.

by any straight line within $\varepsilon$ on the $L_\infty$ 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 $m = 1, 2, \ldots, H' - 1$. 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 $S[t_{j_{H'-1}} : t_{j_{H'}+1}]$ $(\subseteq S[t_{y_{H'-1}} : t_{y_{H'}}])$ can be approximated by $l^{(H')}$ within $\varepsilon$. However, from the property (P3) of Lemma 1, we obtain that the subseries $S[t_{j_{H'-1}} : t_{j_{H'}+1}]$ cannot be approximated by any straight line within $\varepsilon$ on the $L_\infty$ 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. □

## 4 Experiments

In this section, we further evaluate our approach through extensive experiments on real-world data sets.

### 4.1 Experimental Setting

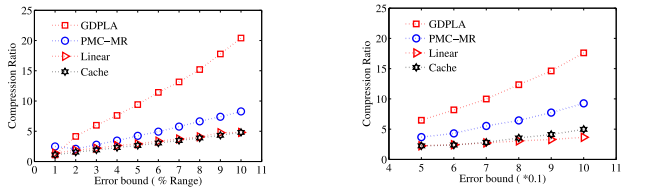
We use two real-world data sets: the Intel-Berkeley Temperature data set (IntelTEM data set) and the University of Washington Relative Humidity data set (WashHUM data set).

i) *IntelTEM data set*. It contains temperature measurements by 54 sensor nodes deployed on a single floor of the Intel-Berkeley Research lab over the period of one month [18]. The change rate of the temperature data is low.

ii) *WashHUM data set*. It includes the relative humidity weather measurements for the station in the University of Washington [19]. The humidity measurements are sampled in a minute basis from June 16 to June 23 in 2013. As shown in Fig. 8, the change rate of humidity is high.

For the above two data sets, 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 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 $x_i$.
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.




(a) Compression ratio vs. $\varepsilon$ (IntelTEM). (b) Compression ratio vs. $\varepsilon$ (WashHUM).

Fig. 9. Compression ratio as a function of the error bound $\varepsilon$.

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.

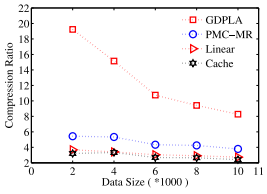
### 4.2 Performance Evaluation

We mainly compare the performance of GDPLA against PMC-MR [15], Cache [16] and Linear filter [12] (they are described in the Introduction section).

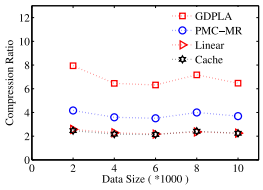
#### 4.2.1 *Results on Compression Ratio*

The first group of experiments (Fig. 9) measures the compression ratio as a function of the error bound $\varepsilon$. Fig. 9a depicts that when the original data size of the IntelTEM data set is 8,000, how the compression ratios of GDPLA, PMC-MR, Linear and Cache change with the increase of error bound $\varepsilon$. As expected, compression ratios increase with the increase of $\varepsilon$ for all the approaches, and when $\varepsilon = 0.1$, the compression ratio of our 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 an 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 $10$ percent of the original sample size, and transmit around $5$ percent of the original sample size when $\varepsilon = 0.1$. Fig. 9b shows that when the original data size of WashHUM is 10,000, how the compression ratios of GDPLA, PMC-MR, Linear and Cache change with the increase of error bound $\varepsilon$. The observed experimental results are similar with the results shown in Fig. 9a. For $\varepsilon = 0.5$, the compression ratio of GDPLA is 1.75, 2.87 and 2.88 times larger than that of PMC-MR, Linear and Cache respectively. We observe that the increase rate of the compression ratio of GDPLA is larger than that of PMC-MR, Linear and Cache. It is for $\varepsilon = 1$ that the compression ratio of GDPLA is 1.90, 4.81 and 3.54 times larger than that of PMC-MR, Linear and Cache respectively.

