Error-bounded Approximation of Data Stream: Methods and Theories *

Qing Xie, Chaoyi Pang, Xiaofang Zhou, Xiangliang Zhang, and Ke Deng

Abstract As the development of sensor network and Internet of Things, the volume of data is rapidly increasing and the streaming data has attracted much attention recently. To efficiently process and explore data streams, the compact data representation is playing an important role, since the data approximations other than the original data items are usually applied in many stream mining tasks, such as clustering, classification and correlation analysis. In this chapter, we focus on the maximum error-bounded approximation of data stream, which represents the streaming data with constrained approximation error on each data point. There are two criteria for the approximation solution: self-adaption over time for varied error bound and real-time processing. We reviewed the existing data approximation techniques and summarized some essential theories such as optimization guarantee. Two op-

Qing Xie(✉)
Wuhan University of Technology, China
e-mail: felixxq@whut.edu.cn

Chaoyi Pang
Ningbo Institute of Technology, Zhejiang University, China
e-mail: chaoyi.pang@gmail.com

Xiaofang Zhou
University of Queensland, Australia
Soochow University, China
e-mail: zxf@itee.uq.edu.au

Xiangliang Zhang(✉)
King Abdullah University of Science and Technology, Saudi Arabia
e-mail: xiangliang.zhang@kaust.edu.sa

Ke Deng
RMIT University, Australia
e-mail: ke.deng@rmit.edu.au

* Reprinted by permission from Springer Nature: Springer, The VLDB Journal, Maximum error-bounded piecewise linear representation for online stream approximation, Q. Xie et al., ©Springer-Verlag Berlin Heidelberg 2014. (doi: https://doi.org/10.1007/s00778-014-0355-0)

timal linear-time algorithms are introduced to construct error-bounded piecewise linear representation for data stream. One generates the line segments by data convex analysis, and the other one is based on the transformed space, which can be extended to a general model. We theoretically analyzed and compared these two different spaces, and proved the theoretical equivalence between them, as well as the two algorithms.

1 Introduction

As the development of information system and sensor networks, especially the increasing attention on Internet of Things (IOT) [1, 17], streaming data is attracting much attention and has now been involved in various applications, including the medical data recording the status of patients [23, 28], financial data denoting the changing trends of stock prices [29], the large amount of data during network communications [18] and those scientific data such as sun spot numbers and ocean surface temperatures. The booming web techniques and the widely popularized mobile devices bring in the impressive popularity of web data access and sharing [30, 31], which further increase the production and proliferation of streaming data. The rapid growth of streaming data, together with its high significance in the area of health, finance, entertainment and communication in daily life, have desired novel techniques and advanced systems to manage and process the streaming data efficiently.

Characterized as continuous, random, varying and rapid in arrival, streaming data in the data stream model differ from conventionally stored data in several ways [2]:

- The data elements in the stream arrive online.
- The system has no control over the order in which data elements arrive to be processed, either within a data stream or across data streams.
- Data streams are potentially unbounded in size.
- Once an element from a data stream has been processed, it is discarded or archived, so that it cannot be retrieved easily unless it is explicitly stored in memory, which typically is small relative to the size of the data streams.

Since the streaming data have high input rate, and are potentially unbounded in size, the query and operation on streaming data usually require approximate results. Therefore it is necessary to find a proper and compact form to represent the data stream, so that the query and operation on streaming data can be more efficiently solved from the compact representations. In practical applications, the dynamics of data such as trends, patterns and outliers are most attractive. In this regard, many techniques such as histogram, line segment and wavelet are employed for effective data approximation and efficient stream query processing [9, 11, 12, 32].

The intensive work on stream approximation research is the size-bounded representation [5, 7]. Its objective is to construct a prescribed number of representations that minimize the approximation error under a specified metric, where L_2 norm (i.e., Euclidean Distance) is mostly used. However, there are two main drawbacks on the

size-bounded constraint and the use of L_2 norm for approximation: firstly, the size-bounded constraint lacks the ability to generate error-guaranteed representations for streaming data since the stream is naturally unbounded in size; secondly, the use of L_2 norm leads to the inability of controlling the approximation error on individual stream data items. To alleviate these drawbacks, researchers have made efforts in constructing the representations with guaranteed maximum allowable approximate error on each data point (L_∞ norm), which is termed as the error-bounded representations. It has been engaged in many real world applications, such as continuous queries over data streams [15, 19], sensor network management [25, 35], and monitoring physiological data for surgery operations [23, 36].

The use of line segments to represent a time series data stream, termed as *Piecewise Linear Representation* (PLR) [4, 6, 16, 32], has been extensively studied for decades under different criteria [22]. The idea of PLR is to represent a complicated wave-like data stream with a number of simple line segments, so that the streaming data can be efficiently archived, and a query on the stream can be approximately answered by a query on the line segments. Compared with the stream itself, the line segments constructed from PLR provide striking visual outlines of stream trends and can be more efficiently processed and represented in the database (Fig. 1). These advantages make PLR the most popular representation technique for the data stream [14], and it has been widely applied to support date indexing [5, 13, 26], similarity search [29, 37], and correlation analysis [33, 35].

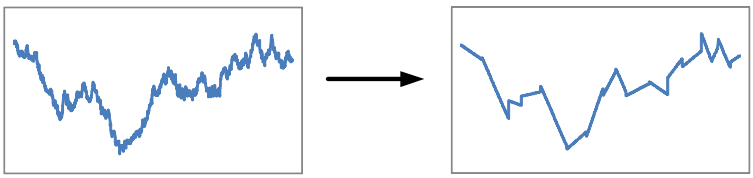


Fig. 1 An example of PLR on a time series data stream.

Histogram is also employed as contemporary error-bounded representation, i.e., Piecewise Constant Representation (PCR). For error-bounded PCR, Lazaridis et al. [15] provided an optimal algorithm to approximate sensor data stream. Gandhi et al. [8] proposed GAMPS that compressed multi-stream with guaranteed L_∞ error as well as a notable worst-case quality of approximation. They also extended the framework to address amnesic approximation and out-of-order approximation [7] using bucket-merging. As the PCR uses constant values (i.e., horizontal line segments) in representing a wave-like stream, it fails to reflect the trends of streaming data. Regarded as a generalized version of error-bounded PCR, the error-bounded PLR usually achieves a greater compression ratio and has more superiority for advanced applications than PCR.

In comparison to wavelet-based methods [10], PLR is more visually comprehensive and convenient for stream queries. For example, the line segments generated by error-bounded PLR on a stream can denote the stream trends and be used *directly* for

answering trend queries. Such queries are crucial in monitoring patients in intensive care as medical specialists [21] believe that more accurate and earlier notifications of adverse events can be predicated from the accumulated trends and variations of the physiological streams. Even through error-bounded wavelet representations can be constructed very efficiently and effectively [23, 24], the wavelet synopses may not be ready to answer stream trends directly.

In this chapter, we introduce the state-of-art optimal algorithms to generate the error-bounded PLR for data stream, which aim to construct the smallest number of line segments with approximation error constrained on each data point. We will investigate the algorithm designed in time domain named as OptimalPLR [32], and also another algorithm proposed in an interesting parameter space, which is named as ParaOptimal [20]. Such innovative transformed space has been applied in the recent research for advanced applications [27, 34]. Theoretically, the optimal results can be achieved by greedy mechanism. In order to adjust a line segment to approximate as many stream points as possible, the general idea is to determine the range of all feasible line segments, which is incrementally updated during the processing of consecutive sequence points. Whenever the current point cannot be approximated within error bound, a line segment can be determined.

Both of ParaOptimal and OptimalPLR are optimal solutions with linear time complexity for error-bounded PLR problem, but derived from different spaces, which provide essential and new sight on this classic problem. We further theoretically compare these two algorithms and the spaces they are based on, so as to provide deeper and more theoretical understanding for this problem. By setting up a mapping between the point and line in these two spaces, we theoretically proved the equivalence of these two spaces, and further linked the two algorithms together, which explained the optimal solution from different views.

2 Preliminary

In this section, along with some notations and new concepts for the study of error-bounded PLR, we formally provide the problem definition and the objective addressed in this chapter. We also present Theorem 1 which guarantees that the smallest number of line segments for the error-bounded PLR can be achieved by maximizing each representative line segment. The general notations used throughout this chapter are summarized in Table 1.

Let $S = \langle s_1, s_2, \ldots, s_k, \ldots \rangle$ denote a data stream where each point $s_i = (x_i, y_i)$ designates the actual value y_i at time stamp x_i. We use $S[i, j] = \langle s_i, s_{i+1}, \ldots, s_j \rangle$ to denote the stream *fragment* on time slot $[x_i, x_j]$ $(i < j)$ and $|S[i, j]| = j - i + 1$ to denote the cardinality of $S[i, j]$.

As an approximation technique, PLR approximates S with line segments. A *line segment* on time slot $[x_i, x_j]$ is representable by the linear function $y = a \cdot x + b$ for $x \in [x_i, x_j]$ with two parameters: slope a and offset b. Since a line can be determined by its slope and a line point or two different line points alternatively, for the conve-

Table 1 Notations

Symbol	Description
δ	Error bound for approximation (> 0)
$S = \langle s_1, s_2, ..., s_k, ... \rangle$	A data stream
$s_i = (x_i, y_i)$	Data point s_i at time stamp x_i with value y_i
$S[i, j] = \langle s_i, s_{i+1}, ..., s_j \rangle$	A (stream) fragment from time x_i to x_j
$\underline{s_i} = (x_i, y_i - \delta)$	Data point with deleted tolerant error
$\overline{s_i} = (x_i, y_i + \delta)$	Data point with added tolerant error
$seg[i, j]$	A δ-representative line on time slot $[x_i, x_j]$
$slp[i, j]$	The slope of $seg[i, j]$
$\underline{slp}[i, j]$ or $\overline{slp}[i, j]$	The minimum or maximum slopes of all $slp[i, j]$
$line(s_i, s_j)$	Line (segment) that passes point s_i and s_j
$line(\rho, s)$	Line (segment) with slope ρ that passes point s
$slope(s_i, s_j)$	The slope of $line(s_i, s_j)$
$\underline{cvx}_k$, $\overline{cvx}_k$	Reduced convex hulls for $S[1, k]$. $\underline{cvx}$ / $\overline{cvx}$ bulge downward/upward

nience of presentation, we also use $line(\rho, s)$ to denote the line that passes point s with slope ρ, and $line(s_i, s_j)$ to denote the line that passes the two points s_i and s_j. With this notation, we use $slope(s_i, s_j)$ as the slope of $line(s_i, s_j)$.

