IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received February 3, 2018, accepted April 13, 2018, date of publication April 18, 2018, date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2828320

A Novel Segmentation and Representation Approach for Streaming Time Series

YUPENG HU[1], PEIYUAN GUAN[2], PENG ZHAN[3], YIMING DING[3], AND XUEQING LI[3]
[1]School of Computer Science and Technology, Shandong University, Tsingtao 266000, China
[2]School of Information Science and Engineering, Central South University, Changsha 410083, China
[3]School of Software, Shandong University, Jinan 250101, China

Corresponding author: Xueqing Li (xqli@sdu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402263 and Grant 91546203, in part by the National High Technology Research and Development Program of China under Grant 2014AA01A302, in part by the special funds of the Taishan Scholar Construction Project, in part by the Independent Innovation Projects of Shandong Province under Grant 2014ZZCX08102, Grant 2014ZZCX03409, and Grant 2014CGZH1106, in part by the Science and Technology Development Projects of Shandong Province under Grant 2014GGX101028, and in part by the Key Research and Development Program of Shandong Province under Grant 2015GGX101009.

ABSTRACT Along with the coming of Internet of Everything era, massive numbers of pervasive connected devices in various fields are continuously producing oceans of time series stream data. In order to carry out different kinds of data mining tasks (similarity search, classification, clustering, and prediction) based on streaming time series efficiently and effectively, segmentation and representation which segment a streaming time series into several subsequences and provide approximative representation for the raw data, should be done as the first step. With the virtue of solid theoretical foundations, piecewise linear representation (PLR) has been gained success in yielding more compact representation and fewer segments. However, the current state of art PLR methods have their own flaws: For one thing, most of current PLR methods focus on the guaranteed error bound instead of the holistic approximation error, which may lead to excessive fitting errors of segments and loss of factual research significance. For another, most of current PLR methods process streaming time series with some fixed criteria, which cannot provide a more flexible way to represent streaming time series. Motivated by the above analysis, we propose a novel continuous segmentation and multi-resolution representation approach based on turning points, which subdivides the streaming time series by a set of temporal feature points and represents the time series flexibly. Our method can not only generate more accurate approximation than the state-of-the-art of PLR algorithm, but also represent the streaming time series in a more flexible way to meet different needs of users. Extensive experiments on different kinds of typical time series datasets have been conducted to demonstrate the superiorities of our method.

INDEX TERMS Internet of Things, streaming time series, online segmentation, multi-resolution representation.

I. INTRODUCTION

Along with the coming of IoE (Internet of Everything) era, massive numbers of omnipresent connected devices (sensors, collector, etc.) in various fields are continuously producing huge amounts of streaming data(e.g., real-time transaction data in smart shopping system [1], GPS-based geographic information data [2], [3], real-time UV monitoring data [4], etc.). Accordingly, how to manage and analyze these data efficiently and effectively has not only become a huge challenge in developing IoT [5], but also greatly promoted the development of related researches [6]–[11].

Nowadays, there has been an explosion of interest in analyzing and mining streaming time series generated from IoT. Streaming time series can be described as an ordered collection of elements including the recorded values and timestamps, which are continuously generated in a high speed and potentially forever [12]. Due to the large amount, high-dimensional and continuous characteristics of streaming time series, several related data analysis and data mining researches such as time series query, similarity measure [13], pattern recognition [14], classification [15], clustering [16] and so on are incapable to do in-depth researches as they used to do in some static and small datasets.

In consideration of the above situation, the segmentation of streaming time series, which provides more compact and approximate representations of the time series raw data, should be done as the first step to reduce both the space and computational cost of storing and transmitting, also to alleviate the workload of data processing. In other words, an efficient segmentation and representation algorithm for streaming time series would be a useful preprocessing tool for other subsequent related works as follows.

- In time series similarity search and pattern recognition tasks, primitive shapes [13] and frequent patterns [17] subsequences should be represented for further similarity measuring.
- In time series classification tasks, the typical prototypes [18] and shapelet candidates [15] should be created for the predefined classes, generated by some representation approaches.
- In time series clustering tasks, some renowned heuristic methods such as K-means need to use several meaningful temporal feature sequences generated by some representation approaches to improve its convergence ability [16], [19].

Scholars have done much work on the time series representation. There are several highly cited approximate representation algorithms, including discrete fourier transform (DFT) [20], discrete wavelet transform (DWT) [21], singular value decomposition (SVD) [22] and piecewise linear representation (PLR) [23]. In these representation methods, PLR has been one of the most widely used algorithms, which divides a time series into segments and uses a linear function to approximate each segment. Compared with other methods, PLR has the advantages of lower index dimension, higher calculation speed and the ability to support efficient similarity search. In addition, PLR is more consistent with human visual experience.

At present, most existing representation-based PLR approaches do not work well for our problem. Some methods [24], [25] process segmentation on static data sets, and may incur a high cost if applied directly to streaming time series. There do exist works that address the segmentation problem for streaming data [26], which is called Sliding Windows (SW) but only for the case where the requirement of approximation is in low level. To solve this problem, Keogh *et al.* [27] consider combining the online nature of Sliding Windows and the superiority of Bottom-Up, which is called sliding window and bottom-up (SWAB). However, this algorithm treats each point of the time series equally and exists some computation redundancy. In order to speed up the efficiency of the online segmentation algorithm, Liu *et al.* [28] propose the feasible space window (FSW) and the stepwise FSW (SFSW), which introduce the concept of feasible space to find the farthest segmenting point of each segment. These methods greatly enhance the efficiency of segmentation, but the processed segments are unable to make more accurate representation than SWAB, which means that the FSW-based segmentation method cannot be used as a preprocessing tool or subroutine in plenty of follow-up data mining tasks.

Furthermore, few PLR methods dynamically return representation results according to the changing needs of users, referred to ''Multi-resolution Representation''. For example, the operating data of the satellite on orbit needs to be represented and displayed in different fitting errors and the salient points in the high frequency transaction data of stock market are eager to be retrieved according to their importance.

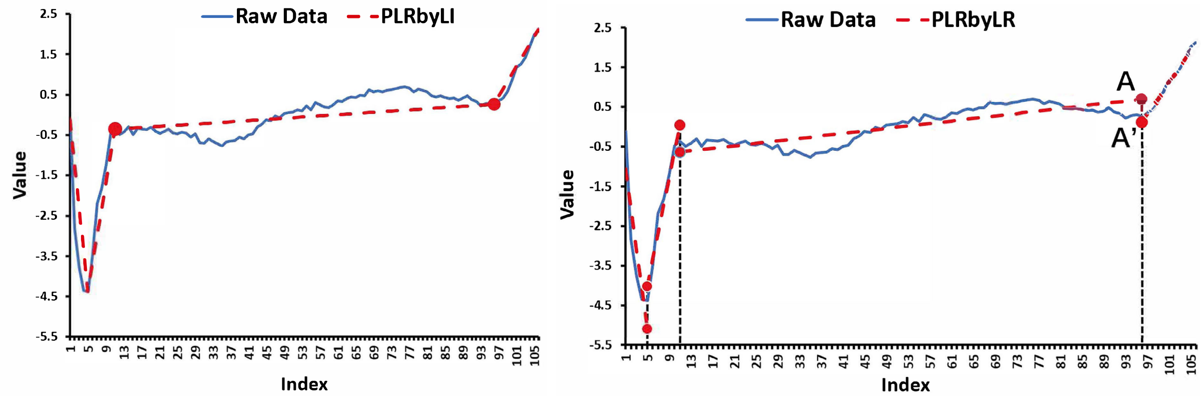
In order to equipoise the efficiency and accuracy of the online PLR method for streaming time series, we propose a novel continuous segmentation and multi-resolution representation algorithm based on turning points called CSMR_TP for short. The core idea of our approach is to refine the result of FSW, which is termed as initial segmentation (IS) in this article, by standing on a more holistic view and also provide more flexible representation results according to the diverse needs of users. Our contributions can be summarized as follows.

1) We propose a novel continuous segmentation and multi-resolution representation algorithm based on turning points (CSMR_TP), which segments streaming time series by a set of temporal feature points and maintains a higher similarity between the processed segments and the raw data. CSMR_TP provides more accurate representation than the most highly cited online PLR algorithm up to date (SWAB, FSW, SFSW).
2) We design an efficient multi-resolution index structure to provide more flexible piecewise linear representation specified by users dynamically. Moreover, CSMR_TP can be a useful preprocessing tool for other subsequent related data mining tasks. A novel acceleration strategy for time series classification by CSMR_TP will be illustrated in Section VI.
3) We compare CSMR_TP with other baseline methods on extensive typical time series datasets to demonstrate the superiority of our approach.

The remainder of the paper is organized as follows. Section II summarizes existing related works. Section III describes some preliminaries. The multi-resolution representation algorithm based on turning points (CSMR_TP) is described in detail in section IV. Section V presents the experimental results and the corresponding analyzes. A novel acceleration strategy for time series classification by CSMR_TP will be illustrated in Section VI. Finally, Section VII offers some conclusions.