Fig. 10a plots when $\varepsilon = 0.05$, how the compression ratio achieved by GDPLA, PMC-MR, Linear and Cache change when the size of the IntelTEM data set varies from 2,000 to 10,000. Compression ratio drops as the increase of the size of the data set for all the approaches. This may be related to the distribution of data. Fig. 10b plots when $\varepsilon = 0.5$, how the compression ratio achieved by GDPLA, PMC-MR, Linear and Cache change when the size of the WashHUM data set varies from 2,000 to 10,000. We observe that the compression ratios of all the approaches change slowly. From Figs. 9 and 10, we can easily know that the compression

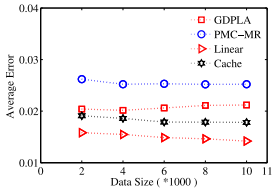



(a) Compression ratio vs. Data size (IntelTEM).

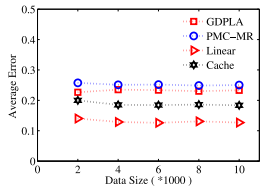



(b) Compression ratio vs. Data size (WashHUM).

Fig. 10. Compression ratio as a function of data size.




(a) Average error vs. data size (IntelTEM).




(b) Average error vs. data size (WashHUM).

Fig. 12. Average error as a function of data size.

ratio of GDPLA is larger than that of PMC-MR, Linear and Cache, which closely conforms our theoretical analysis.

#### 4.2.2 Results on Average Error

In Fig. 11a, we show the average errors of GDPLA, PMC-MR, Linear and Cache with the increase of error bound $\varepsilon$ when the original data size of IntelTEM is 8,000. We can observe that the average errors are roughly proportional to $\varepsilon$ for all the approaches, and the average errors of all the approaches are roughly less than half of the maximal absolute error. The similar experimental results on WashHUM are reported in Fig. 11b.

Fig. 12a compares the average errors of GDPLA, PMC-MR, Linear and Cache on the IntelTEM data set, where the size of the data set varies from 2,000 to 10,000 and the error is fixed at $\varepsilon = 0.05$. 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 the IntelTEM data set varies from 2,000 to 10,000. The average error of GDPLA is slightly larger than that of Cache and less than that of PMC-MR. Fig. 12b compares the average errors of GDPLA, PMC-MR, Linear and Cache on the WashHUM data set, where the size of the data set varies from 2,000 to 10,000 and the error is fixed at $\varepsilon = 0.5$. We can see that the average errors of the four approaches are less than 0.3, and the average errors do not change much when the size of the WashHUM data set varies from 2,000 to 10,000. The average error of GDPLA is also slightly larger than that of Cache and less than that of PMC-MR.

#### 4.2.3 Results on Average Processing Time

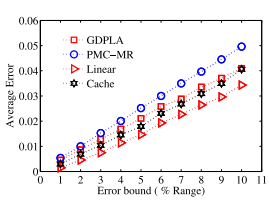
Fig. 13a compares the processing time per data point of GDPLA, PMC-MR, Linear and Cache on the IntelTEM data set when $\varepsilon$ varies from 0.01 to 0.1, where the data size is 8,000. The processing time per data point of GDPLA is slightly greater than that of other approaches. Fig. 13b compares the processing time per data point of GDPLA, PMC-MR, Linear and Cache on the WashHUM data set when $\varepsilon$ varies from 0.5 to 1.0, where the data size is 10,000. The processing time per data point of GDPLA is larger than that of other approaches. For GDPLA, energy overhead spent in calculation is worthwhile because the energy overhead of WSNs is mainly dominated by communication and the compression ratio of GDPLA is far more than that of other approaches and therefore GDPLA transmits much less data than other approaches, which results in less energy overhead by GDPLA.

Table 2 shows the number of the slope comparisons required by GDPLA for different error bounds. For the IntelTEM data set, $\varepsilon$ varies from 0.01 to 0.05 and the data size is 8,000. We can easily observe that the number of slope comparisons is less than 48,000 (6*8,000) for varying $\varepsilon$ from 0.01 to 0.05. For the WashHUM data set, $\varepsilon$ varies from 0.5 to 1.0 and the data size is 10,000. It is evident that the number of slope comparisons is less than 60,000 (6*10,000) for varying $\varepsilon$ from 0.5 to 1.0.