We define a stream fragment $S[i, j]$ ($i < j$) as *δ-representable* (at time slot $[t_i, t_j]$) if there exists a line segment identified by $y'_h = a \cdot x_h + b$, such that $|y'_h - y_h| \leq \delta$ holds for each point $s_h = (x_h, y_h)$ of $S[i, j]$ ($i \leq h \leq j$). In this situation, such line segment is defined as a *δ-representative* for fragment $S[i, j]$. (Notice that there can be more than one δ-representative.) For simplicity, we use $seg[i, j]$ to represent the δ-representative, and $slp[i, j]$ to denote the slope of $seg[i, j]$. Here, we use the maximum error metric L_∞ to guarantee the approximation quality at each stream data point. Furthermore, if fragment $S[i, j]$ is δ-representable but $S[i, j + 1]$ is not, then fragment $S[i, j]$ is *maximally* δ-representable on time slot $[t_i, t_j]$[2] and accordingly its $seg[i, j]$ is called maximal δ-representative.

With these designations, the *Error-bounded Piecewise Linear Representation* (Error-bounded PLR) problem discussed in this chapter is precisely defined as follows:

Definition 1. [Error-bounded PLR] Given a predefined error bound $\delta > 0$ and a data stream fragment $S[1, n] = \langle s_1, s_2, \ldots, s_n \rangle$, the (optimal) error-bounded PLR is to construct a (minimal) number of δ-representative line segments $\{seg[i_1, i_2 - 1], seg[i_2, i_3 - 1], \ldots, seg[i_k, i_{k+1} - 1]\}$ to represent $S[1, n]$, where $i_1 = 1$ and $i_{k+1} - 1 = n$.

By the definition, we specify that this problem focuses on generating disconnected line segments, i.e., the consecutive segments do not share same end points.

The following theorem has been explored in literature [4, 15] which indicates that the optimal error-bounded PLR can be solved through computing maximal δ-representative line segments.

[2] It should be noted that $S[i - 1, j]$ can be δ-representable even if $S[i, j]$ is maximally δ-representable.

Theorem 1. *Given an error bound $\delta > 0$ and stream fragment $S[1,n]$, assume that $seg[i_j, i_{j+1}-1]$ is a maximal δ-representative of $S[i_j, i_{j+1}-1]$ on time slot $[i_j, i_{j+1}-1]$ for $1 \leq j \leq k$ with $i_1 = 1$ and $i_{k+1}-1 = n$. Then the error-bounded PLR on fragment $S[1,n]$ has at least k line segments.*

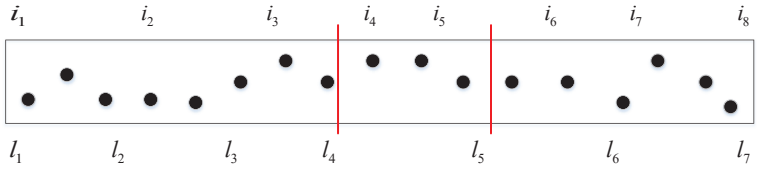


Fig. 2 The proof of Theorem 1: $\alpha = 4$.

Proof. Clearly, the claim is true for $k = 1$. For $k > 1$, assume that an optimal error-bounded PLR solution for fragment $S[1,n]$ is the set of segments $seg'[l_h, l_{h+1}-1]$ for $1 \leq h \leq m$, where $l_1 = 1$, $l_{m+1}-1 = n$ and $m < k$ (Fig. 2).

As claimed in the theorem, since each segment is a maximal δ-representative, both $i_1 = l_1 = 1$ and $l_2 \leq i_2$ hold. Let α be the index value such that $l_\alpha \leq i_\alpha$ and $l_{\alpha+1} > i_{\alpha+1}$ holds. Alternatively, since both $m < k$ and $l_{m+1}-1 = n$ hold, $i_{m+1} < l_{m+1}$ holds. Hence, we have that α exists and $1 \leq \alpha \leq m$ is confirmed.

Thus, we have $l_\alpha \leq i_\alpha < i_{\alpha+1}-1 < l_{\alpha+1}-1$. It means that $S[i_\alpha, l_{\alpha+1}-1]$ is δ-representable and is contradictory to the hypothesis that $seg[i_\alpha, i_{\alpha+1}-1]$ is a maximal δ-representative line segment. Therefore, $k \leq m$ holds, and the result is proven. □

With Theorem 1, it is inspired that the optimal error-bounded PLR results can be achieved by maximizing each δ-representative line segment, which is naturally the greedy mechanism. The algorithms introduced in this chapter all follow such mechanism.

3 OptimalPLR: an optimal algorithm to generate error-bounded PLR

In this section, we will introduce an optimal algorithm to generate the error-bounded PLR, named OptimalPLR [32]. It can theoretically construct the minimal number of line segments, and achieve the linear time efficiency. The algorithm is based on the essential analysis on the convex outline of the data stream.

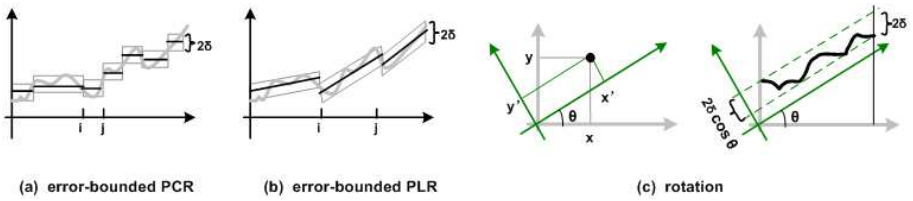


Fig. 3 Rotation examples.

3.1 Extreme slopes of maximal δ-representative

Generally, there can exist many δ-representative segments with various slopes for a δ-representable stream fragment. The range of the candidates' slopes depends on the tolerant error δ and the stream itself. We first discuss the complete slope range of the δ-representative segments for a δ-representable stream fragment, and then study the slope reductions and alterations when a new point is added to the stream fragment. Such analysis can provide the intuition of how to maximize the δ-representatives. Our discussion is based on the rotation of Cartesian coordinate system.

3.1.1 Slope rotation and extreme slopes

We start from the methods proposed by Buragohain et al. [4] and Lazaridis et al. [15] for error-bounded PCR. As a simplified version of error-bounded PLR, error-bounded PCR uses constant values rather than general line segments for the representation. Constant values can be regarded as a line with zero slope that is denoted by a horizontal line in a Cartesian coordinate plane as in Fig. 3(a). Under a predefined error bound $\delta > 0$, the optimal error-bounded PCR aims at constructing the smallest number of representation $B = \{c_{i_1}, c_{i_2}, ..., c_{i_h}\}$ for a given stream fragment $S[1,n] = \langle s_1, s_2, \ldots, s_n \rangle$ such that

$$\begin{cases} 0 = i_0 < i_k < i_h = n & \text{for } 0 < k < h, \\ |y_j - c_{i_k}| \leq \delta & \text{for } i_{k-1} < j \leq i_k \text{ and } 1 \leq k \leq h. \end{cases}$$

That is, using constant value c_{i_k} represents y_j for $i_{k-1} < j \leq i_k$ and $1 \leq k \leq h$.

To build B, the scheme of [4, 15] greedily checks the points of the stream fragment in order and finds the maximally δ-representable stream fragment iteratively. Here, the stream fragment is only approximated by horizontal lines. To choose the maximally δ-representable fragment started from x_i, the scheme needs to find the maximum time stamp x_j such that $\max_{i \leq i_1 < i_2 \leq j} |y_{i_1} - y_{i_2}| \leq 2\delta$. In fact, the scheme constructs the longest horizontal rectangle with 2δ width starting from x_i to cover the maximal number of steam points. The intuition is depicted in Fig. 3(a) in general.

Extending the idea to error-bounded PLR, we have the following observation that can be proven easily according to the error bound definition.

Observation 1 *A stream fragment $S[i,j]$ is δ-representable if and only if there exists a parallelogram of 2δ width in the vertical direction such that no points of $S[i,j]$ are placed outside of the parallelogram.*

The intuition of this observation is illustrated in Fig. 3(b). We will use this observation to study the feasible slopes of a line segment approximating a δ-representable stream fragment through rotating the Cartesian coordinate plane.

To simplify the discussion, we assume that $S[1,k-1] = \langle s_1, s_2, \ldots, s_{k-1} \rangle$ is δ-representable, and we use a line segment to represent it. For each point $s_i = (x_i, y_i)$ of $S[1,k-1]$ where $1 \leq i < k$, its new coordinates $s'_i = (x'_i, y'_i)$, after having the axis rotated around the origin by an angle of $-\pi/2 < \theta < \pi/2$, must satisfy:

$$\begin{cases} x'_i = x_i \cos\theta + y_i \sin\theta, \\ y'_i = -x_i \sin\theta + y_i \cos\theta, \end{cases}$$

as illustrated in Fig. 3(c), which is derived by rotation matrix. According to the observation, since fragment $S[1,k-1]$ is δ-representable, there exists $-\pi/2 < \theta < \pi/2$ such that for each pair of $\{i,j\}$, $|y'_i - y'_j| \leq 2\delta|\cos\theta|$. Since $-\pi/2 < \theta < \pi/2$, $|\cos\theta| \neq 0$ and

$$|y'_i - y'_j| = |[(-x_i \tan\theta + y_i) - (-x_j \tan\theta + y_j)]\cos\theta|$$

hold, we have $|(x_j - x_i)\tan\theta - (y_j - y_i)| \leq 2\delta$. By extending this formula, we have[3]:

$$\frac{(y_j - \delta) - (y_i + \delta)}{(x_j - x_i)} \leq \tan\theta \leq \frac{(y_j + \delta) - (y_i - \delta)}{(x_j - x_i)}. \tag{1}$$

In fact, θ is the intersection angle of x-axis and the parallelogram's non-vertical edge, which is also the inclination angle of a possible δ-representative for $S[1,k-1]$ (i.e., the centerline parallel to the parallelogram's non-vertical edge). In this sense, the $\tan\theta$ is the slope of the δ-representative line segment.