II. RELATED WORK

Segmentation and representation for streaming time series can be considered as a discretization problem [29]. Compared with other methods, piecewise linear representation (PLR) is more consistent with human visual experience, which also has lower index dimension, faster calculation speed for many



(a) (b)

FIGURE 1. Fitting lines of PLR based on LI and LR. (a) PLR based on linear interpolation. (b) PLR based on linear regression.

practical applications [13]–[15]. For such reasons, PLR is more fit for continuously segmenting and approximatively representing streaming time series, which can be described as follow:

For a given time series $T = (a_1, a_2, \ldots, a_i, \ldots, a_n)$ of length n, which can be a constant value or continue to grow without limited.

The T will be divided into sequences $S = (S_1, S_2, \ldots, S_k)$ while $(1 \leq k \leq n-1)$ and be represented by a series of linear functions. The similarity between the processed subsequences and the raw data should retain at a reasonable level, which can be specified by user.

Given that PLR methods are going to approximate a time series with straight lines, there are two major approaches to find the approximating line, introduced in [27] and [29] respectively.

- Linear Interpolation (LI): The approximating line for the sequence $x = (x_1, x_2, \ldots, x_i, \ldots, x_n)$, where element $x_i = (t_i, v_i)$ indicates that the recorded value is v_i at the time t_i, is the straight line connected from x_1 to x_n, which can be completed in constant time.
- Linear Regression (LR): The approximating straight line for the same sequence x is the best fitting line, which can be obtained by using the least squares strategy proposed by [30] and can be completed in linear time depending on the length of sequence.

In order to make the differences between LI and LR more clear, we select both of them separately in PLR with the identical ECG time series dataset in [31]. The comparative results are illustrated in Fig. 1. The PLR result based on LI is shown in Fig. 1(a), and the PLR result based on LR is shown in Fig. 1(b) respectively. It is obvious that LI can provide a more smooth piecewise approximation, on the contrary, LR can produce a very disjointed look on some data points. The biggest difference between the above two is the fitting errors of segmentation points, in other words, the fitting error of single point would be presented as two points at the identical time stamp. For example, in Fig. 1(b), point A and point A' at $index_{96}$, which means the fitting error of the segmenting point (fitting error of single points) is uncertain. In general, compared with LR, PLR with LI has the aesthetic superiority together with lower computational complexity, which can be used in some kinds of visualization applications [32]–[34]. In contrast, PLR with LR can provide relatively lower fitting errors of segments than PLR with LI, the cumulative fitting error of segments in Fig. 1(b) is 23.53 less than the cumulative fitting error which is 31.55 in Fig. 1(a).

In this paper, we focus on the descriptions of multi-resolution piecewise linear representation based on turning points, so either LI or LR can be utilized as the approximation technique in CSMR_TP. In order to display the multi-resolution data representation more clearly, we select Linear Interpolation for algorithm description and experimental analysis in follow-up sections.

Currently, all of PLR methods can be subdivided into two main segmentation strategies: Offline PLR and Online PLR.

A. OFFLINE PLR

Offline PLR: These algorithms mainly focus on the piecewise linear representation for all kinds of static time seres, in other words, the whole datasets need to be collected before PLR conducting. According to the different linear representation strategies on static datasets, Offline PLR methods can also be subdivided into the following two categories:

- PLR based Top-Down algorithm (PLR-TD) [35]: PLR-TD begins with an unsegmented sequence and introduces one cutting point at a time, repeating this process until the stopping criterion is met. According to the temporal features of time series data, Ji et al propose a new piecewise linear representation based on important data point (PLR-IDP) for time series, which finds the important points by calculating the fitting errors of single points and segments [25]. As a consequence of this, the fitting error of PLR-IDP is much smaller than traditional PLR-TD methods.
- PLR based Bottom-Up algorithm (PLR-BU) [36]: PLR-BU begins with n-1 segments, and then two adjacent segments are merged into one by choosing

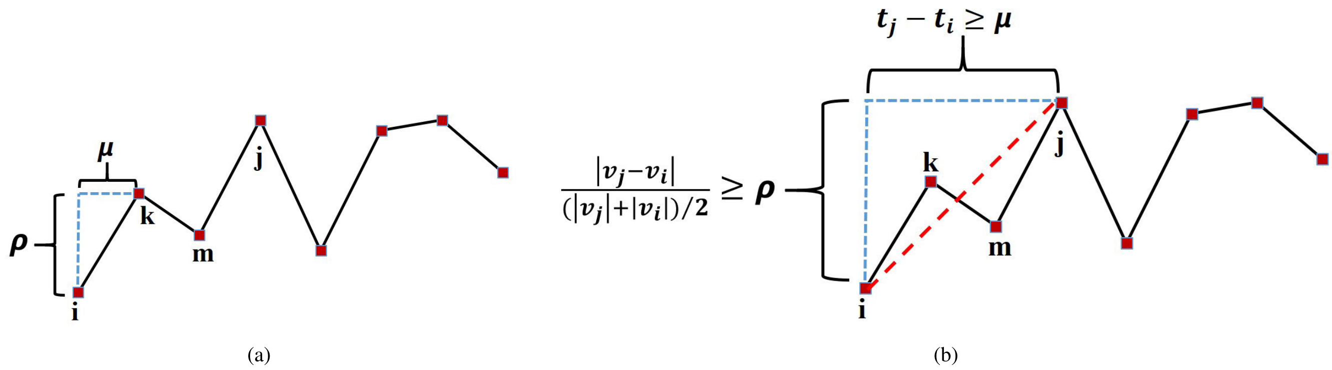


FIGURE 2. The definition of turning points. (a) The definition of μ and ρ. (b) The TP selection from TPCs.

the lowest cost of merging pairs, iterating this process until the lowest merge cost of two adjacent segments monotonically increases up to the threshold.

B. ONLINE PLR

Onine PLR: These algorithms mainly focus on PLR for streaming time seres. Instead of collecting whole data sequence in the beginning, the real-time arriving data sequence can be acquired and piecewise linear represented at the same time. The traditional online PLR algorithm is PLR based on slide window (PLR-SW) [37], which initializes the first data point of time series as the left endpoint of a segment and then trying to find the right endpoint by sequential scanning time series data. However, the results of segmentation are less than satisfactory, due to the lacking of global view segmentation on the whole dataset. To solve this problem, Keogh *et al.* [27] combines the online nature of SW and the superiority of BU together called sliding window and bottom-up (SWAB) to improve the fitting precision of PLR-SW. In order to speed up the efficiency of online PLR, Liu *et al.* [28] propose the feasible space window (FSW) and stepwise FSW (SFSW) method, which propose the concept of feasible space to find the farthest segmenting point of each segment. FSW method greatly enhances the efficiency of segmentation, but the processed subsequences are unable to make more accurate representation, because the fitting errors of segments have not been considered.

Through the comparative analysis of the above algorithms, the major advantage of Online PLR is the ability to continuous segmentation, which meets our requirement for streaming time series segmentation. The main problem of them is the accuracy of representation which cannot be guaranteed, compared with its offline counterparts. Moreover, there is limited capability to provide a more flexible representation to meet the diverse needs from different users by the above online PLR methods.

III. PRELIMINARIES

A. THE TURNING POINTS DEFINITION

For a time series $T = (a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n)$, $1 \leq i \leq j \leq n$. Each a_i in T is arriving at a specific time instance t_i and the length of T denoted as n is growing continuously. When we plan to divide the streaming time series into some continuous subsequences, it is advisable for us to segment the stream according to their temporal features (e.g., the financial streaming time series can be partitioned by the trend of price rise, fall or flat, the satellite orbital time series data stream can be segmented by the change of perigee and apogee etc.). The temporal feature is constructed by a sequence of data points and each point actually has the different influence on the variation trend. Judging from the intuition, these local maximum and minimum points seemingly can be defined as the TPs since they indicate the change in the trend of the time series which has been proposed by Yin *et al.* [38] on the financial time series. Moreover, considering the efficiency of the multi-resolution index structure construction, we optimize the definition of turning points (TPs) in [39], which can be described as follow.

1) T could be segmented by initial segmentation(IS) [28] to form a series of segments $S = (S_1, S_2, S_m, \ldots, S_k)$ while $(1 \leq m \leq k \leq n-1)$. Supposing a_i and a_j are the begin point and the end point of S_m respectively, a_i and a_j would be selected as TPs.
2) Except a_i and a_j, all the local maximum and minimum points,inflection points and step points within the scope of S_m would be selected as TP candidates called TPCs for short. Assuming the TPC $a_x = (t_x, v_x)$ has been selected as TP, $a_y = (t_y, v_y)$ is a subsequent TPC of a_x. If a_x meets the following inequations, it should be selected as TP in this paper.

$$t_y - t_x \geq \mu \tag{1}$$

$$\frac{|v_y - v_x|}{(|v_y| + |v_x|)/2} \geq \rho \tag{2}$$

μ denotes the distance between two TPCs in time, ρ denotes the fluctuation of two TPCs in trend, as shown in Fig. 2(a).