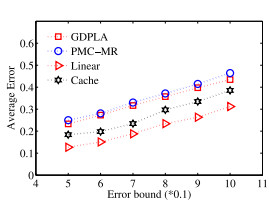
Table 3 demonstrates the number of slope comparisons required by GDPLA for varying data size from 2,000 to 10,000. For the IntelTEM data set, $\varepsilon$ is set to 0.05 and the results demonstrate that the number of slope comparisons is less than the data size multiplied by 6. The similar experimental results are shown for the WashHUM data set. As expected, the experimental results validate our theoretical analysis.

## 5 Related Work

The approximation representation of data streams resulting from many applications including WSN applications has been an active research area since the last few years. A large body of research works on time series segmentation has been directed towards finding techniques for data reduction in order to cope with the large sizes of the raw data. There has been increasing interest in the piecewise linear approximation techniques [12], [22]. The idea behind PLA is that a sequence of line segments can be

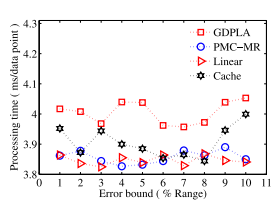



(a) Average error vs. $\varepsilon$ (IntelTEM).

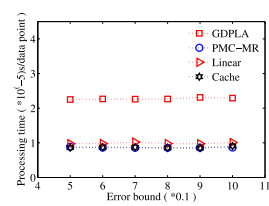



(b) Average error vs. $\varepsilon$ (WashHUM).

Fig. 11. Average error as a function of the error bound $\varepsilon$.




(a) Average processing time vs. $\varepsilon$ (IntelTEM).




(b) Average processing time vs. $\varepsilon$ (WashHUM).

Fig. 13. Average processing time as a function of the error bound $\varepsilon$.

TABLE 2
Number of Slope Comparisons as a Function of $\varepsilon$

| Data size = 8000 (IntelTEM) | | Data size = 10000 (WashHUM) | |
|---|---|---|---|
| $\varepsilon$ | Number of slope comparisons | $\varepsilon$ | Number of slope comparisons |
| 0.01 | 21,242 | 0.5 | 39,010 |
| 0.02 | 27,537 | 0.6 | 40,422 |
| 0.03 | 30,124 | 0.7 | 41,555 |
| 0.04 | 31,269 | 0.8 | 42,237 |
| 0.05 | 32,334 | 0.9 | 43,027 |

TABLE 3
Number of Slope Comparisons as a Function of Data Size

| Error bound $\varepsilon = 0.05$ (IntelTEM) | | Error bound $\varepsilon = 0.5$ (WashHUM) | |
|---|---|---|---|
| Size of dataset | Number of slope comparisons | Size of dataset | Number of slope comparisons |
| 2,000 | 8,571 | 2,000 | 8,179 |
| 4,000 | 16,870 | 4,000 | 1,5720 |
| 6,000 | 24,655 | 6,000 | 23,393 |
| 8,000 | 32,334 | 8,000 | 31,753 |
| 10,000 | 39,754 | 10,000 | 39,010 |

used to represent a time series with 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. [23] 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 the 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 stop 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. The Sliding Window (SW) algorithms [12] work by anchoring the left point of a potential segment at the first data point of a time series, then attempting to approximate the data to the right with increasing longer segments. At some point i, the error for the potential segment is greater than the user-specified threshold, so the subsequence from the anchor to i-1 is transformed into a segment. The anchor is moved to location i, and the process repeats until the entire time series has been transformed into a piecewise linear approximation. The SWAB algorithm [12] is based on the ideas of Sliding Window algorithm and Bottom-Up algorithm, and SWAB is shown to have a performance close to an optimal algorithm. However, for disconnected piecewise linear approximation using the $L$-norm error metric, our algorithm GDPLA is an optimal algorithm. Therefore, the performance of GDPLA is not worse than that of SWAB.