Since for each pair of $\{i,j\}$, the relationship of Eq. (1) must hold, we can derive the range of $\tan\theta$. Let $\underline{slp}[1,k-1]$ and $\overline{slp}[1,k-1]$ denote the minimum and maximum line slopes of the δ-representative for fragment $S[1,k-1]$ respectively, and we have

$$\begin{cases} \underline{slp}[1,k-1] = \max\limits_{1 \leq i < j \leq k-1} \frac{(y_j - \delta) - (y_i + \delta)}{(x_j - x_i)} \\ \qquad\qquad = \frac{(y_c - \delta) - (y_a + \delta)}{(x_c - x_a)}, \\ \overline{slp}[1,k-1] = \min\limits_{1 \leq i < j \leq k-1} \frac{(y_j + \delta) - (y_i - \delta)}{(x_j - x_i)} \\ \qquad\qquad = \frac{(y_d + \delta) - (y_b - \delta)}{(x_d - x_b)}, \end{cases} \tag{2}$$

[3] Without the loss of generality, we assume that $x_i < x_j$.

Then, for a possible δ-representative line segment of $S[1,k-1]$, its slope $slp[1,k-1]$ satisfies:

$$\underline{slp}[1,k-1] \leq slp[1,k-1] \leq \overline{slp}[1,k-1]. \tag{3}$$

In fact, Eq. (3) denotes that the stream fragment $S[1,k-1]$ is δ-representable if and only if $\underline{slp}[1,k-1] \leq \overline{slp}[1,k-1]$. In the situations of $\underline{slp}[1,k-1] < \overline{slp}[1,k-1]$, $slp[1,k-1]$ can have many feasible values within the range $[\underline{slp}[1,k-1], \overline{slp}[1,k-1]]$. Hence, it is natural to discover the feasible range of the line slope in terms of $\underline{slp}[1,k-1]$ and $\overline{slp}[1,k-1]$.

Specifically, we define the extreme points as those identify the δ-representatives with extreme slopes. Based on Eq. (2), we define:

$$\begin{cases} \{a,c\} = \underset{1\leq i<j\leq k-1}{\arg\max} \frac{(y_j-\delta)-(y_i+\delta)}{(x_j-x_i)}, & (a<c) \\ \{b,d\} = \underset{1\leq i<j\leq k-1}{\arg\min} \frac{(y_j+\delta)-(y_i-\delta)}{(x_j-x_i)}, & (b<d) \end{cases}$$

where points s_a and s_b (s_c and s_d, respectively) are the rightmost (leftmost, respectively) points that satisfy $1 \leq a < c \leq k-1$ and $1 \leq b < d \leq k-1$. Assume $\underline{s_i} = (x_i, y_i - \delta)$ denotes data point with deleted tolerant error, and $\overline{s_i} = (x_i, y_i + \delta)$ denotes data point with added tolerant error, then $\overline{s_a}$, $\underline{s_c}$, $\underline{s_b}$ and $\overline{s_d}$ are the extreme points. As previously mentioned, we use $line(\overline{s_a}, \underline{s_c})$ to denote the line that passes points $\overline{s_a}$ and $\underline{s_c}$, and use $line(\underline{s_b}, \overline{s_d})$ to denote the line that passes points $\underline{s_b}$ and $\overline{s_d}$. In this way, $line(\overline{s_a}, \underline{s_c})$ and $line(\underline{s_b}, \overline{s_d})$ are the bounding lines with extreme slopes for all δ-representatives (Fig. 4).

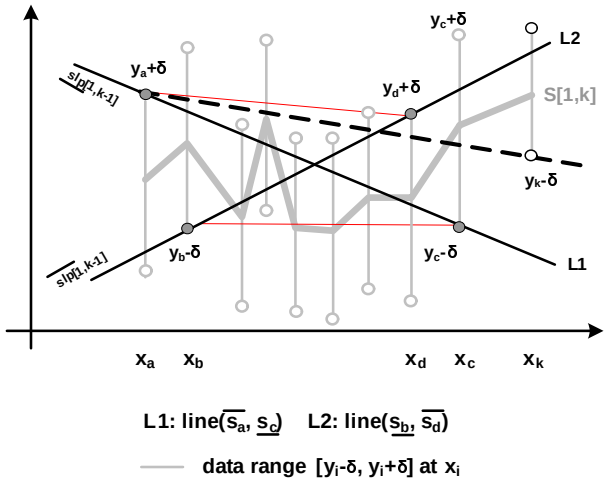


Fig. 4 Extreme slope evolution.

According to Eqs. (2) and (3), we have the following lemma:

Lemma 1. *For each $1 \leq i \leq k-1$, data range $[y_i-\delta, y_i+\delta]$ at time stamp x_i is intersected or met by line$(\overline{s_a}, \underline{s_c})$ and line$(\underline{s_b}, \overline{s_d})$.*

The proof is straightforward: If any range $[y_i-\delta, y_i+\delta]$ is not intersected or met by one of these two lines, it will be contradictory with the definition of $\underline{slp}[1,k-1]$ or $\overline{slp}[1,k-1]$.

As depicted in Fig. 4, Lemma 1 means that each data range $[y_i-\delta, y_i+\delta]$, denoted by the thick vertical gray line at x_i, interacts (or meets) with both *line*$(\overline{s_a}, \underline{s_c})$ (labeled by $L1$) and *line*$(\underline{s_b}, \overline{s_d})$ (labeled by $L2$).

3.1.2 Slope evolution and reduction

Let $S[1,k]$ be the fragment created by the addition of a new point s_k at the end of fragment $S[1,k-1]$. As inspired by Theorem 1, we need to check if $S[1,k]$ is δ-representable greedily and resolve $slp[1,k]$, $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ to derive the potential maximal δ-representative. We will interpret how to simplify the process in three stages: Increment, Localization, and the Reduction of convex hull.

First, derived from Lemma 1, we have the following corollary [4, 6] to incrementally determine whether $S[1,k]$ is δ-representable when $S[1,k-1]$ is.

Corollary 1. *Suppose fragment $S[1,k-1]$ is δ-representable, fragment $S[1,k]$ is δ-representable if and only if the range $[y_k-\delta, y_k+\delta]$ at time x_k is not located below line$(\overline{s_a}, \underline{s_c})$ or above line$(\underline{s_b}, \overline{s_d})$.*

Proof. For sufficiency, if the range $[y_k-\delta, y_k+\delta]$ at time x_k is not located below *line*$(\overline{s_a}, \underline{s_c})$ or above *line*$(\underline{s_b}, \overline{s_d})$, it is obvious that s_k can be approximated within error bound δ by at least one $seg[1,k-1]$, so $S[1,k]$ is δ-representable.

For necessity, given $S[1,k]$ is δ-representable, according to Lemma 1, data range $[y_k-\delta, y_k+\delta]$ must be intersected or met by the bounding lines of $S[1,k]$. Since $S[1,k-1]$ is always δ-representable, the bounding lines of $S[1,k]$ must be covered by those of $S[1,k-1]$, i.e., *line*$(\overline{s_a}, \underline{s_c})$ and *line*$(\underline{s_b}, \overline{s_d})$. If the range $[y_k-\delta, y_k+\delta]$ is located below *line*$(\overline{s_a}, \underline{s_c})$ or above *line*$(\underline{s_b}, \overline{s_d})$, then it will be also out of the bounding lines of $S[1,k]$, which conflicts with Lemma 1.

Combining the above two parts, the corollary is proven. □

Such corollary provides the method to quick check whether we should keep updating the extreme slopes for $S[1,k]$, or we can determine a fragment $S[1,k-1]$ as maximally δ-representable. Assuming that $S[1,k]$ is verified as δ-representable, the next question is how to obtain $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$. Computing them directly via Eq. (2) can be expensive in time cost as they require the complete computation of the intersection of slopes delineated by the equation. In the following, we will simplify the computation of $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ by incremental and localizing strategies in terms of $\underline{slp}[1,k-1]$ and $\overline{slp}[1,k-1]$.

Increment We will refine Eq. (2) and express $slp[1,k]$ in terms of $slp[1,k-1]$. From Eq. (2), we have

$$\begin{cases} \underline{slp}[1,k] = \max_{1\le i<j\le k} \frac{(y_j-\delta)-(y_i+\delta)}{(x_j-x_i)}, \\ \overline{slp}[1,k] = \min_{1\le i<j\le k} \frac{(y_j+\delta)-(y_i-\delta)}{(x_j-x_i)}. \end{cases}$$

According to the definitions of $\underline{slp}[1,k-1]$ and $\overline{slp}[1,k-1]$, we have

$$\begin{cases} \underline{slp}[1,k] = \max\limits_{1\le i<k} \{\frac{(y_k-\delta)-(y_i+\delta)}{(x_k-x_i)}, \underline{slp}[1,k-1]\}, \\ \overline{slp}[1,k] = \min\limits_{1\le i<k} \{\frac{(y_k+\delta)-(y_i-\delta)}{(x_k-x_i)}, \overline{slp}[1,k-1]\}. \end{cases} \tag{4}$$

Equation (4) indicates that $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ can be derived by comparing point s_i with s_k ($i<k$) rather than each pair of points s_i and s_j ($i<j\le k$). Therefore, the extreme slopes can be more efficiently computed from Eq. (4) than from Eq. (2). However, based on the incremental strategy, we demonstrate that this process can be further simplified, according to the Lemma 1.

Localization With the availability of $\underline{slp}[1,k-1]$, $\overline{slp}[1,k-1]$ and $\{\overline{s_a},\underline{s_c},\underline{s_b},\overline{s_d}\}$ defined before, we can further reduce the range of points needed to check for the extreme slopes. We take the update of $\overline{slp}[1,k]$ as an example to explain the reduction.

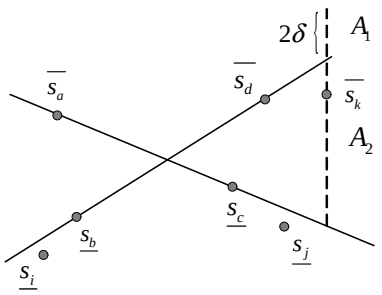


Fig. 5 Localization of slope reduction.