With the help of Equation (1) and Equation (2), we could not only discard unimportant fluctuations, but also retain all major peaks and valleys in the original streaming time series. As shown in Fig. 2(b), a_x has been selected as TP in S_m, we separately measure the time intervals and the trend fluctuations between a_x and its subsequent TPCs, which would be compared with μ and ρ respectively. In this instance, starting from a_k, Equation (1) and Equation (2) are not satisfied concurrently until a_y, so a_y would be selected as TP and two TPCs (k,m) can be smoothed. In other words, we can directly draw a straight line from the point i to point j so as to discard unimportant fluctuations in stream.

After a_y has been selected as TP, we would continue to find the next TP from the subsequent TPCs of a_y in the same way. Analogously, all the TPs in S_m could be found completely. In addition, there is one more thing that needs to be clearly noted that all time series date sets for analyzing in our paper have already been cleaned by smoothing-based data cleaning strategies introduced in [40]. For such reason, the influence of outliers and data noise has already been ignored right from the start.

B. THE SEGMENTATION AND REPRESENTATION CRITERIA

To ensure the effect of segmentation and the efficiency of multi-resolution, There are two criteria to evaluate the goodness of fit for a potential segment, which would be introduced below.

1) THE MAXIMUM ERROR FOR SINGLE POINT (ME_SP)

ME_SP is used to evaluate the fitting error of the single data in segment. In the traditional Online PLR methods, a segment continues to grow until the maximum vertical distance (MVD) for a certain data point exceeds ME_SP. Therefore, we could utilize ME_SP to generate an initial segment (named S) in IS process and identify all TPCs and TPs in S at the same time.

2) THE MAXIMUM ERROR FOR ENTIRE SEGMENT (ME_ES)

ME_ES is used to evaluate the fitting error for the entire segment, and we use this segmentation criterion to eliminate the insufficiency for only relying on ME_SP to segment the streaming time series, in other words, ME_SP could only guarantee the fitting error of single point under a certain threshold, but fail to control the fitting error of the entire segment in a reasonable range. Consequently, ME_ES should be utilized to guarantee the holistic accurate representation for each segment.

IV. ALGORITHM

In this section, we will describe our continuous segmentation and multi-resolution representation algorithm based on turning points(CSMR_TP) in detail. CSMR_TP can be divided into three main steps as follows.

A. INITIAL SEGMENTATION AND TURNING POINTS SELECTION

The initial segmentation can divide the streaming time series into several segments by IS and ensure the fitting error of each point is under ME_SP. The initial segmentation could find all TPs (the definition of TPs in Section III), which can be utilized for optimizing the initial segmentation results. We can take part of time series monitoring data for a cold storage facility of Longda Foodstuff Group Co., Ltd (referred to as Longda in what follows) with the prespecified threshold value for ME_SP is 30.0 and ME_ES is 50.0 for CSMR_TP. Fig. 3(a) and Fig. 3(b) describe this process, the red dots in Fig. 3(b) denote the initial segmenting points and the red dash lines denote the initial segments, whose fitting errors of each single points are limited no more than ME_SP. At the same time, all the TPCs and TPs in these initial segments have also been identified, shown in Fig. 3(c).

B. SEGMENTATION REFINING AND INDEXING

After the above process, we will use the ME_ES to evaluate the fitting errors of initial segments by accumulating the fitting errors of single points. We can make use of TPs to refine the initial segmentation results. The final piecewise linear segmentation result is shown in Fig. 3(d) and the final fitting errors of each segment is strictly limited to ME_ES below. More importantly, the multi-resolution index structure can be constructed simultaneously, the initial segmentation results and the iterative refining results can also be stored into the index to prepare for providing a more flexible piecewise linear representation dynamically.

Considering the main steps of segmentation, the trend of fitting accuracy could be characterized as the gradual progress from rough to refined, in other words, the multi-resolution index should provide a more flexible PLR to the same time series subsequence within a range of fitting accuracy.

According with the main steps of CSMR_TP, the multi-resolution index can also be constructed from top to bottom. We define some objects of the index in advance, and then we introduce the establishment process in code.

Objects of *Segment*: *Segment* would denote all kinds of PLR segments (no matter initial segments, iterative segments and final segments) in CSMR_TP, which are the basis for the establishment of index. Therefore, *Segment* is expressed as $Segment = \{p_b, p_e, s_{error}, s_{avg}, tpcList, tpList\}$. The attributes of *Segment* are described in Table 1.

TABLE 1. Attributes of ***segment***.

attributes	description
p_b	The begin point of the segment.
p_e	The end point of the segment.
s_{err}	The fitting error of the segment.
$tpcList$	All TPCs in the segment.
$tpList$	All TPs in the segment.

Objects of *IndexNode*: All *Segment*s should be defined as *IndexNode*s before inserted into index tree. The object *IndexNode* can be designed as $IndexNode = \{seg, subNodeList, parentNode\}$ and the attributes of *IndexNode* are described in Table 2.

Objects of *IndexTree***:** All *IndexNode*s are planned to put into index tree, and the object *IndexTree* is defined as $tree = \{root\}$, *root* is an object of *IndexNode*.

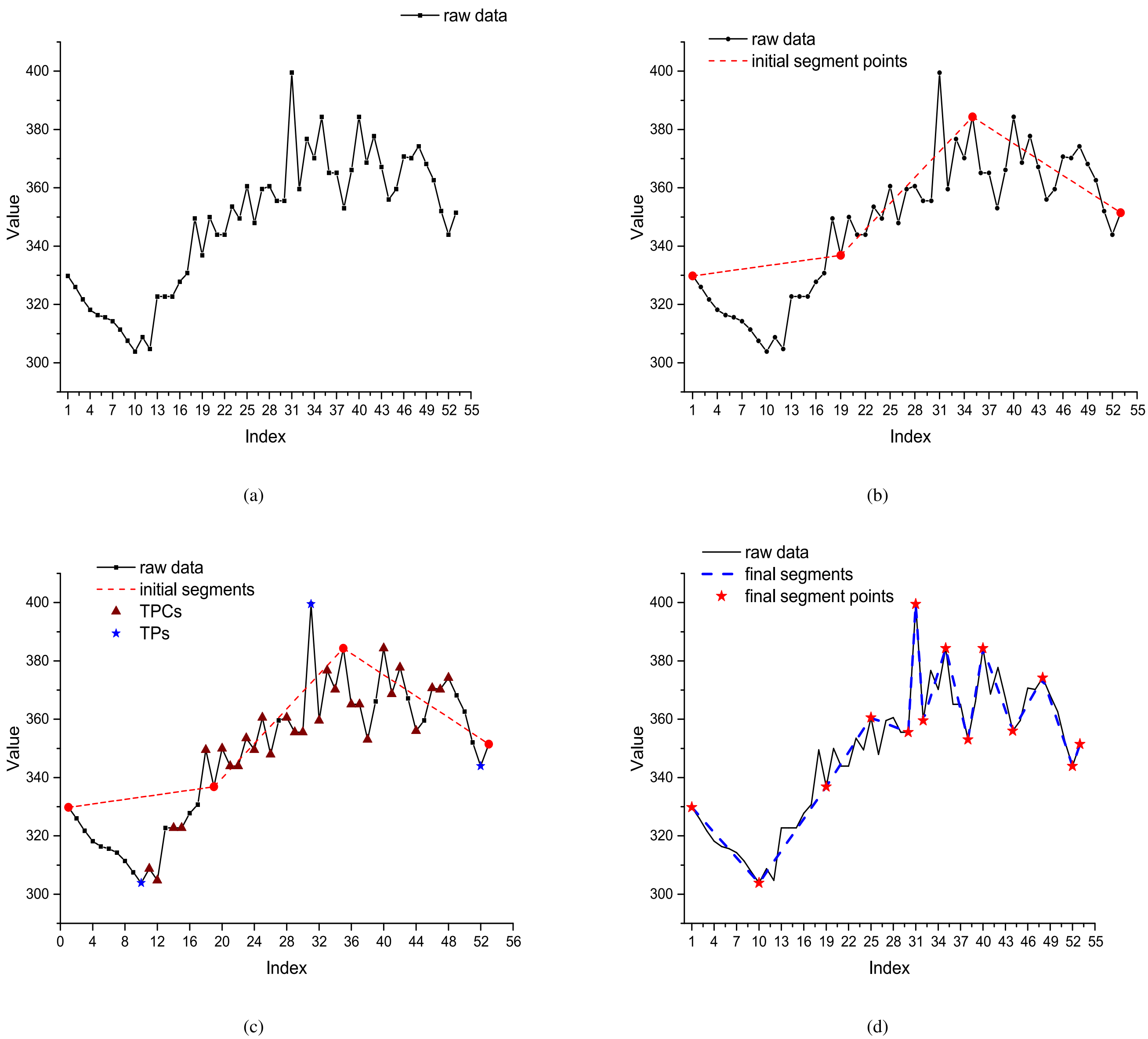


FIGURE 3. The major steps of CSMR_TP. (a) Part of time series monitoring data of Longda. (b) Initial segmentation on the raw data. (c) Identification for TPCs and TPs. (d) Final segmentation on the raw data.