Online algorithms handle time series based on the data seen so far not the whole time series. A classic Sliding Window method [20] is such a method. It works by initializing the first data point of a time series as the left endpoint of a segment and then trying to put one more data point into the segment in each step. The interpolating line or regression line between the two endpoints of the segment is used as the approximation, and the corresponding representation error is calculated. If the representation error of the current segment exceeds the predefined maximal error after adding point $i$ into it, point $i-1$ will be set as the right endpoint of the current segment. Then, point $i-1$ or point $i$ is taken as the left endpoint of the next segment in the interpolation approximation or regression approximation, respectively. By repeating the above process, the whole time series can be piecewisely approximated by straight lines. Some methods have been proposed based on the classic SW method, such as Feasible Space Window (FSW) [13], which introduces the concept of Feasible Space (FS). FS is an area in the time series data value space so that any straight line in this area can approximate each data point seen so far but not compressed within a given error bound. FSW aims to find the farthest segmenting point to make each segment as long as possible given an error bound on each data point. FSW always finds the farthest endpoint of a segment along the incoming data stream. In contrast with other methods, FSW dramatically reduces the number of segments given the same maximum error tolerance.

Elmeleegy et al. [14] propose an piecewise linear approximation algorithm:slide fiters, which is also an optimal algorithm, but its time complexity is quadratic. Lazaridis and Mehrotra [15] propose an algorithm PMC-MR, which is an optimal online algorithm for constructing a Piecewise Constant Approximation (PCA) of a time series which guarantees that the compressed representation satisfies an error bound on the $L_\infty$ distance. Buragohain et al. [21] propose an online streaming algorithm MIN-INCREMENT, which achieves a $(1+\varepsilon)-$approximation of a $B$-bucket histogram using $O(\varepsilon^{-1}BlogU)$ space, where $U$ is the size of the domain for data values. Liu et al. [22] 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 $O(n^2logn)$ time and takes $O(n)$ memory space. Olston et al. [16] 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], [17]. 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 $\varepsilon$ 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 disconnected 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 $\varepsilon$. To remedy

this deficiency, we present an optimal online algorithm GDPLA, which uses the piecewise linear approximation techniques to approximate a time series with the error constraint, and GDPLA only has a running time of $O(n)$.

## 6 Conclusion

In this paper, we tackle the problem of online compression of data streams in resource-constrained WSNs and present a new data compression algorithm GDPLA, which uses a set of disconnected 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 or equal to an error bound $\varepsilon$. 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 on two real data sets demonstrate that the compression performance of GDPLA is far superior to other approaches. However, the processing time of GDPLA is slightly higher than that of other algorithms. Equipped with the profound insights gained in this study, we find that an interesting open question is whether one can use connected line segments to approximate a time series with the required precision and guarantee that it is optimal in the sense that it generates the minimum number of connected line segments in linear time.

## Acknowledgements

This work was 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, and the National Natural Science Foundation of China (NSFC) under Grant No. 61190115, 60933001, 61173022, 61100030. A preliminary version of this paper appeared in the Proceedings of IEEE INFOCOM 2013. Guohua Li finished this work during his PhD study in the Harbin Institute of Technology.

## References

[1] L. Mo, Y. He, Y. Liu, J. Zhao, S.-J. Tang, X.-Y. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in *Proc. 7th ACM Conf. Embedded Netw. Sens. Syst.*, 2009, pp. 99–112.

[2] J. Li, S. Cheng, H. Gao, and Z. Cai, "Approximate physical world reconstruction algorithms in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, 17 Mar. 2014. IEEE Computer Society Digital Library. IEEE Computer Society.

[3] Y. Li, C. Ai, Z. Cai, and R. Beyah, "Sensor scheduling for p-percent coverage in wireless sensor networks," *Cluster Comput.*, vol. 14, no. 1, pp. 27–40, 2011.

[4] C. Ai, L. Guo, Z. Cai, and Y. Li, "Processing area queries in wireless sensor networks," in *Proc. 5th Int. Conf. Mobile Ad-Hoc Sens. Netw.*, 2009, pp. 1–8.