As exemplified in the Fig. 5, according to Corollary 1, if $S[1,k]$ is δ-representable, $\overline{s_k}$ must be within area A_1 or A_2 (as indicated by the dotted line, and the length of A_1 is 2δ). For $i\in[1,b)$, if $\overline{s_k}$ is in A_1 area, $slope(\underline{s_i},\overline{s_k}) \ge \overline{slp}[1,k-1]$ holds; if $\overline{s_k}$ is in A_2 area, the range $[y_b-\delta, y_b+\delta]$ is above $\overline{line}(\underline{s_i},\overline{s_k})$. For $i\in(c,k-1]$, the range $[y_c-\delta, y_c+\delta]$ is always above $line(\underline{s_i},\overline{s_k})$. According to the Lemma 1 and the definition of extreme slopes, if $S[1,k]$ is δ-representable, there is no need to check the points before s_b and after s_c. The similar conclusion can be made on the case of $\underline{slp}[1,k]$ update. Thus, Eq. (4) can be rewritten into

$$\begin{cases} \underline{slp}[1,k] = \max\limits_{a\le i\le d}\{\frac{(y_k-\delta)-(y_i+\delta)}{(x_k-x_i)}, \underline{slp}[1,k-1]\}, \\ \overline{slp}[1,k] = \min\limits_{b\le i\le c}\{\frac{(y_k+\delta)-(y_i-\delta)}{(x_k-x_i)}, \overline{slp}[1,k-1]\}. \end{cases} \tag{5}$$

Convex Reduction In the following, based on the definition of the extreme lines and Lemma 1, we further indicate that the computation can be constrained to those localized points on the convex hulls.

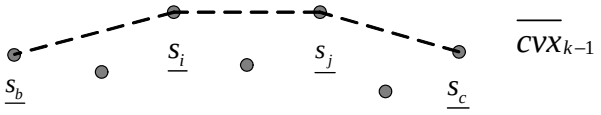


Fig. 6 Example of the convex hull points $\overline{cvx}$.

For stream fragment $S[1,k-1]$, let $\overline{cvx}_{k-1}$ denote the set of convex points of the sequence $\{\underline{s_b}, \underline{s_{b+1}}, \ldots, \underline{s_c}\}$, which are points with deleted tolerant error. Here, $\overline{cvx}_{k-1}$ bulges upward and is depicted by a sequence of point in the ascent time stamp order. For example, $\overline{cvx}_{k-1} = \langle \underline{s_b}, \underline{s_i}, \underline{s_j}, \underline{s_c} \rangle$ in Fig. 6. Similarly, we define $\underline{cvx}_{k-1}$ to be the points of the convex hull of $\{\overline{s_a}, \overline{s_{a+1}}, \ldots, \overline{s_d}\}$ that bulges downward. With the notations of $\underline{cvx}_{k-1}$ and $\overline{cvx}_{k-1}$, we have the following lemma.

Lemma 2. *If $\underline{slp}[1,k] = slope(\overline{s_i}, \underline{s_k})$ and $x_a \le x_i \le x_d$, then $\overline{s_i}$ is in $\underline{cvx}_{k-1}$. Similarly, if $\overline{slp}[1,k] = slope(\underline{s_j}, \overline{s_k})$ and $x_b \le x_j \le x_c$, then $\underline{s_j}$ is in $\overline{cvx}_{k-1}$.*

Proof. We only prove the case of $\underline{slp}[1,k]$. According to the definition of convex hull, if $\overline{s_i}$ is not in the convex hull, $line(\overline{s_i}, \underline{s_k})$ must go inside the convex hull, and there is at least one point $\overline{s_h}$ for $x_a \le x_h \le x_d$ below $line(\overline{s_i}, \underline{s_k})$. If $\underline{slp}[1,k] = slope(\overline{s_i}, \underline{s_k})$, it will be contradictory with Lemma 1 and Corollary 1, so $\overline{s_i}$ is in $\underline{cvx}_{k-1}$. □

Lemma 2 implies that Eq. (5) can be rewritten into

$$\begin{cases} \underline{slp}[1,k] = \max\limits_{\overline{s_i}\in\underline{cvx}_{k-1}}\{\frac{(y_k-\delta)-(y_i+\delta)}{(x_k-x_i)}, \underline{slp}[1,k-1]\}, \\ \overline{slp}[1,k] = \min\limits_{\underline{s_j}\in\overline{cvx}_{k-1}}\{\frac{(y_k+\delta)-(y_j-\delta)}{(x_k-x_j)}, \overline{slp}[1,k-1]\}. \end{cases} \tag{6}$$

Clearly, $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ can be more efficiently obtained from Eq. (6) than from Eq. (5) as the number of points in the convex hulls can be significantly smaller than the total number of data points in the relevant intervals.

3.2 Optimization strategies

Based on the previous discussion, we will provide some useful theorems, which derive the design of the optimal algorithm for error-bounded PLR generation. The update of extreme slopes and the convex hulls are the major points we will focus on. In the following, we will study the maintenance of $\underline{cvx}_{k-1}$ and $\overline{cvx}_{k-1}$ and the derivation of $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ based on Eq. (6). The results are demonstrated in Theorem 2. For simplicity, we will only discuss the results for the update of $\underline{cvx}_k$ and $\underline{slp}[1,k]$ as the analogous results exist for $\overline{cvx}_k$ and $\overline{slp}[1,k]$.

3.2.1 Computing extreme slopes

In the process of computing $\underline{slp}[1,k]$, we first decide whether $\underline{slp}[1,k] > \underline{slp}[1,k-1]$ and whether the bounding lines with extreme slopes need to update. Theorem 2 suggests that only the first and last points of the convex hulls need to be used to determine further processing. If the extreme slope $\underline{slp}[1,k]$ needs to update from $\underline{slp}[1,k-1]$, Theorem 2 also concludes that the computation can be incrementally performed and the time cost for computing $\underline{slp}[1,k]$ is proportional to the number of removed points from $\underline{cvx}_{k-1}$.

Theorem 2. *Suppose* $\underline{cvx}_{k-1} = \langle \overline{s_{i_1}}, \ldots, \overline{s_{i_h}} \rangle$, *here* $\overline{s_{i_1}} = \overline{s_a}$ *and* $\overline{s_{i_h}} = \overline{s_d}$, *then the following results hold for* $\underline{slp}[1,k]$ *and* $\underline{cvx}_k$.

(1) If $slope(\overline{s_{i_1}}, \underline{s_k}) \leq \underline{slp}[1,k-1]$ *then* $\underline{slp}[1,k] = \underline{slp}[1,k-1]$ *holds.*
(2) If $slope(\overline{s_{i_1}}, \underline{s_k}) > \underline{slp}[1,k-1]$ *then* $\underline{slp}[1,k] = \max_e \frac{(y_k-\delta)-(y_{i_e}+\delta)}{(x_k-x_{i_e})}$ *where* $1 \leq e \leq h$, *and* $\overline{s_{i_m}} \notin \underline{cvx}_k$ *for* $1 \leq m < e$.

Proof. The proof of claim (1): If $slope(\overline{s_{i_1}}, \underline{s_k}) \leq \underline{slp}[1,k-1]$, then point $\underline{s_k} = (x_k, y_k - \delta)$ is either below or on the line of $line(\overline{s_a}, \underline{s_c})$. Again, under the assumption, each point $\overline{s_{i_e}} \in \underline{cvx}_{k-1}$ is not below the line of $line(\overline{s_a}, \underline{s_c})$. As a result,

$$\max_{\overline{s_i} \in \underline{cvx}_{k-1}} \{ \frac{(y_k - \delta) - (y_i + \delta)}{(x_k - x_i)} \} \leq \underline{slp}[1,k-1]$$

holds, which leads to $\underline{slp}[1,k] = \underline{slp}[1,k-1]$ from Eq. (6).

The proof of claim (2): Clearly, if $slope(\overline{s_{i_1}}, \underline{s_k}) > \underline{slp}[1,k-1]$ then point $\underline{s_k} = (x_k, y_k - \delta)$ is above the line of $line(\overline{s_a}, \underline{s_c})$, and the new extreme slope can be derived from the points on the convex hull. From Eq. (6), we have $\underline{slp}[1,k] = \max_e \frac{(y_k-\delta)-(y_{i_e}+\delta)}{(x_k-x_{i_e})}$ where $1 \leq e \leq h$. Moreover, since we confirm the new extreme line is determined by $\overline{s_{i_e}}$ and $\underline{s_k}$, s_{i_e} will replace as new s_a, so all the points on the convex hull before $\overline{s_{i_e}}$ will be removed from the new convex hull $\underline{cvx}_k$. □

In fact, from the theorem and its proof, we can conclude that when we determine that the extreme slope should be updated, the new extreme slope is actually the slope of tangent line from $\underline{s_k}$ to the convex hull $\underline{cvx}_{k-1}$.

3.2.2 Updating convex hulls

If the extreme slopes are not updated when processing the new point, then the convex hulls keep unchanged. Or else if we have updated the extreme slopes after adding the new stream point, the convex hulls should be also updated. The maintenance of convex hull is formed by two parts: *point deletion* and *point addition*. We also take the updating of $\underline{slp}[1,k]$ as an example to explain the process.

The *point deletion* part has been previously mentioned in Theorem 2(2), that is, if $\underline{slp}[1,k] = slope(\overline{s_i}, \underline{s_k})$ for $\overline{s_i} \in \underline{cvx}_{k-1}$, then the earlier points of $\underline{cvx}_{k-1}$ before $\overline{s_i}$ are NOT in $\underline{cvx}_k$.

For the *point addition* part, we have $\overline{cvx}_k \subseteq \overline{cvx}'_{k-1} \cup \{\underline{s_k}\}$, where $\overline{cvx}'_{k-1}$ denotes the updated $\overline{cvx}_{k-1}$ after removing those earlier points in *point deletion*. If $\underline{s_k}$ determines the new extreme slope, it should also be added into $\overline{cvx}_k$, since it will replace as new $\underline{s_c}$ as defined before. After adding $\underline{s_k}$ to the tail of $\overline{cvx}'_{k-1}$, some previous convex points may have to be deleted in order to keep the convex characteristic. Such process can be derived from the triangle check technique in [3], which will be explained later.

3.3 Error-bounded PLR algorithm

Now we present the linear-time algorithm OptimalPLR for the error-bounded PLR problem. Let us assume that the given stream fragment has n points, and the error bound for each data point is δ. The OptimalPLR algorithm outputs a set of maximal δ-representative line segments with a minimized space cost upper bounded by n, and achieves the minimized number of segments. For the convenience of algorithm presentation, we use $\underline{\rho}$ or $\overline{\rho}$ to represent the extreme slopes during processing. In general, the strategy used in this algorithm is to progressively propagate the extreme slopes $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ upon new stream points in the process of generating maximal δ-representative segments.