TABLE 2. Attributes of *indexnode*.

attributes	description
seg	The object of Segment.
$subNodeList$	The chlid IndexNodes list of current IndexNode.
$parentNode$	The parent IndexNode of current IndexNode.

The whole multi-resolution PLR method for streaming time series is shown in Algorithm 1. Streaming time series will be ordered into the prespecified buffer, meanwhile, not only initial segments would be generated by IS operation [28], all the TPCs and TPs in each initial segment would also be identified simultaneously. After that, with the help of TPCs and TPs in each initial segment, the initial segments could be evaluated and refined by standing on a more holistic view, the multi-resolution *IndexTree* could also be constructed concurrently. After the above two steps have been completed, buffer would be refreshed to remove part of the processed time series and to receive new time series. Repeat the above operation until the entire stream has been processed completely.

In Line 2 and Line 6 of Algorithm 1, we initialize the variables p_b and select the begin point p_b as TP. From Line 7 to Line 29, repeat a cycle of operations:

1) if p_{cur} is a TPC or TP, it would be added in *tpcList* or *tpList* respectively.
2) Utilized the slope between p_b and p_{cur} to calculate the fitting error of p_{cur} named $error_{sp}$.(more details in IS [28])

Algorithm 1 Continuous Segmentaion and PLR

Require: Streaming time series $X = (x_1, x_2, ę, x_i, \cdots)$,
the prespecified buffer *buf* with length n,
the threshold value ME_SP and ME_ES.
Ensure: *root* of *indextree*

1: **do**
2: **if** $null \neq X$ **then**
3: $buf \Longleftarrow X$
4: $p_b = x_1$;
5: $p_{cur} = p_{b++}$;
6: *tpLlist*.add(p_b);
7: **while** *buf* is not full **do**
8: **if** p_{cur} is a TPC point **then**
9: *tpcLlist*.add(p_{cur});
10: **if** p_{cur} is a TP point **then**
11: *tpList*.add(p_{cur});
12: **end if**
13: **end if**
14: $error_{sp}$ = calcSlope(p_b,p_{cur});
15: **if** ME_SP $\geq error_{sp}$ **then**
16: $p_{cur} = p_{cur++}$;
17: $error_{sp}$ = calcSlope(p_b,p_{cur});
18: **else**
19: $p_e = p_{cur}$;
20: *tpList*.add(p_e);
21: Segment *seg*= new Segment(p_b,p_b,*tpcList*,*tpList*);
22: *segInitList*.add(*seg*);
23: *tpcList*.flush();
24: *tpList*.flush();
25: $p_b = p_{cur}$;
26: $p_{cur} = p_{cur++}$;
27: *tpList*.add(p_b);
28: **end if**
29: **end while**
30: **if** *segInitList.length* > 0 **then**
31: indexTreeConstruction(*segInitList*,ME_ES);
32: **end if**
33: refreshBuf(*buf*);
34: **end if**
35: **Until** no more time series flow in *buf*
36: return *root*

3) Determine whether $error_{sp}$ exceeds ME_SP.
4) If the value does not exceed ME_SP, p_b and p_{cur} would in the identical initial segment.
5) Otherwise, p_{cur} is the end point of the current segment, which would be added into initial segment list *segInitList*. p_{cur} is also the begin point of the next segment to prepare for the next initial segmentation.

When *buf* is full, all the initial segments in *segInitList* would be utilized to construct *IndexTree* from Line 30 to Line 31. The related subroutine is shown in Algorithm 2. After that, *buf* would be refreshed in Line 33 to prepare for receiving the subsequent time series.

Algorithm 2 IndexTree Construction

Require: *segInitList* stores all initial segments,
the threshold value of ME_ES.

1: **if** *null* $\neq$ *segInitList* **then**
2: p_b = *segInitList*.get(0).getBegin();
3: p_e = *segInitList*.get(length-1).getEnd();
4: Segment *seg*= new Segment(p_b,p_e);
5: IndexNode *parentNode* = new IndexNode(*seg*);
6: **if** *root* == *null* **then**
7: *root* = *parentNode*;
8: **else**
9: *parentNode*.setParent(*root*);
10: *root*.setRange(*parentNode*);
11: **end if**
12: **while** *segInitList.length* > 0 **do**
13: Segments *s* = *segInitList*.getindex(0);
14: IndexNode *cur* = new IndexNode(*s*);
15: **if** *s*.getErr() > ME_ES **then**
16: refinebyTPs(*cur*,$s.tp_{list}$,$s.tpc_{list}$,ME_ES);
17: **end if**
18: *cur*.setParent(*parentNode*);
19: *parentNode*.setRange(*cur*);
20: *segInitList*.remove(0);
21: **end while**
22: **end if**

Repeat the above operation until the entire stream has been processed completely. Finally, *root* will be return for multi-resolution representing.

The construction algorithm of the multi-resolution index is shown in Algorithm 2. After the initial segmentation, we can evaluate the fitting errors of the initial segments and create *IndexNodes* concurrently. In Line 2 and Line 3 of Algorithm 2, the begin point p_b of first initial segment and the end point p_e of last initial segment would be gotten to form a new segment *seg*, which could be used to create a new *parentNode* subsequently. From Line 6 to Line 11, the relation between *root* and *parentNode* in *IndexTree* has been determined. From Line 12 to Line 22, repeat a cycle of operations, which could not only create a series of *IndexNode* correlated with segments in *segInitList*, but also refine the original PLR of the initial segments, whose s_{err}s are much larger than ME_ES.

1) Obtain the Segment *s* from *segInitList*.
2) Create a new *IndexNode cur*.
3) Determine whether the fitting error of *s* exceeds ME_ES.
4) If the value does not exceed ME_ES, *cur* could be inserted in *IndexTree* immediately, as a child *IndexNode* of *parentNode*.
5) Otherwise, *s* would be subdivided into subsequences by TPs to ensure the fitting errors of such sequences are no more than ME_ES. In this process, a series of children nodes, correlated with above subsequences, have also been created, whose relationship with current

IndexNode cur in *IndexTree* needs to be further determined. The corresponding subroutine in Line 19 of Algorithm 2 is shown in Algorithm 3.

Algorithm 3 Refine the Fitting Error of Current Segment

Require: the current IndexNode *curNode*, *tpList* and *tpcList* of *curNode*, the threshold value of ME_ES.

1: List *listForTP* = *tpList*;
2: **while** *listForTP*.length $\neq 0$ **do**
3:     int *begin* = *listForTP*.get&remove(0);
4:     int *end* = *listForTP*.get&remove(1);
5:     double *err* = calcErr(*begin*,*end*);
6:     *curtpcList* = getTPC(*begin*,*end*,*tpcList*);
7:     Segment *subSeg* = new Segment(*begin*,*end*,*err*,$curtpc_{list}$);
8:     **if** *err* $\leq$ ME_ES **then**
9:         IndexNode *subNode* = new IndexNode(*subSeg*);
10:         *curNode*.setRange(*subNode*);
11:         *subNode*.setParent(*curNode*);
12:     **else**
13:         List *listForTPC* = *curtpcList*;
14:         *subSegList* = *bottomUP*(listForTPC ME_ES);
15:         **while** *subSegList*.length $\neq 0$ **do**
16:             Segment *seg* = *subSegList*.get&remove(0);
17:             IndexNode *node* = new IndexNode(*seg*);
18:             *curNode*.setRange(*node*);
19:             *node*.setParent(*curNode*);
20:         **end while**
21:     **end if**
22: **end while**

After all the segments in *segInitList* have been processed, all the fitting errors of segments are no more than ME_ES, all the *IndexNodes* correlated with the above segments have also been inserted into the appropriate locations of *IndexTree*.

The specific process for refining the initial segments,whose s_{err}s are much higher than ME_ES, is shown in Algorithm 3. When a segment is decided to subdivide into a series of subsequences to reduce the original s_{err}, all the *TPs* and *TPCs* would be utilized to accelerate the process for reducing s_{err} of segment. From Line 2 to Line 23 of Algorithm 3, repeat a cycle of operations.

1) Generate a series of subsegments by all the TPs in *listForTP*.
2) Determine whether the s_{err} of *subSeg* exceeds ME_ES.
3) If the value does not exceed ME_ES, a related new *IndexNode subNode* would be created and inserted into *IndexTree* immediately, as a child *IndexNode* of *curNode*.
4) Otherwise, *subSeg* would be subdivided into subsequences by TPCs for further processing. In this situation, *subSeg* would be refined by the traditional PLR-BU method [36] to ensure all the fitting errors of subsegments in *subSegList* are no more than ME_ES. After that, a series of *IndexNode*s, correlated with above subsegments, have also been created and inserted into the appropriate locations of *IndexTree*, as children nodes of *curNode*.