[5] S. Cheng, J. Li, and Z. Cai, "$O(\epsilon)$-Approximation to physical world by sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 3080–3092.

[6] K. P. Chan and A. W. Fu, "Efficient time series matching by wavelets," in *Proc. 15th Int. Conf. Data Eng.*, 1999, pp.126–133.

[7] M. N. Garofalakis and A. Kumar, "Wavelet synopses for general error metrics," *ACM Trans. Database Syst.*, vol. 30, no. 4, pp. 888–928, 2005.

[8] L. Keselbrener and S. Akselrod, "Selective discrete Fourier transform algorithm for time-frequency analysis: Method and application on simulated and cardiovascular signals," *IEEE Trans. Biomed. Eng.*, vol. 43, no. 8, pp. 789–802, Aug. 1996.

[9] 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.

[10] Y. Wu, D. Agrawal, and A. Abbadi, "A comparison of DFT and DWT based similarity search in time-series databases," in *Proc. 9th Int. Conf. Inf. Knowl. Manage.*, 2000, pp. 488–495.

[11] V. Britanak and K. R. Rao, "A new fast algorithm for the unified forward and inverse MDCT/MDST computation," *Signal Process.*, vol. 82, no. 3, pp. 433–459, 2002.

[12] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 289–296.

[13] X. Liu, Z. Lin, and H. Wang, "Novel online methods for time series segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1616–1626, Dec. 2008.

[14] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, and W. Zwaenepoel, "Online piece-wise linear approximation of numerical Streams with precision guarantees," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 145–156, 2009.

[15] I. Lazaridis and S.Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 429–440.

[16] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2003, pp. 563–574.

[17] A. Jain, E. Y. Chang, and Y. Wang, "Adaptive stream resource management using Kalman filters," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp.11–22.

[18] (2004) [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[19] (2013) [Online]. Available: http://www-k12.atmos.washington.edu/k12/grayskies/

[20] U. Appel and A.V. Brandt, "Adaptive sequential segmentation of piecewise stationary time series," *Inf. Sci.*, vol. 29, no. 1, pp. 27–56, 1983.

[21] C. Buragohain, N. Shrivastava, and S. Suri, "Space efficient streaming algorithms for the maximum error histogram," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, 2007, pp.1026–1035.

[22] C. Liu, K. Wu, and J. Pei, "An energy efficient data collection framework for wireless sensor networks by exploiting spatiotemporal ," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, pp. 1010–1023, Jul. 2007.

[23] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal histograms with quality guarantees," in *Proc. 24rd Int. Conf. Very Large Data Bases*, 1998, pp. 24–27.

**Jianzhong Li** is a professor in the School of Computer Science and Technology at the Harbin Institute of Technology, China. In the past, he worked as a visiting scholar at the University of California at Berkeley, as a staff scientist in the Information Research Group at the Lawrence Berkeley National Laboratory, and as a visiting professor at the University of Minnesota. His research interests include data management systems, sensor networks, and data intensive computing. He has published more than 150 papers in refereed journals and conferences, and has been involved in the program committees of major computer science and technology conferences, including SIGMOD, VLDB, ICDE, INFOCOM, ICDCS, and WWW. He has also served on the editorial boards for distinguished journals, including the *IEEE Transactions on Knowledge and Data Engineering*. He is a member of the IEEE.

**Guohua Li** received the BS degree in mathematics and the MS degree in computer science from the Harbin Institute of Technology, China, in 2007 and 2009, respectively. Currently, he is working toward the PhD degree in the School of Computer Science and Technology at the Harbin Institute of Technology. His research interests include congestion control and data compression in sensor networks.

**Hong Gao** is a professor in the School of Computer Science and Technology at the Harbin Institute of Technology, China. She has published more than 40 papers in the refereed journals and conferences, including the *IEEE Transactions*, SIGMOD, *Journal of Combinatorial Optimization*, and has been involved in the program committees of major computer science and technology conferences. Her research interests include data mining, data management systems, and sensor networks.