3.3.1 Description of OptimalPLR

Given a stream fragment $S[1,n]$, the OptimalPLR algorithm generates maximal δ-representative line segments starting from x_1 successively. In the process of generating the maximal δ-representative segment from x_i (we assume $i = 1$ without the loss of generality), the OptimalPLR algorithm needs to maintain $\underline{cvx}_{k-1}$, $\overline{cvx}_{k-1}$, $\underline{\rho}$, and $\overline{\rho}$ to compute $\underline{slp}[1,k]$ and $\overline{slp}[1,k]$ as indicated in Sect. 3.2. In this part, we first describe the OptimalPLR algorithm and then discuss its time and space complexities.

The general steps of OptimalPLR are interpreted in Fig. 7, and we depict the procedure for constructing a maximal δ-representative line segment in Algorithm 1. Referring to the figure and the algorithm, we describe the OptimalPLR in details.

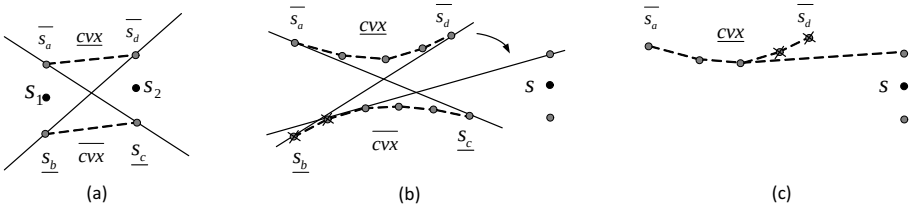


Fig. 7 Optimal algorithm: (a)Initialization; (b)Updating of extreme slopes; (c)Updating of convex hull.

Algorithm 1: OptimalPLR

Input: **S**: stream fragment starts from x_1; δ: the specified error bound
Output: A maximal δ-representative line segment starts from x_1

% **Initialization**
$s_1 = (x_1, y_1)$; $s_2 = (x_2, y_2)$;
$\overline{s_a} = (x_1, y_1 + \delta)$; $\underline{s_c} = (x_2, y_2 - \delta)$;
$\underline{s_b} = (x_1, y_1 - \delta)$; $\overline{s_d} = (x_2, y_2 + \delta)$;
$\underline{\rho} = slope(\overline{s_a}, \underline{s_c})$; $\overline{\rho} = slope(\underline{s_b}, \overline{s_d})$;
$\underline{cvx} = \langle \overline{s_a}, \overline{s_d} \rangle$; $\overline{cvx} = \langle \underline{s_b}, \underline{s_c} \rangle$;
% **Processing**
while *s is NOT outside the two lines, line*$(\overline{s_a}, \underline{s_c})$ *and line*$(\underline{s_b}, \overline{s_d})$*, more than* δ **do**
    % Maintain $\underline{\rho}$, $\overline{\rho}$, $\underline{cvx}$ and $\overline{cvx}$
    if $y + \delta < \overline{\rho}(x - x_b) - y_b$ **then**
        find the point q of $\overline{cvx}$ that minimizes $slope(q, \overline{s})$;
        let $\underline{s_b}$ be q and $\overline{s_d}$ be $\overline{s}$;
        delete all the points of $\overline{cvx}$ prior to point q;
        $\overline{\rho} = slope(\underline{s_b}, \overline{s})$;
        insert $\overline{s}$ to the tail of $\underline{cvx}$, and update $\underline{cvx}$ by triangle check([3]);
    end
    if $y - \delta > \underline{\rho}(x - x_a) - y_a$ **then**
        find the point q of $\underline{cvx}$ that maximizes $slope(q, \underline{s})$;
        let $\overline{s_a}$ be q and $\underline{s_c}$ be $\underline{s}$;
        delete all the points of $\underline{cvx}$ prior to point q;
        $\underline{\rho} = slope(\overline{s_a}, \underline{s})$;
        insert $\underline{s}$ to the tail of $\overline{cvx}$, and update $\overline{cvx}$ by triangle check;
    end
end
% **Producing a line segment**
Let $s_o = (x_o, y_o)$ be the intersection of $line(\overline{s_a}, \underline{s_c})$ and $line(\underline{s_b}, \overline{s_d})$;
$\rho = (\underline{\rho} + \overline{\rho})/2$;
return *a line segment: pass point* $s_o = (x_o, y_o)$ *with slope* ρ

Initialization. Primely we initialize the extreme lines and the convex hulls by the first two stream points s_1 and s_2 by setting $\overline{cvx} = \langle \underline{s_b}, \underline{s_c} \rangle$ and $\underline{cvx} = \langle \overline{s_a}, \overline{s_d} \rangle$. ($a = b = 1, c = d = 2$)

Extreme slope updating. As stated in Sect. 3.2, the extreme lines and the convex hulls may need to be updated when a new point s is read in. The condition of Line (8) implies that point s is in the current segment.

Taking the update of $\overline{\rho}$ as an example, since the approximate value for point s is within the range of $[\underline{s},\overline{s}]$, if $\overline{s}$ is NOT under the extreme line $line(\underline{s_b},\overline{s_d})$, the maximum slope will not be updated and the extreme line $line(\underline{s_b},\overline{s_d})$ will be used as the new extreme line. Otherwise, the maximum slope should be reduced. As exemplified in Fig. 7(b), the strategy is to find the point q of $\overline{cvx}$ that minimizes $slope(q,\overline{s})$, which can be found by the tangent line of the convex hull from $\overline{s}$. The new extreme line is then defined as $line(q,\overline{s})$ and the new $\overline{cvx}$ is updated from the old one by removing the points before q.

Convex hull updating. After updating the extreme line, $\overline{s}$ should be merged into the upper convex hull $\underline{cvx}$. Figure 7(c) depicts the merging strategy. After inserting $\overline{s}$ into the tail of $\underline{cvx}$, the triangle check [3] needs to be carried out to maintain the convex characteristic. It starts by examining the three most recent consecutive points and then moving backwards. If the middle point is above or on the line formed by the other two points, then the middle point is removed. This process is continued for the remaining three most recent consecutive points until the middle point is no longer being removed (refer to [3] for the details of convex hull algorithm).

The correctness of OptimalPLR algorithm is evidenced from both Theorem 1 and the derivation of Eq. (6) as the deduction process preserves the maximum range of slopes for δ-representative segments.

3.3.2 Complexity Analysis

In the following, we discuss the time and space complexity of OptimalPLR.

Time complexity: To show that the time complexity of the OptimalPLR algorithm is $O(n)$ for stream fragment $S[1,n]$, it is sufficient to show the time complexity of Algorithm 1 is $O(k)$ for maximally δ-representable fragment $S[1,k]$.

Clearly, the iteration times of while loop is bounded by k for fragment $S[1,k]$. In each loop, the extreme slopes ($\underline{slp}$ and $\overline{slp}$) and convex hulls ($\underline{cvx}$ and $\overline{cvx}$) need to be updated for the newly inserted stream point. As indicated in Line (10-15), the costs of updating extreme slopes are dominated by the costs of updating convex hulls. In the process of updating convex hulls, each convex hull ($\underline{cvx}$ or $\overline{cvx}$) needs to be maintained by deleting some earlier stream points from it (e.g., Line (13)) and/or inserting a recent stream point into it (e.g., Line (15)). Once a point is deleted from $\underline{cvx}$ (or $\overline{cvx}$), the point will not be inserted back into $\underline{cvx}$ (or $\overline{cvx}$). Therefore, the total costs for maintaining $\underline{cvx}$ (or $\overline{cvx}$) in the process of constructing the segment are bounded by $2k$. Thus, the time cost of Algorithm 1 on fragment $S[1,k]$ is bounded by $(2k+2k)+ck=(4+c)k$, where c is a constant number that summarizes other costs in the loop. We conclude that the time complexity of OptimalPLR is $O(n)$.

Space complexity: Since each early obtained segment is not used for latter computation and can be output directly, the space cost of the OptimalPLR algorithm on stream fragment $S[1,n]$ is proportional to the space cost on generating a maximal δ-representative segment of $S[1,n]$.

During the process of generating a segment, we only need to maintain *cvx* plus a constant number of points such as ρ, at each while loop of Algorithm 1. Therefore, the space complexity of the OptimalPLR algorithm is $O(n_{cx})$, where n_{cx} is the size of maximum *cvx*. Let n_{sg} denote the maximum number of stream points in the derived maximal δ-representative segments of $S[1,n]$. Then, we have $n_{cx} \leq n_{sg} \leq n$ holds. Therefore, the space cost of the OptimalPLR algorithm is also bounded by $O(n)$.

In summary, we have the following major result for the OptimalPLR algorithm.

Theorem 3. *Given a stream fragment $S[1,n]$ and the error bound δ, the OptimalPLR algorithm is an optimal algorithm for error-bounded PLR with $O(n)$ time and $O(n_{cx})$ space complexities where $n_{cx} \leq n$.*

3.3.3 Discussions of OptimalPLR

In this section, we will discuss some important issues of OptimalPLR algorithm, including the application for high-dimensional data and the adaption for varied error bound.

Extension for high-dimensional data

High-dimensional data are popular in streaming data nowadays, so here we discuss whether OptimalPLR can work on high-dimensional data and how it can be extended effectively.

Since the aim is to generate error-bounded PLR for data stream, it is essential how we define the error bound. Different from the case that the stream point value is a single number, for high-dimensional data, we have to consider whether the error bound is defined on each dimension, or on the data point in high-dimensional space.

Case 1: error bound defined on each dimension.
If the error bound is same for each dimension, it will be easy to extend. We can simply treat the data values of each dimension as a data stream and carry out the original OptimalPLR algorithm to generate line segments. To combine the segments of different dimensions together, when we find the first maximal δ-representative for any dimension, we stop and determine a maximally representable segment, and then start a new segmentation. Such strategy is also applied in [6].