With the help of the above three algorithms, online segmentation for streaming time series has been finished completely and the index *IndexTree* has also been constructed concurrently. After that, *IndexTree* could provide a more flexible piecewise linear representation to meet the different needs of users. Although the requirements for piecewise linear representation vary wildly, which can be subdivided into two main categories: the restriction on fitting error and the restriction on the number of segments. Accordingly, the corresponding algorithms based on the above two aspects will be given.

Algorithm 4 MPLR by the Specified Fitting Error of Segment

Require: *root* of *indextree*, the specified ME_ES.
Ensure: *list* for multi-resolution PLR.

1: List *subList* = *root*.getSubNodes();
2: **while** *subList*.length $\neq 0$ **do**
3:     *childNode* = *subList*.get&remove(0);
4:     *seg* = *childNode*.getSeg();
5:     **if** *seg*.getErr()$\leq$ ME_ES **then**
6:         *list*.add(*seg*);
7:     **else**
8:         *adjustList* = readjustErr(*childNode*,ME_ES);
9:         *list*.merge(*adjustList*);
10:     **end if**
11: **end while**
12: **return** *list*;

The MPLR based on different ME_ES is shown in Algorithm 4. According to the specified fitting errors of segments, the corresponding PLR will be given. The basic principles of the method are as follows.

1) Get all the children nodes of *root* into *subList*.
2) Determine whether the s_{err} of each *child* in *subList* exceeds ME_ES.
3) If the value does not exceed ME_ES, the *seg* of *child* would be added into *list*.
4) Otherwise, the fitting error of *seg* would be further adjusted. The corresponding subroutine in Line 8 of Algorithm 4 is shown in Algorithm 5.
5) The adjusted subsegments of *seg* would also be added into *list*.

When the cycle ends, *list* will be return, in which the s_{err} of each segment in *list* is no more than ME_ES.

The specific process for readjusting the fitting error of segment in Algorithm 5 could be subdivided into five steps as follow.

1) Generate all the children nodes of *curNode* into *subList*.
2) Determine whether the s_{err} of segment in each *child* exceeds ME_ES.

Algorithm 5 Readjust Fitting Errors of Segments

```
Require: curNode of indextree, the specified ME_ES.
Ensure:  relist which has been adjusted.
List subList = curNode.getSubNodes();
while subList.length ≠ 0 do
    int tag = 1;
    subNode = subList.get&remove(0);
    seg = subNode.getSeg();
    if seg.getErr() ≤ ME_ES then
        relist.add(seg);
    else
        adjustList = readjustErr(subNode,ME_ES);
        if null ≠ adjustList then
            relist.merge(adjustList);
        else
            relist = reCalcErrByBU(subNode,ME_ES);
        end if
    end if
end while
return relist;
```

Algorithm 6 MPLR by the Specified Number of Segments

```
Require: root of indextree, the specified number of
         segments segNum.
Ensure:  segList for multi-resolution PLR.
prioList = root.getSubNodes();
while prioList.length < segNum do
    node = prioList.get&remove(0);
    subNodeList = node.getSubNodes();
    prioList.merge(subNodeList);
end while
for all node in prioList do
    seg = node.getSeg();
    segList.add(seg);
end for
sortByOrder(segList)
while segList.length > segNum do
    lowcost_Pair = findAdjSegPair(segList);
    segList.merge(lowcost_Pair);
end while
return segList;
```

3) If the value does not exceed ME_ES, *seg* of *child* would be added into *relist*.
4) Otherwise, all the children nodes of *child* would be traversed recursively to continue the above judgment. In the worst case, the s_{err} of the segment in *child* is still higher than ME_ES and *child* is already a leaf node, the segment in *child* would be processed by the traditional PLR-BU method.
5) When the cycle ends, *relist* will be return in Line 8 of Algotitham 4, in which the s_{err} of each subsegment is no more than ME_ES.

The MPLR based on different number of segments is shown in Algorithm 6. The basic principles of the method are as follows.

1) Get all the children nodes of *root* into the priority list *prioList*, in which the nodes are ordered by s_{err} of their own segments.
2) Determine whether the length of *prioList* exceeds *segNum*.
3) If the value does not exceed *segNum*, *node* with the largest s_{err} would be removed out of *prioList* and all children nodes of *node* in the next level, would be added into *prioList* concurrently.
4) Otherwise, the segments of each *node* in *prioList* would be added into *segList* and arranged in chronological order.
5) merge the lowest merging cost of two adjacent segments in *segList* until the length of *segList* is equal to *segNum*.

segList will be return, whose length meets the number requirement.

C. THE PERFORMANCE ANALYSIS OF CSMR_TP

The cost of the representation is depending on the time complexity of the algorithms. The time complexities of several highly cited online PLR algorithms up to date can be compared. For a given time series T of n data points, the time complexity of FSW is $O(m*n)$, the time complexity of SFSW is $O(m*n^2)$, and the time complexity of SWAB is $O(l*n)$, where m is the number of piecewise linear segments and l is the average length of the time series sequences.

For CSMR_TP, time complexity evaluation can be divided into three main steps:

1) The time complexity for initial segmentation, which can be done by scanning the sequences in buffer with the number of segments (m). So the time cost of this operation is $O(m*n)$.
2) The evaluation of s_{err} of initial segments should consider the number of segmenting points. In our method the buffer can store about k segments (k+1 segmenting points) in average, so the time complexity of the evaluation step is $O(k*l)$.
3) The refining segments begin with TPs and TPCs, which is similar to SWAB, but our method optimizes the process from two aspects. For one thing, the merging operation begins with the consecutive sequences instead of connecting two adjacent points, for the other, not only the leftmost segment would be removed from the buffer, the consecutive segments behind the leftmost can also be removed to obtain as few segments as possible. With the above optimization, the efficiency of our approach can be significantly improved compared with the SWAB. Subsequently, the time complexity for searching a proper position for a new *IndexNode* in *IndexTree* is $O(logl)$. The *IndexNode* creation and insertion can be completed in unit time, so the time complexity is $O(1)$. According to the above analysis, the time complexity of the whole process is no worse than $O(l*n*logl)$.

Through the above analysis, the time complexity of MPLR-IDP can be approximated as $O(l * n * logl)$.

V. EXPERIMENT AND ANALYSIS

In this section, we evaluate the performance of our online segmentation algorithm: CSMR_TP by compared with the SWAB, FSW and SFSW, which are nearly the most highly cited online segmentation algorithm based on the PLR up to date.

A. DATASET AND EVALUATION METRICS

In order to accomplish the experiment, we select some kinds of typical time series datasets which contain an average of 10,000 records from different fields, including medicine, finance, industry provided by UCR Time Series Archive [31], and we also choose some representative industrial streaming time series including the monitoring data of Jinan municipal steam heating system(JMSHSD) from December 2013 to March 2016 (i.e.,100,532 data points), the monitoring data of Dong Fang Hong satellite (DFHSD) from January 2015 to June 2015 (i.e.,320,675 data points), which is the Chinese satellite dataset provided by China Academy of Space Technology.

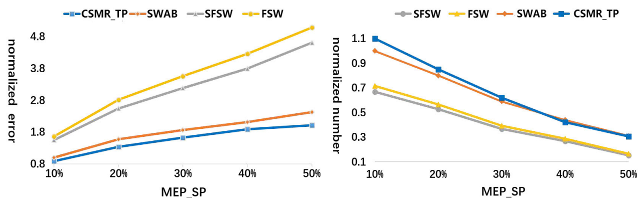
In our experiment, the goal of our segmentation is to minimize the holistic representation error and obtain as few segments as possible in a more efficient manner. Therefore, in order to evaluate the segmentation for a given streaming time series, we consider the representation error, as well as the number of segments based on two parameters: the ME_SP and ME_ES.

B. COMPARISON WITH EXISTING METHODS

We compare our methods with three baseline methods (FSW, SFSW, SWAB) by varying the ME_SP and ME_ES. Before the experiments, considering the variety and complexity of the datasets, some conditions need to be defined in advance.

First of all, we adopt the Maximum Error Percentage for Single Point (MEP_SP) to substitute the absolute ME_SP, which is proposed by Liu *et al.* [28] and MEP_SP can eliminate the influences on different data sets by specifying the percentage of the value range on different data sets. Whats more, with the change of the MEP_SP, the ME_ES in CSMR_TP should also be changed simultaneously to avoid that the ME_SP exceeds the ME_ES, so we will adopt the Maximum Error Percentage for Entire Segment (MEP_ES) to substitute ME_ES, and the MEP_ES is set as an integral multiple of N which is an integer greater than 2. Last but not least, we consider the result of the SWAB with MEP_SP whose value is 10% as the benchmark (set as 1), and we can normalize the results of other methods with the benchmark.