Case 2: error bound defined on high-dimension point.
If the error bound is defined in high-dimensional space, it will be more complicated. We will only discuss the data stream in 3D space, in which each stream point at time stamp x_k is identified by $\{y_k, z_k\}$. In this case, the error bound δ will be defined as follows: For each stream point s_k, the Euclidean distance between original point

and the approximate point satisfies $\sqrt{(y'_k - y_k)^2 + (z'_k - z_k)^2} \leq \delta$. From geometrical view, the error bound describes a circle with radius δ around the point s_k, so the OptimalPLR cannot be applied for this situation directly. However, we can still find an approximated way to embed OptimalPLR.

First we have the following relationship:

$$\sqrt{2|y'_k - y_k||z'_k - z_k|} \leq \sqrt{(y'_k - y_k)^2 + (z'_k - z_k)^2} \leq \delta.$$

If we define $|y'_k - y_k| \leq \delta \Big/ \sqrt{2}$ and $|z'_k - z_k| \leq \delta \Big/ \sqrt{2}$, then it can be guaranteed that the above relationship always holds. In this way, instead of dealing with the error bound in high-dimensional space, we can define alternative error bound in each dimension, and the solution can be designed as Case 1.

It should be noticed that such strategy cannot guarantee optimal results (minimized number of segments), since we are not maintaining all feasible line segments. However, it can be executed efficiently and promise the bounded approximation error in high-dimensional space.

Adaption to varied error bound

In practical applications, the real-time approximation criteria may vary during the stream processing, so the error bound for different data points may change according to the current data status. For example, when the sampling rate increases, the system may need to reduce the approximation error, so the error bound should be reduced accordingly. In those applications with periodic variation, e.g., the traffic monitoring system, the approximation error margin for traffic flow in rush hour may be much smaller than that in slack hours.

Fortunately, OptimalPLR algorithm can be easily extended to adapt to varied error bound. During the algorithm, assume the approximate error bound for each data point s_i is δ_i. In the algorithm procedure, the parameter δ can be replaced by δ_i when processing s_i. In this way, the algorithm can execute as usual and at the same time, adapt to different error bounds.

4 ParaOptimal: an optimal algorithm in transformed space

In this section, we will introduce an interesting optimal algorithm to generate the error-bounded PLR in transformed space, named ParaOptimal [20]. It designs the algorithm in a special slope-offset parameter space, also with linear time efficiency. Furthermore, we will introduce the generalization of such algorithm model.

4.1 Description of ParaOptimal

Similarly, the data stream is a sequence of data points which are numerical numbers, and we denote it as $S = \langle s_1, s_2, \ldots, s_n \rangle$. Each data point s_i consists of the value y_i and the time stamp x_i. The approximate value of s_i can be determined by $y'_i = a \cdot x_i + b$, where the a and b respectively stand for the slope and the offset of the line segment approximating s_i. The error bound δ is defined to restrict the approximate error on single point, i.e., $\forall i \in [1,n], |y'_i - y_i| \leq \delta$.

The ParaOptimal is a greedy and incremental algorithm in which the line segments are generated and adjusted to approximate as many points as possible. The basic idea of ParaOptimal is to determine the line function by maintaining the feasible region of all line candidates in the parameter space. More specifically, for a point s_i, the approximate value should be within the error bound, that is,

$$y_i - \delta \leq a \cdot x_i + b \leq y_i + \delta.$$

It means any pair of parameters $\{a,b\}$ that meets the above inequalities can identify a candidate line function, which composes the solution set. If more data points are approximated, the solution set of line functions approximating all data points will be the intersection of all solution sets for individual points.

Considering the parameter pair in the parameter space (Fig. 8), each pair can be viewed as a point, and the error bound provides linear constraints on the parameters. The ParaOptimal algorithm runs in this way: Each new point provides two linear constraints; the algorithm incrementally updates the feasible region of candidate line parameters by intersecting the feasible region with the solution region of new linear constraints; a new line segment is determined if the feasible region is empty, and then, the algorithm initializes the feasible region for a new segmentation. Figure 8 exemplifies the feasible region update in parameter space.

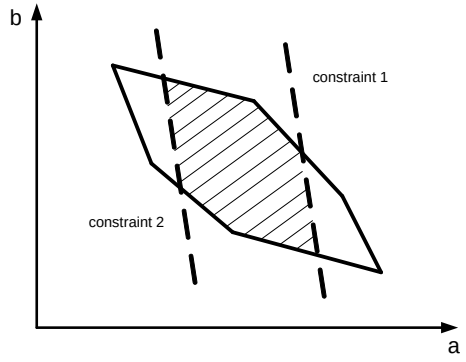


Fig. 8 Feasible region update in parameter space.

4.1.1 Theoretical preparation

Primely, we provide some basic properties and lemmas for the further description.

Lemma 3. *For the candidate line parameters, any edge of the feasible region belongs to a boundary of a linear constraint.*

Proof. Since the feasible region is the intersection of all the solution regions for all linear constraints, it is obvious that the boundaries of the feasible region are composed of the boundaries of linear constraints, so any edge is part of a boundary of a linear constraint. □

Corollary 2. *The feasible region of the candidate line parameters is a convex polygon.*

Proof. Given an edge of the feasible region, it is part of a boundary of certain linear constraint. Since the feasible region is the intersection of all solution regions of linear constraints, it must be the subset of the solution region of corresponding linear constraint, so all the feasible region lies on one side of the given edge. According to the convex definition, the feasible region is a convex polygon. □

Lemma 4. *For the boundary of linear constraints provided by error bound, the slopes are less than zero, and decreasing monotonously as the order of data points.*

Proof. Given an upper error bound of point s_i, the linear constraint is identified by $a \cdot x_i + b \leq y_i + \delta$, so the slope of the linear constraint boundary is $-x_i$. Since $x_i > 0$ always holds, the slope is less than zero.

For any two points s_i and s_j, if $x_i < x_j$, then the slope of the constraint boundary corresponding to s_i is greater than that of s_j. □

Corollary 3. *The left most point of the feasible region is the highest point; the right most point of the feasible region is the lowest point.*

Proof. Given a point on the feasible region boundary, since all the edge slopes are less than zero, it must be lower than the boundary point to its left. The rest can be done in the same manner, and we will have the point is lower than the leftmost point of the feasible region. So, the leftmost point is the highest point. Similarly, we can have the rightmost point of the feasible region is the lowest point. □

Now, we describe the ParaOptimal algorithm step by step in the following parts.

4.1.2 Initialization

In the parameter space, the error bound of a data point will provide two linear constraints for the candidate line parameters, which compose a pair of parallels. When the segmentation starts, to restrict the feasible region with bounded size, the algorithm considers the first two points for initialization and generates the feasible region. Figure 9 demonstrates this process.

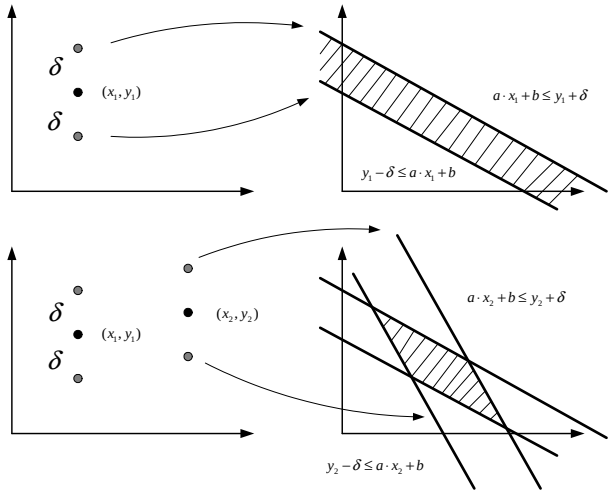


Fig. 9 Initialization of feasible region.

4.1.3 Feasible region update

When the algorithm processes a new data point, it updates the current feasible region according to the new linear constraints and checks whether it is possible to process more points, or else a line segment can be determined. For simplification, we only discuss the linear constraint by the upper error bound.

Assume we process a new data point s_i, and the linear constraint provided by the upper error bound is $a \cdot x_i + b \le y_i + \delta$. Since x_i is always greater than zero, the solution area for the above linear constraint is the half space to the left of the boundary, which is identified by linear function $a \cdot x_i + b = y_i + \delta$. As exemplified in Fig. 10 in the clockwise order, we divide the edge points of feasible region into $\langle \overline{p}_1, \overline{p}_2, \ldots, \overline{p}_t \rangle$ and $\langle \underline{p}_1, \underline{p}_2, \ldots, \underline{p}_s \rangle$ by the leftmost point p_l and rightmost point p_r. Each point is a 2-tuple $\{a, b\}$ in parameter space.

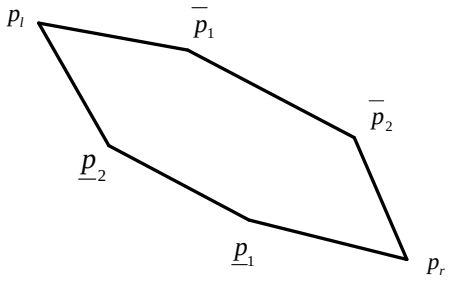
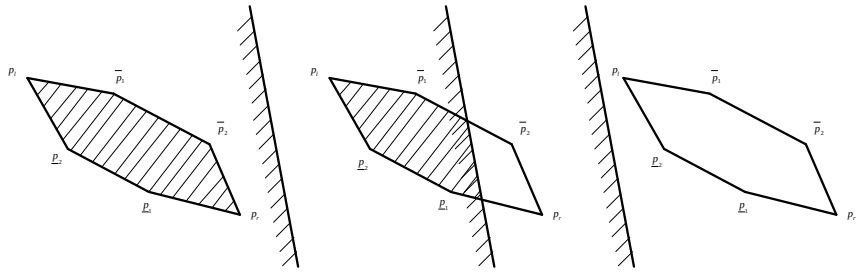


Fig. 10 Points definition of feasible region.

The feasible region update can be categorized into three cases (Fig. 11).



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 11 Three cases for feasible region update.

Case 1: $p_r.a \cdot x_i + p_r.b < y_i + \delta$. In this case, the whole feasible region satisfies the linear constraint, so the updated feasible region will remain the same and no extra processing is needed.

Case 2: $p_l.a \cdot x_i + p_l.b \leq y_i + \delta \leq p_r.a \cdot x_i + p_r.b$. In this case, the solution area of the linear constraint will intersect with the current feasible region, and the updated feasible region should be determined. Since we discuss the linear constraint from the upper error bound, we can first determine the new p_r for the new feasible region. We start from the first point of the sequence $\langle \underline{p}_1, \underline{p}_2, \ldots, \underline{p}_s \rangle$, and check backwards until the current point $\underline{p}_i$ lies on the right side of the boundary of linear constraint, that is, $\underline{p}_i.a \cdot x_i + \underline{p}_i.b \leq y_i + \delta$. Then, we calculate the intersection point generated by the boundary line of linear constraint, and the line identified by point $\underline{p}_i$ and point $\underline{p}_{i-1}$. (For $\underline{p}_1$, $\underline{p}_{i-1}$ is p_r.) We set the intersection point as new p_r. Then, the sequence of $\langle \overline{p}_i \rangle$ should be updated similarly as the previous procedure did, and then, the feasible region can be updated according to the new boundary points.

Case 3: $p_l.a \cdot x_i + p_l.b > y_i + \delta$. In this case, the whole feasible region will fall outside the solution area of linear constraint, so the updated region will be empty. The line segment can be determined, and new segmentation can be initialized.

After the feasible region update, if the new feasible region is empty, we can randomly choose a point in the old feasible region and apply the line segment identified by the chosen parameter to approximate the streaming points until the current point. The current point will be applied to initialize a new segmentation. Otherwise, the feasible region is not empty, and the algorithm will continue reading points and updating feasible region. All the procedures will work until the data stream ends.

Formally, the procedure for constructing a maximal δ-representative line segment by ParaOptimal is described in Algorithm 2.

Algorithm 2: ParaOptimal

Input: **S**: stream fragment starts from x_1; δ: the specified error bound
Output: A maximal δ-representative line segment starts from x_1

% **Initialization**
$s_1 = (x_1, y_1)$; $s_2 = (x_2, y_2)$;
$p_l = (\frac{y_2 - y_1 - 2\delta}{x_2 - x_1}, \frac{x_2 y_1 - x_1 y_2 + x_2\delta + x_1\delta}{x_2 - x_1})$; $p_r = (\frac{y_2 - y_1 + 2\delta}{x_2 - x_1}, \frac{x_2 y_1 - x_1 y_2 - x_2\delta - x_1\delta}{x_2 - x_1})$;
$\overline{p}_1 = (\frac{y_2 - y_1}{x_2 - x_1}, \frac{x_2 y_1 - x_1 y_2 + x_2\delta - x_1\delta}{x_2 - x_1})$; $\underline{p}_1 = (\frac{y_2 - y_1}{x_2 - x_1}, \frac{x_2 y_1 - x_1 y_2 - x_2\delta + x_1\delta}{x_2 - x_1})$;
Maintain the edge points of feasible region in clockwise order as p_l, $\langle \overline{p}_1 \rangle$,$p_r$,$\langle \underline{p}_1 \rangle$;
% **Processing a new point** $s = (x, y)$
while $p_l.a \cdot x + p_l.b \leq y + \delta$ **do**
    % Update the edge points of feasible region
    if $p_l.a \cdot x + p_l.b \leq y + \delta \leq p_r.a \cdot x + p_r.b$ **then**
        set the new p_r as the intersection point of line $b = -x \cdot a + y + \delta$ with the sequence $\langle \underline{p}_1, \underline{p}_2, \ldots, \underline{p}_s \rangle$;
        update the sequence $\langle \overline{p}_1, \overline{p}_2, \ldots, \overline{p}_t \rangle$ after intersecting with line $b = -x \cdot a + y + \delta$;
        set the new p_l as the intersection point of line $b = -x \cdot a + y - \delta$ with the sequence $\langle \overline{p}_1, \overline{p}_2, \ldots, \overline{p}_t \rangle$;
        update the sequence $\langle \underline{p}_1, \underline{p}_2, \ldots, \underline{p}_s \rangle$ after intersecting with line $b = -x \cdot a + y - \delta$;
    end
end
% **Producing a line segment**
Let (a_o, b_o) be one point in the feasible region;
return *a line segment:* $y = a_o \cdot x + b_o$

4.2 Generalization of ParaOptimal

The core idea of ParaOptimal is to deal with the problem in the transformed space, which can inspire a more general model to approximate the data stream.

Generally speaking, an application may need to approximate the data with an k-degree polynomial, i.e., the approximate value of s_i is determined by $y'_i = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_k x_i^k$. Given the error bound δ, we have

$$y_i - \delta \leq a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_k x_i^k \leq y_i + \delta.$$

These inequalities identify two hyperplanes in the parameter space of $\{a_i\}$ ($i = 1, 2, \ldots, k$), which are the linear constraints of the solution set. Similarly, if more data points are considered, the intersection of all sub-spaces identified by the hyperplanes can compose the final solution set. In this way, the ParaOptimal can be generalized to process the polynomial approximation of any order. Specifically, when $k = 0$, the problem is the PCR (Piecewise Constant Representation); when $k = 1$, the problem is exactly the PLR. It should be noticed that when $k \geq 2$, the algorithm will be executed in high-dimensional space, and the complexity will increase to $O(n^k)$ [34].

In addition, ParaOptimal algorithm can also support varied error bound, i.e., for point s_i, its corresponding error bound is defined as δ_i. During the processing, only

the offset of the linear constraints will be adjusted accordingly, which will not affect the algorithm mechanism.

5 Theoretical analysis of the equivalence

The OptimalPLR and ParaOptimal both achieve the optimal results with linear complexity, but process the data points in different spaces, i.e. time-value stream space and slope-offset parameter space. As a result, we are motivated to compare these two algorithms theoretically. In this part, we will formally analyze the relationship and difference between the stream space and the parameter space, and discuss the equivalence between the two spaces and these two algorithms, which is able to provide a deeper understanding for the optimal algorithm.

5.1 Mapping of two spaces

First, we put our effort to discover the relationship between the stream point space and line parameter space, namely S-space and P-space. We have the following theorem:

Theorem 4. *The line function identified by two points in S-space, is the intersection point of two lines in P-space, which are corresponding to the linear constraints brought by related points in S-space. Vise versa.*

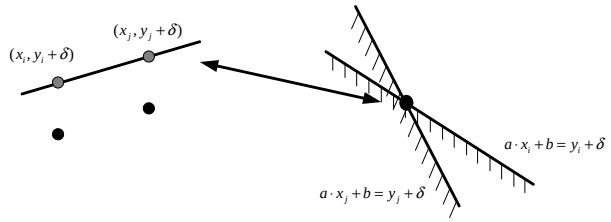


Fig. 12 Mapping between two spaces.

Proof. As denoted in Fig. 12, consider the stream points s_i and s_j, and for the line function to approximate the data points, the function parameters must satisfy the following inequality set for upper error bound:

$$\begin{cases} a \cdot x_i + b \leq y_i + \delta \\ a \cdot x_j + b \leq y_j + \delta \end{cases}$$

Therefore, the boundary lines for corresponding linear constraints in P-space are $a \cdot x_i + b = y_i + \delta$ and $a \cdot x_j + b = y_j + \delta$, respectively. In this way, we can calculate the intersection point of these two lines by combining and solving the above equations.

Notice that in S-space, the line identified by the upper error bound points of s_i and s_j is the line crossing $(x_i, y_i + \delta)$ and $(x_j, y_j + \delta)$. So, the line function parameters are just mapped to the intersection point in the P-space as described above.

With the same argument, we claim that the line function identified by two points in P-space is the intersection point of corresponding lines in S-space. □

5.2 Equivalence discussion

Based on Theorem 4, we can formally establish the equivalence between OptimalPLR and ParaOptimal algorithms.

For OptimalPLR, we model the convex hull which constrains the line candidates in the following way. As depicted in Fig. 13, the segment sequence composing the upper convex hull is denoted as $\langle \overline{q}_1, \overline{q}_2, \ldots, \overline{q}_s \rangle$, and the sequence composing the lower convex hull is $\langle \underline{q}_1, \underline{q}_2, \ldots, \underline{q}_t \rangle$. The line segments with maximum slope and minimum slope are q_h and q_l respectively.

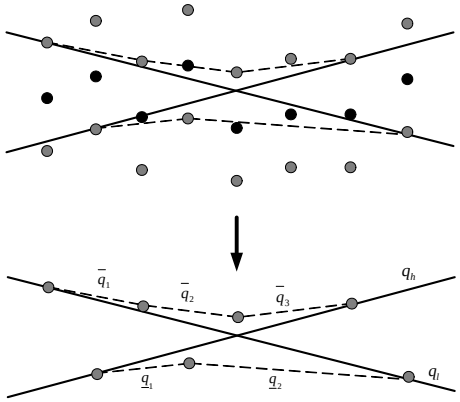


Fig. 13 Segment definition of convex hull in S-space.

Define $p \sim q$ as the mapping relationship between point p and segment q, and the following theorem illustrates the equivalence between OptimalPLR and ParaOptimal in different spaces.

Theorem 5. *The points on the feasible region in P-space are one-to-one mapped with those segments composing the convex hull in S-space. More specifically,* $\overline{p}_i \sim \overline{q}_i$, $\underline{p}_i \sim \underline{q}_i$, $p_r \sim q_h$ *and* $p_l \sim q_l$.

We apply mathematical induction method to prove the theorem.

Proof. **Step 1**: For the first two points of initialization, the theorem holds obviously (Fig. 14).

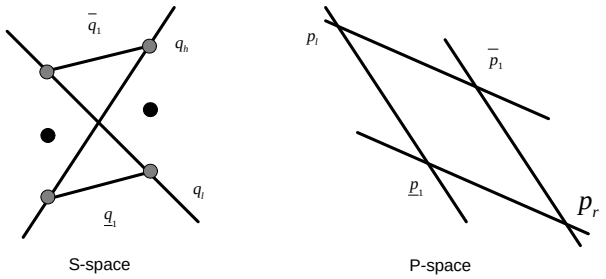


Fig. 14 Mapping for the first two points.

Step 2: Assume for the first k points, the theorem holds.