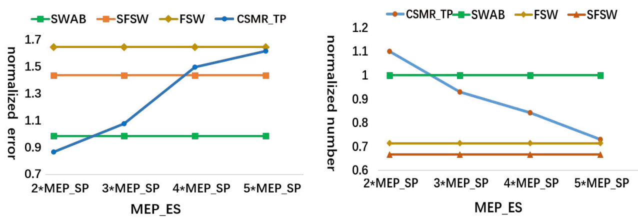
When we vary the MEP_SP from 10% to 50% in Fig. 4(a), we can see that the CSMR_TP has the lowest the normalized representation error, which means this method can provide more accurate representation than other methods, and the error of all methods gradually increase with the rising of MEP_SP. However, the error of CSMR_TP grows more



(a) (b)

FIGURE 4. The MEP_SP analysis. (a) Normalized error w.r.t MEP_SP. (b) Normalized number w.r.t MEP_SP.

slowly than the other three methods because of the restriction of the MEP_ES, in other words, the CSMR_TP can guarantee a relative accurate representation even though the single point fitting error is constantly increased. In Fig. 4(b), we can also find that the normalized number of segments of all methods gradually decrease with the rising of MEP_SP, and when the value of MEP_SP is up to 40%, the number of CSMR_TP is less than the SWAB because of the optimal merging step in our method.



(a) (b)

FIGURE 5. The MEP_ES analysis. (a) Normalized error w.r.t MEP_ES. (b) Normalized number w.r.t MEP_ES.

When we vary the MEP_ES from 2*MEP_SP to 5*MEP_SP in Fig. 5(a), the three baseline methods are shown as three straight lines parallel to the X axis, because these methods do not use MEP_ES to segment the time series. However, the CSMR_TP relies on MEP_ES to refine the result of initial segmentation. We can see that the normalized representation error of CSMR_TP constantly increases with the gradually loosened restriction of MEP_ES and the error of CSMR_TP is close unlimitedly to the FSW. In Fig. 5(b), the normalized number of segments of CSMR_TP continues to decrease with the change of the MEP_ES, finally the number of segments of CSMR_TP is nearly the same as the FSW, which means that the segmentation effect of CSMR_TP would be no worse than FSW, even in the worst case.

Besides, CSMR_TP can provide a more flexible piecewise linear representation based on different conditions (ME_ES or number of segmenting points) to meet the different needs for data analysis and data mining. In order to further illustrate this point, part of time series monitoring data of Longda Foodstuff Group Co., Ltd (referred to as Longda in what follows) is shown in Fig. 6(a), and the multi-resolution index of Longda is shown in Fig. 6(b). The fitting errors of segments are also shown in each *indexNode* in Fig. 6(b).

185.0 of *node*6 in Fig. 6(b) denotes the fitting error of segment,which begins at point 19 and ends at point 31.

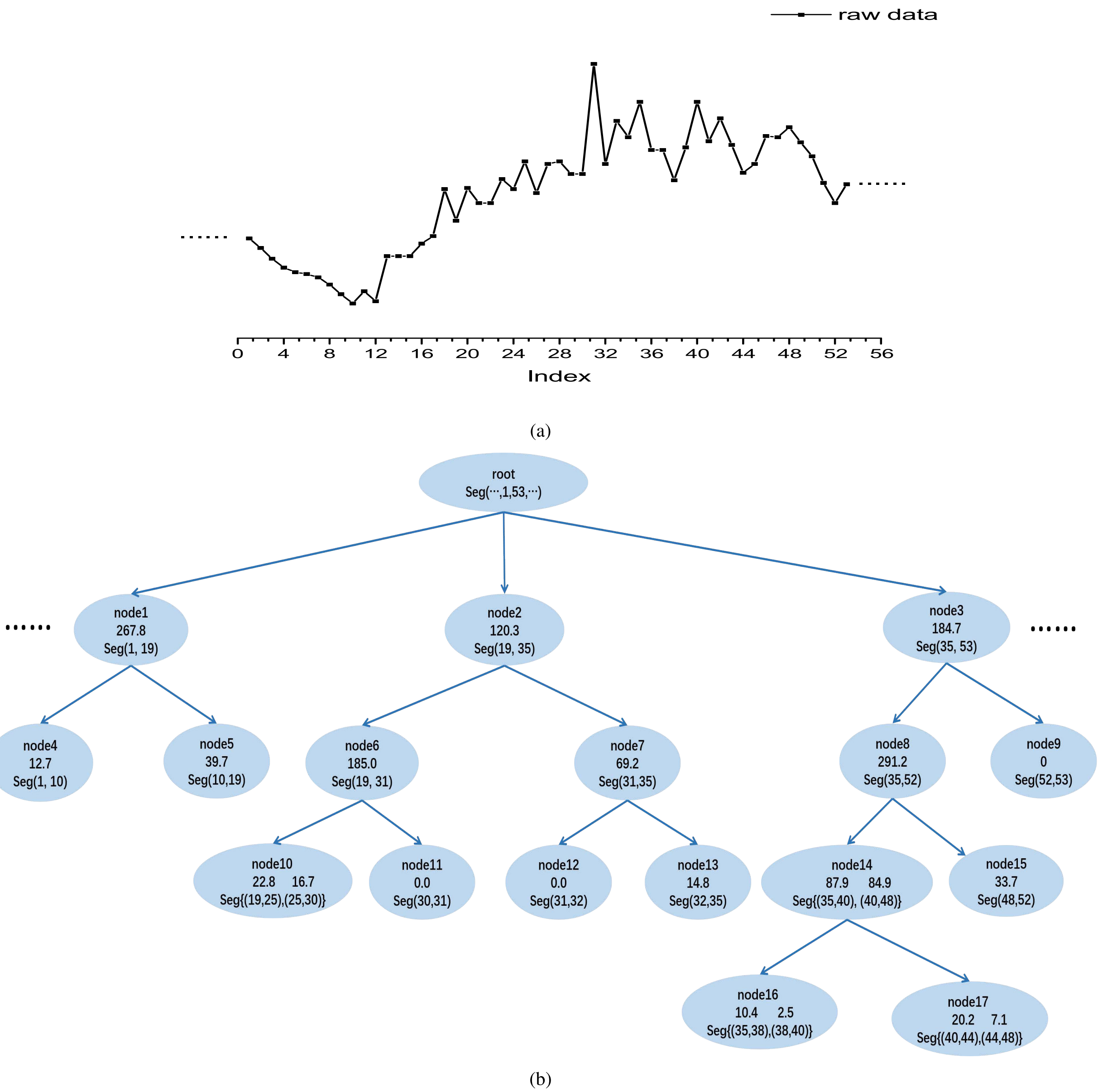


FIGURE 6. The multi-resolution index. (a) Part of time series monitoring data of Longda. (b) The multi-resolution index of Longda.

With the help of index Tree, we can not only obtain various piecewise linear representation with different segmenting points, but also get diverse PLR with different ME_ES in the same time series. For instance, we can specify PLR in Longda with the restriction that the fitting errors of segments should not exceed 80.0 shown in Fig. 7(a).We can also specify PLR in Longda by six numbers of segmenting points shown in Fig. 7(b), whose fitting errors of segments are also labeled in Fig. 7.

VI. EXTENSIONAL APPLICATION

In this section, the effectiveness of CSMR_TP as a preprocessing tool for time series classification would be pointed out and further analyzed.

Time series classification (TSC) has been attracting great interest over the past decade. While dozens of techniques have been introduced, recent empirical evidence has strongly suggested that shapelets based TSC algorithms outperform many previous TSC algorithms in terms of accuracy,efficiency and interpretability [41]. According to the concept of shapelets in [42], Shapelets are not only subsequences extracted from one time series, but also have distinctly representative characteristics of class membership. With the help of shapelets, TSC can utilize the similarity between two shapelets, rather than the similarity between two entire time series, to complete time series classification. In consequence, the overall performance of these shapelet based TSC methods can be greatly enhanced,

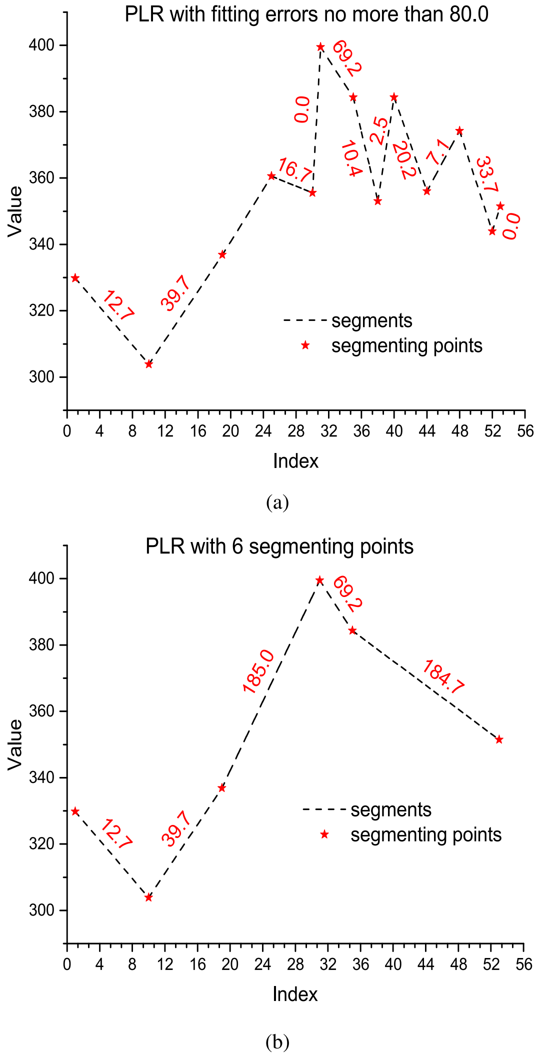


FIGURE 7. The multi-resolution PLR for time series.

moreover the appropriate shapelets can provide enough information to make the results of classification more explainable. Therefore, after that, an evolutionary algorithm by utilizing shapelets for TSC have been proposed [43]. Shapelet Transformation(ST), which not only optimizes the process of shapelets evaluation, but also allows various classification strategies(SVM,Random forest,etc.) to be adopted to classify time series objects after the shapelets selection process has been completed [44]. In other words, the shapelets selection process and the classification operation are relative independence between each other. After the corresponding transformation is completed, a larger number of off-the-shelf classification algorithms could be applied in ST. However, the running time of ST is still large, which would definitely hamper the classification efficiency and obstruct the scalability of ST. To be honest, previous studies have also been carried out to improve the efficiency of ST [44]–[46]. However, up to now, previous research works are mainly focus on reducing the time complexity on evaluating every shapelet candidate, whilst there are still too much shapelet candidates waiting for evaluation. As shown in [44] and [47], nearly all subsequences of time series are selected as shapelet candidates. Therefore, the overall efficiency of ST has not been well improved. There is still a challenge in how to select appropriate representative subsequences as shapelets from the whole data set effectively and efficiently. Obviously, The key for a subsequence to be selected as a shapelet lies in whether it can represent the corresponding temporal features clearly enough, which could be distinctly represented the corresponding characteristics of class membership.

Motivated by the above analysis, considering the definition of TP (in Section III), which can reflect the temporal features of the segment, we utilize TPs in the index tree constructed by CSMR_TP to simplify the number of shapelets candidates as much as possible, which can not only speed the shapelet selection process, but also further improve the overall efficiency of ST.

A. CSMR_TP FOR SHAPELET SELECTION

According to the above analysis, supposing that there are n representative time series named RTS, which are the original shapelets candidates for ST. Without loss of generality, RTS can be expressed as $RTS = \{T_1, T_2, \cdots, T_j, \cdots, T_n\}$ $(1 \leq j \leq n)$. In order to simplify all the original shapelets candidates, CSMR_TP would be used on RTS to construct the corresponding index tree $IndexTree$.

The profound implication of the shapelet definition in [47] implied that shapelets candidates should be refined as subsequences which can be represented a specific class maximally. In other words, instead of the entire T_j, some subsequences which have certain significances in T_j would be further extracted to represent class [48]. With the help of $IndexTree$, all the TPs in each T_j can be completely identified to help the generation of subsequences. As [48] described, a shapelet must have a certain significance. In other words, subsequences with no obvious features can be filtered out from the shapelets candidates.

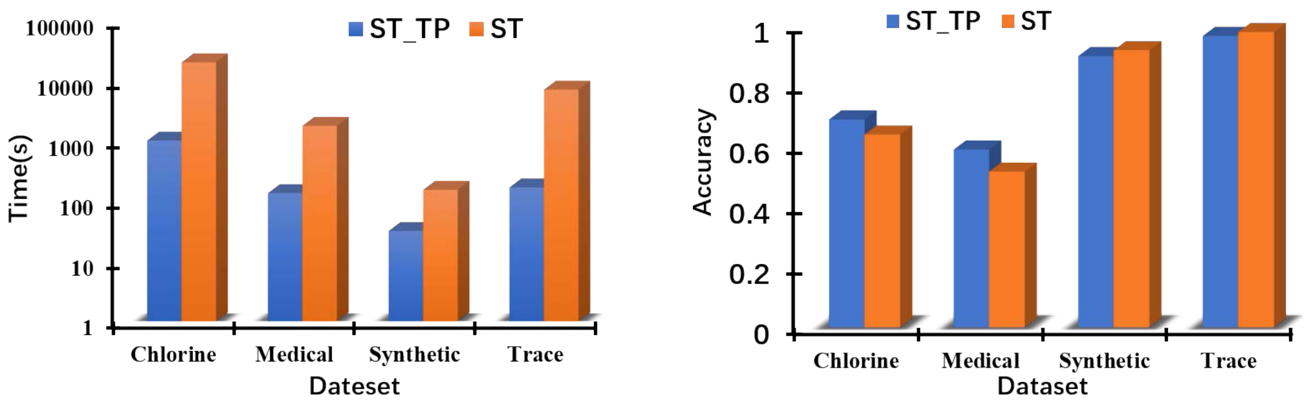
Based on this principle, with the help of TPs, we can generate a series of subsequences named $subs_i$ for each T_i in RTS.

There are two rules to be followed to generate each subsequence s belonging to $subs_i$.

1) the begin point and the end point of s must be TPs.
2) the above two points cannot be the adjacent TPs.

After all the subsequences in each $subs_i$ have been generated completely, the new data collection $SubTS$, expressed as $SubTS = \{sub_1, sub_2, \cdots, sub_i, \cdots, sub_n\}$ $(1 \leq i \leq n)$ has also been formed simultaneously, which could be used for classification instead of original RTS. Moreover, We can further use $informationgain$ [49] to select K shapelets in $SubTS$ to form the final shapelets collection FS for ST. The algorithm for the entire process is shown in Algorithm 7, which could be subdivided into three steps as follow.

1) Get all TPs of each T in RTS.
2) Generate a series of sequences from each T by TPs to form the entire $SubTS$



(a) (b)

FIGURE 8. **Performance comparison on different datasets. (a) Run time comparison on different datasets. (b) Accuracy comparison on different datasets.**

Algorithm 7 Shapelets Selection Based on TPs

Require: The original shapelets candidates $RTS = \{T_1, T_2, \cdots, T_j, \cdots, T_n\}$, the *IndexTree* for *RTS*

Ensure: *cSet* for all shapelets.

```
cList = Φ
for all T in RTS do
    tpList = IndexTree.getRoot().Search(T);
    tpSet.add(tpList);
end for
for i = 1 to tpSet.length do
    tpList = tpSet.get(i);
    subList = subTS(tpList,RTS.get(i));
    candidates.add(subList);
end for
cList = infoJudge(candidates,K);
return cList;
```

3) Select K shapelets from *SubTS* with information gain as the final shapelets for ST.

B. PERFORMANCE ANALYSIS

According to the above analysis, the original shapelets selection process has been replaced by our accelerate strategy based on TPs to speed up the efficiency of shapelets selection. In order to distinguish the above two methods, the new proposed method would be named as ST_TP, which has the identical post-processing in ST. We will conduct a series of comparative experiments on 4 highly cited datasets in different fields from the UEA & UCR Time Series Classification Repository [50]. In all comparison experiments, the number of shapelets(k) is set at the half number of time series sequences(n) in the training data set, i.e., $k = n * 50\%$.

In these experiments, we use Rotation Forest as classifier to complete TSC. The corresponding comparison results on running time for shapelets selection are shown in Fig. 8 (a) and the comparison results on the entire classification accuracy are shown in Fig. 8 (b) respectively.

According to the results in Fig. 8 (a), it is obvious that the efficiency of ST_TP is higher than ST by more than one order of magnitude in average. The efficiency of ST_TP on *Chlorine* (ChlorineConcentration) is lower than that of ST_TP on the other three datasets. The reason is that in *Chlorine*, the corresponding number of TPs is also more and the efficiency of selection would be relatively lower than the other three cases.

According to the results in Fig. 8 (b), we can find that the classification accuracies of ST_TP on *Trace* and *Synthetic* (SyntheticControl) are slightly lower than that of ST on the same datasets. The reason is that different numbers of TPs and TPs with disparate degrees of importance would affect the corresponding classification accuracy, therefore, for a certain dataset with the relatively small number of TPs, the corresponding classification accuracy would be reduced.

In general, the average classification accuracy based on the above two methods are basically in the same level, which means that using shapelets instead of entire time series in TSC would not cause a substantial decline in the accuracy of classification results, but will greatly improve the corresponding classification efficiency.

VII. CONCLUSION

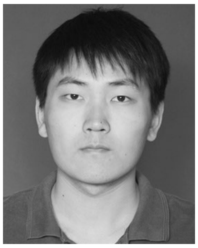
In this paper, we propose a novel online segmentation and multi-resolution algorithm (CSMR_TP) which performs well on segmenting streaming time series, and holds the main characteristics of time series. More importantly, CSMR_TP can provide a more flexible piecewise linear representation to meet the diverse needs from different users. Furthermore, TPs in the multi-resolution index can be used to improve the efficiency of time series classification. The extensive numeric experiments demonstrate the advantages of our algorithm. In future, we plan to utilize other characterization methods to further improve the efficiency of CSMR_TP, and use this algorithm as a preprocessing tool for time series classification and time series anomaly detection.