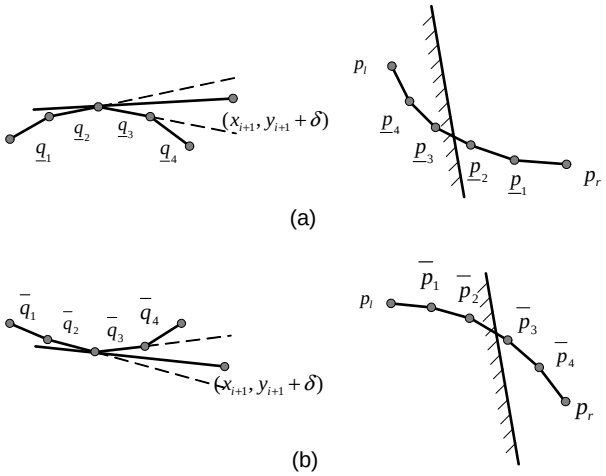


Fig. 15 Mapping for k+1 points.

Step 3: Now, we discuss when processing the $k+1$ point and still only consider the upper error bound of s_{k+1}. The case of lower error bound can be proved with similar argument.

- **Case 1**: $q_h.a \cdot x_{i+1} + q_h.b < y_{i+1} + \delta$. In this case, the segment indicating the maximum slope q_h will not change. Since $q_h \sim p_r$, Case 1 describes the same situation as the first case in Fig. 11.
- **Case 2**: $q_l.a \cdot x_{i+1} + q_l.b \leq y_{i+1} + \delta \leq q_h.a \cdot xi + 1 + q_h.b$.
  In this case, q_h should be adjusted. According to OptimalPLR, the new q_h is the tangent line from the point $(x_{i+1}, y_{i+1} + \delta)$ toward the lower convex hull.

As described in Fig. 15(a), the point of tangency is the intersection point of $\underline{q}_2$ and $\underline{q}_3$, one of which is out of the upper error bound, and the other of which is within the upper error bound. According to Theorem 4, the intersection point is the segment connecting $\underline{p}_2$ and $\underline{p}_3$. Therefore, the new q_h is mapped with new p_r. For the lower convex hull, $\underline{q}_1$ and $\underline{q}_2$ should be deleted, and also for the feasible region in the P-space, $\underline{p}_1$ and $\underline{p}_2$ are deleted. So, the rest of $\underline{q}_i$ and $\underline{p}_i$ is still mapped.

Furthermore, the upper convex hull should also change according to the new q_h. The OptimalPLR proposes to find the tangent line from point $(x_{i+1}, y_{i+1} + \delta)$ toward the upper convex hull. Similar as lower convex hull, we can find the tangency point, and $\overline{q}_3$ and $\overline{q}_4$ are deleted. Accordingly, in P-space, $\overline{p}_3$ and $\overline{p}_4$ are deleted, and the rest of $\overline{q}_i$ and $\overline{p}_i$ is still mapped.

- **Case 3**: $q_l.a \cdot x_{i+1} + q_l.b > y_{i+1} + \delta$. In this case, the convex hull to indicate the possible line segment candidates is empty, which is exactly same as the third case in Fig. 11.

Based on the above analysis, we prove that ParaOptimal and OptimalPLR are theoretically equivalent. □

The theoretical meaning above is that it guarantees that the two methods are basically the same, which reveals the inherent reason deriving the optimal approximation with linear complexity and also provides the understanding of the algorithm from different view. However, the two algorithms may have different performance in practice, due to the different ways of recording intermediate data during the processing, which results from the space they are based on. Such difference may affect the processing efficiency in both memory and time cost under certain situation.

6 Summary

In this chapter, the significant piecewise linear representation (PLR) for online stream approximation problem is investigated with given error bound in L_∞ norm. Based on the theoretical analysis, highly practical algorithms in time domain and parameter domain are introduced to achieve the optimal results with linear time and space complexity. The relationship between the time-value space and the parameter space is further investigated, which provides complete and better theoretical understanding for this classic problem. In the future, we will further study the application of PLR, and focus on the predictive stream analysis based on PLR patterns.

Acknowledgements This work is partially supported by Natural Science Foundation of China (Grant No.61602353), Natural Science Foundation of Hubei Province (Grant No.2017CFB505) and the Fundamental Research Funds for the Central Universities (Grant No. WUT:2017IVA053 and WUT:2017IVB028).

References

1. Atzori, L., Lera, A., Morabito, G.: The internet of things: a survey. Computer Networks **54**, 2787–2805 (2010)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 1–16 (2002)
3. Berg, M.D., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry Algorithms and Applications. Springer Berlin Heidelberg (2008)
4. Buragohain, C., Shrivastava, N., Suri, S.: Space efficient streaming algorithms for the maximum error histogram. In: Proceedings of the 23rd International Conference on Data Engineering, pp. 1026–1035 (2007)
5. Chen, Q., Chen, L., Lian, X., Liu, Y., Yu, J.X.: Indexable pla for efficient similarity search. In: Proceedings of the 33rd international conference on Very large data bases, pp. 435–446 (2007)
6. Elmeleegy, H., Elmagarmid, A.K., Cecchet, E., Aref, W.G., Zwaenepoel, W.: Online piecewise linear approximation of numerical streams with precision guarantees. Proceedings of the VLDB Endowment **2**, 145–156 (2009)
7. Gandhi, S., Foschini, L., Suri, S.: Space-efficient online approximation of time series data: Streams, amnesia, and out-of-order. In: Proceedings of IEEE 26th International Conference on Data Engineering, pp. 924–935 (2010)
8. Gandhi, S., Nath, S., Suri, S., Liu, J.: Gamps: compressing multi sensor data by grouping and amplitude scaling. In: Proceedings of the ACM SIGMOD International Conference on Management of data, pp. 771–784 (2009)
9. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: Surfing wavelets on streams: one-pass summaries for approximate aggregate queries. In: Proceedings of the International Conference on Very Large Data Bases, pp. 79–88 (2001)
10. Guha, S., Harb, B.: Approximation algorithms for wavelet transform coding of data streams. IEEE Transactions on Information Theory **54**, 811–830 (2008)
11. Guha, S., Shim, K.: A note on linear time algorithms for maximum error histograms. IEEE Transactions on Knowledge and Data Engineering **19**, 993–997 (2007)
12. Jagadish, H.V., Jin, H., Ooi, B.C., Tan, K.L.: Global optimization of histograms. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 223–234 (2001)
13. Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M.: Locally adaptive dimensionality reduction for indexing large time series databases. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 151–162 (2001)
14. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings of the 1st IEEE International Conference on Data Mining, pp. 289–296 (2001)
15. Lazaridis, I., Mehrota, S.: Capturing sensor-generated time series with quality guarantees. In: Proceedings of the 19th International Conference on Data Engineering, pp. 429–440 (2003)
16. Li, G., Li, J., Gao, H.: ε-approximation to data streams in sensor networks. In: Proceedings of IEEE INFOCOM, pp. 1663–1671 (2013)
17. Li, S., Xu, L.D., Zhao, S.: The internet of things: a survey. Information Systems Frontiers **17**, 243–259 (2015)
18. Nguyen, B., Abiteboul, S., Cobena, G., Preda, M.: Monitoring xml data on the web. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 437–448 (2001)
19. Olston, C., Jiang, J., Widom, J.: Adaptive filters for continuous queries over distributed data streams. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 563–574 (2003)
20. O'Rourke, J.: An on-line algorithm for fitting straight lines between data ranges. Communications of the ACM **24**(9), 574–578 (1981)

21. Paix, A.D., Williamson, J.A., Runciman, W.B.: Crisis management during anaesthesia: difficult intubation. In: Qual. Saf. Health Care (2005)
22. Palpanas, T., Vlachos, M., Keogh, E.: Online amnesic approximation of streaming time series. In: Proceedings of the 20th International Conference on Data Engineering, pp. 339–349 (2004)
23. Pang, C., Zhang, Q., Hansen, D., Maeder, A.: Unrestricted wavelet synopses under maximum error bound. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 732–743 (2009)
24. Pang, C., Zhang, Q., Zhou, X., Hansen, D., Wang, S., Maeder, A.: Computing unrestricted synopses under maximum error bound. Algorithmica **65**, 1–42 (2013)
25. Sathe, S., Papaioannou, T.G., Jeung, H., Aberer, K.: A survey of model-based sensor data acquisition and management. In: Managing and Mining Sensor Data, pp. 9–50. Springer (2013)
26. Shatkay, H., Zdonik, S.B.: Approximate queries and representations for large data sequences. In: Proceedings of the 12th International Conference on Data Engineering, pp. 536–545 (1996)
27. Soroush, E., Wu, K., Pei, J.: Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, pp. 391–400 (2008)
28. Vullings, H.J.L.M., Verhaegen, M.H.G., Verbruggen, H.B.: Ecg segmentation using time-warping. Advances in Intelligent Data Analysis Reasoning about Data **2**, 275–285 (1997)
29. Wu, H., Salzberg, B., Zhang, D.: Online event-driven subsequence matching over financial data streams. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 23–34 (2004)
30. Xie, Q., Huang, Z., Shen, H., Zhou, X., Pang, C.: Efficient and continuous near-duplicate video detection. In: Proceedings of the 12th International Asia-Pacific Web Conference, pp. 260–266 (2010)
31. Xie, Q., Huang, Z., Shen, H.T., Zhou, X., Pang, C.: Quick identification of near-duplicate video sequences with cut signature. World Wide Web Journal **15**, 355–382 (2012)
32. Xie, Q., Pang, C., Zhou, X., Zhang, X., Deng, K.: Maximum error-bounded piecewise linear representation for online stream approximation. VLDB J. **23**, 915–937 (2014)
33. Xie, Q., Shang, S., Yuan, B., Pang, C., Zhang, X.: Local correlation detection with linearity enhancement in streaming data. In: Proceedings of the ACM International Conference on Information and Knowledge Management, pp. 309–318 (2013)
34. Xu, Z., Zhang, R., Kotagiri, R., Parampalli, U.: An adaptive algorithm for online time series segmentation with error bound guarantee. In: Proceedings of the 15th International Conference on Extending Database Technology, pp. 192–203 (2012)
35. Yu, L., Li, J., Gao, H., Fang, X.: Enabling ε-approximate querying in sensor networks. Proc. VLDB Endow. **2**(1), 169 – 180 (2009)
36. Zhang, Q., Pang, C., Hansen, D.: On multidimensional wavelet synopses for maximum error bounds. In: Proceedings of 14th International Conference on Database Systems for Advanced Applications, pp. 646–661 (2009)
37. Zhou, M., Wong, M.H.: A segment-wise time warping method for time scaling searching. Information Sciences **173**, 227–254 (2005)