REFERENCES

[1] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng, "IoT applications on secure smart shopping system," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1945–1954, Dec. 2017.
[2] N. Capurso, T. Song, W. Cheng, J. Yu, and X. Cheng, "An Android-based mechanism for energy efficient localization depending on indoor/outdoor context," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 299–307, Apr. 2017.
[3] Z. Lv *et al.*, "Managing big city information based on WebVRGIS," *IEEE Access*, vol. 4, pp. 407–415, 2016.
[4] B. Mei, R. Li, W. Cheng, J. Yu, and X. Cheng, "Ultraviolet radiation measurement via smart devices," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 934–944, Aug. 2017.
[5] Y. Sun and A. J. Jara, "An extensible and active semantic model of information organizing for the Internet of Things," *Pers. Ubiquitous Comput.*, vol. 18, no. 8, pp. 1821–1833, 2014.
[6] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of Things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, Mar. 2016.
[7] Y. Sun, J. Zhang, and R. Bie, "Measuring semantic-based structural similarity in multi-relational networks," *Int. J. Data Warehousing Mining*, vol. 12, no. 1, pp. 20–33, 2016.
[8] Y. Sun, C. Lu, R. Bie, and J. Zhang, "Semantic relation computing theory and its application," *J. Netw. Comput. Appl.*, vol. 59, pp. 219–229, Jan. 2016.
[9] Y. Sun, H. Yan, C. Lu, R. Bie, and Z. Zhou, "Constructing the Web of events from raw data in the Web of things," *Mobile Inf. Syst.*, vol. 10, no. 1, pp. 105–125, 2014.
[10] G. Xu, Z. Wu, G. Li, and E. Chen, "Improving contextual advertising matching by using Wikipedia thesaurus knowledge," *Knowl. Inf. Syst.*, vol. 43, no. 3, pp. 599–631, Jun. 2015.
[11] G. Li, Z. Cai, X. Kang, Z. Wu, and Y. Wang, "ESPSA: A prediction-based algorithm for streaming time series segmentation," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6098–6105, Oct. 2014.
[12] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel, "Online amnesic approximation of streaming time series," in *Proc. 20th Int. Conf. Data Eng.*, Apr. 2004, pp. 339–349.
[13] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 493–498.
[14] Y. Hu, C. Ji, M. Jing, and X. Li, "A K-motifs discovery approach for large time-series data analysis," in *Proc. Asia–Pacific Web Conf.*, 2016, pp. 492–496.
[15] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with COTE: The collective of transformation-based ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015.
[16] U. M. Fayyad, C. Reina, and P. S. Bradley, "Initialization of iterative refinement clustering algorithms," in *Proc. KDD*, 1998, pp. 194–198.
[17] J. Lonardi, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in *Proc. 2nd Workshop Temporal Data Mining*, 2002, pp. 53–68.
[18] Z. Wu *et al.*, "An efficient Wikipedia semantic matching approach to text document classification," *Inf. Sci.*, vol. 393, pp. 15–28, Jul. 2017.
[19] G. Li, L. Jiang, Z. Wu, and Y. Wang, "Finding time series discord based on bit representation clustering," *Knowl.-Based Syst.*, vol. 54, no. 4, pp. 243–254, Dec. 2013.
[20] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Foundations of Data Organization and Algorithms*. 1993, pp. 69–84.
[21] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proc. 15th Int. Conf. Data Eng.*, Mar. 1999, pp. 126–133.
[22] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," in *ACM SIGMOD Rec.*, vol. 26, no. 2, pp. 289–300, Jun. 1997.
[23] E. J. Keogh and P. Smyth, "A probabilistic approach to fast pattern matching in time series databases," in *Proc. KDD*, 1997, pp. 24–30.
[24] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Proc. 19th Int. Conf. Data Eng.*, Mar. 2003, pp. 429–440.
[25] C. Ji, S. Liu, C. Yang, L. Wu, L. Pan, and X. Meng, "A piecewise linear representation method based on importance data points for time series data," in *Proc. IEEE 20th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2016, pp. 111–116.
[26] C. Wang and X. S. Wang, "Supporting content-based searches on time series via approximation," in *Proc. 12th Int. Conf. Sci. Statist. Database Manage.*, Jul. 2000, pp. 69–81.
[27] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov./Dec. 2001, pp. 289–296.
[28] 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.
[29] T.-C. Fu, "A review on time series data mining," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, 2011.
[30] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in *Proc. 12th Int. Conf. Data Eng.*, Feb./Mar. 1996, pp. 536–545.
[31] Y. Chen *et al.* (Jul. 2015). *The UCR Time Series Classification Archive*. [Online]. Available: http://www.cs.ucr.edu/~eamonn/time_series_data/
[32] Z. Lv, A. Tek, F. Da Silva, C. Empereur-Mot, M. Chavent, and M. Baaden, "Game on, science—How video game technology may help biologists tackle visualization challenges," *PLoS ONE*, vol. 8, no. 3, p. e57990, 2013.
[33] Z. Lv, A. Halawani, S. Feng, H. Li, and S. U. Réhman, "Multimodal hand and foot gesture interaction for handheld devices," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 1s, Sep. 2014, Art. no. 10.
[34] Z. Lv, A. Halawani, S. Feng, S. ur Réhman, and H. Li, "Touch-less interactive augmented reality game on vision-based wearable device," *Pers. Ubiquitous Comput.*, vol. 19, nos. 3–4, pp. 551–567, Jul. 2015.
[35] E. J. Keogh and M. J. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Proc. KDD*, vol. 98. 1998, pp. 239–243.
[36] S. Park, D. Lee, and W. W. Chu, "Fast retrieval of similar subsequences in long sequence databases," in *Proc. Workshop Knowl. Data Eng. Exchange (KDEX)*, 1999, pp. 60–67.
[37] Y. Qu, C. Wang, and X. S. Wang, "Supporting fast search in time series for movement patterns in multiple scales," in *Proc. 7th Int. Conf. Inf. Knowl. Manage.*, 1998, pp. 251–258.
[38] J. Yin, Y.-W. Si, and Z. Gong, "Financial time series segmentation based on turning points," in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Jun. 2011, pp. 394–399.
[39] Y. Hu, C. Ji, M. Jing, Y. Ding, S. Kuai, and X. Li, "A continuous segmentation algorithm for streaming time series," in *Proc. Int. Conf. Collaborative Comput., Netw., Appl. Worksharing*, 2016, pp. 140–151.
[40] E. S. Gardner, "Exponential smoothing: The state of the art—Part II," *Int. J. Forecasting*, vol. 22, no. 4, pp. 637–666, Oct./Dec. 2006.
[41] Q. He, Zhidong, F. Zhuang, T. Shang, and Z. Shi, "Fast time series classification based on infrequent shapelets," in *Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 1. Dec. 2012, pp. 215–219.
[42] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 947–956.
[43] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.
[44] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining Knowl. Discovery*, vol. 28, no. 4, pp. 851–881, 2014.
[45] Z. Xing, J. Pei, P. S. Yu, and K. Wang, "Extracting interpretable features for early classification on time series," in *Proc. SIAM Int. Conf. Data Mining*, 2011, pp. 247–258.
[46] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: An expressive primitive for time series classification," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1154–1162.
[47] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 289–297.
[48] Z. Zhang, H. Zhang, Y. Wen, and X. Yuan, "Accelerating time series shapelets discovery with key points," in *Proc. Asia–Pacific Web Conf.*, 2016, pp. 330–342.
[49] L. Ye and E. Keogh, "Time series shapelets: A novel technique that allows accurate, interpretable and fast classification," *Data Mining Knowl. Discovery*, vol. 22, nos. 1–2, pp. 149–182, 2011.
[50] A. Bagnall, J. Lines, W. Vickers, and E. Keogh. (2016). *The UEA & UCR Time Series Classification Repository*. [Online]. Available: http://www.timeseriesclassification.com

YUPENG HU received the Ph.D. degree in software engineering from Shandong University. He is currently a Postdoctoral Fellow with the School of Computer Science and Technology, Shandong University. His research interests include information retrieval, data mining, and explainable AI.

PEIYUAN GUAN received the master's degree from Central South University, China, where he is currently pursuing the Ph.D. degree. His research interests include edge computing, Internet of Things, and game theory.



PENG ZHAN received the B.S. and M.S. degrees from Shandong University, China, where he is currently pursuing the Ph.D. degree. His main research interests include data mining, machine learning, and deep learning.

YIMING DING received the B.S. degree from Shandong University, China, where she is currently a Graduate Student. Her main research interests include data mining and machine learning.

XUEQING LI received the B.S., M.S., and Ph.D. degrees from Shandong University, China. He is currently a Professor with Shandong University. His main research interests include artificial intelligence, service computing, and computer vision.