An Improved Quantile-Point-Based Evolutionary Segmentation Representation Method of Financial Time Series

Lei Liu
School of Computer and Software Engineering, Xihua University, China
1163454848@qq.com

Zheng Pei
School of Computer and Software Engineering, Xihua University, China
pqyz@263.net

Peng Chen*
School of Computer and Software Engineering, Xihua University, China
chenpeng@mail.xhu.edu.cn

Zhisheng Gao
School of Computer and Software Engineering, Xihua University, China
46235604@qq.com

Zhihao Gan
School of Computer and Software Engineering, Xihua University, China
444624141@qq.com

Kang Feng
School of Computer and Software Engineering, Xihua University, China
1831147112@qq.com

Abstract: *Effective and concise feature representation is crucial for time series mining. However, traditional time series feature representation approaches are inadequate for Financial Time Series (FTS) due to FTS' complex, highly noisy, dynamic and non-linear characteristics. Thus, we proposed an improved linear segmentation method named MS-BU-GA in this work. The critical data points that can represent financial time series are added to the feature representation result. Specifically, firstly, we propose a division criterion based on the quantile segmentation points. On the basis of this criterion, we perform segmentation of the time series under the constraint of the maximum segment fitting error. Then, a bottom-up mechanism is adopted to merge the above segmentation results under the maximum segment fitting error. Next, we apply Genetic Algorithm (GA) to the merged results for further optimization, which reduced the overall segment representation fitting error and the integrated factor of segment representation error and number of segments. The experimental result shows that the MS-BU-GA has outperformed existing methods in segment number and representation error. The overall average representation error is decreased by 21.73% and the integrated factor of the number of segments and the segment representation error is reduced by 23.14%.*

Keywords: *Time series, feature representation, quantile segmentation points, linear segmentation, genetic algorithm.*

Received January 31, 2021; accepted January 9, 2022
https://doi.org/10.34028/iajit/19/6/4

1. Introduction

With the rapid development of Internet technology and network communication technology, we have come to the 4G/5G era. Nowadays, the term Big Data has become increasingly familiar to people. People use this term to describe and define the huge amount of data produced in the age of information explosion. As of 2012, the data volume has jumped from the TB level to the PB, EB, and ZB level. According to the research results of the International Data Corporation, the data volume in 2020 has increased by 44 times compared to the 0.49ZB of data generated globally in 2008. Among the massive data, one type of data is a set of random variables sorted by time. This situation is the result of observing a certain potential process at a given sampling rate at equal intervals. Accordingly, the use of time series data mining technology to find potential information and value in time series databases has attracted increasingly attention in the research. The research results have been successfully used in various fields, such as finance [7, 22] education, economics, and electronic information, Aerospace meteorology [18], and industry and medical [1, 3].

In the era of big data, how to efficiently represent a time series is necessary, especially in the face of current streaming data, which has strong real-time performance during the collection. We have difficulty operating on the original time series data. We need to retain the inherent important data features while reducing the time series dimension. Researchers have proposed many methods to reduce the dimensionality of the data, thereby reducing the cost of processing massive data. This type of method is called Piecewise Linear Representation (PLR), which is a simple and intuitive time series feature representation method, and has received the attention and adoption of many researchers. Such as the Discrete Fourier Transform (DFT) proposed by Pavlidis [13], discrete wavelet transform [3, 14, 15], and other methods can perform corresponding the feature representations to the original time series to a certain extent, and retain the primary data of the original time series feature. The segmented aggregation approximation method Piecewise Aggregate

Approximation (PAA) was proposed in [20]. Keogh's [9] experiment shows that PAA is simple, intuitive, fast, and has a huge breakthrough compared with the DFT proposed by Pavlidis [13]. Hu *et al*. [6] also regarded inflection points, step points, and other related timing points that can reflect the trend of time series as turning points. Zhu *et al*. [24] used feature points as segmentation points for time series feature representation. Zhan *et al*. [23] proposed a new online segmentation algorithm based on the important turning points. The adding of these turning points can not only indicate the beginning and end of key transactions in related financial activities, but also reflect the important data characteristics of the original time series. To obtain a comprehensive measurement, this work will not only express the error from the segment feature but also evaluate the integrated factor from the number of segments and the error.

In business intelligence applications, banking and financial industries, and industrial engineering, the data we face have the characteristics of continuous growth over time. The high-dimensional, dynamic, and uncertain characteristics of data hinder the development of time series data mining effectiveness. Therefore, the feature representation method has become an important means to improve the current time series data mining algorithm and the time series data preprocessing. We consider the complex, highly noisy, dynamic and non-linear characteristics of Financial Time Series (FTS). Reducing the dimensionality of the data while retaining its inherent important trend characteristics and retaining some critical turning points is quite challenging. Finally, the financial time series processed by the representation method can be applied by the subsequent data mining method, and its rules and knowledge can be mined. Therefore, FTS feature representation is a critical data preprocessing tool and the basis for subsequent research on FTS data mining [5]. To solve these challenges, the linear representation method based on the median feature proposed in this work reflects the trend characteristics of FTS points. We combined the bottom-up algorithm [9] to merge the above preliminary segmentation results and reduce the segmentation number satisfying the user-defined segmentation error. Moreover, we used the biological evolvability of the Genetic Algorithm (GA) [11] to select the segmentation results obtained from the previous merging operation as the current population to obtain better segmentation results. Furthermore, in the selection operator of the GA algorithm, we consider some critical data points of the FTS to retain some practical turning points in the final feature representation results, such as Turning-Points Important (TBP) and Turning-Points Based (TIP) in the paper [22]. Thus, making the representation results more realistic and laying a good foundation for subsequent data mining work.

We summarize our contributions as follows:

1. We propose a division criterion based on the quantile segmentation points. Based on this segmentation criterion, we can efficiently segment FTS. Moreover, it can reflect a dividing point of the current trading volume and indicate the critical time points in the related financial trading activities to provide users accurate guidance at the critical time.
2. We use GA to optimize the segmentation representation of FTS. In the selection operator of the GA algorithm, essential data points that can represent FTS are added to the feature representation result. Thus, it makes the final feature representation results to be more realistic.

The remainder of this paper is organized as follows: Section 2 describes the related works. Section 3 presents the problem description. Section 4 provides the details of our proposed algorithm. Section 5 presents the detailed comparative experiments. Section 6 draws the conclusion.

2. Related Work

The related research of time series data includes many types of research, such as time series feature representation [2], time series similarity search [4], time series classification, time series clustering, time series forecasting, and time series visualization [5, 19]. With such a huge database, researchers continue to propose new methods to reduce the dimensionality of the data, thereby reducing the cost of processing massive data. The approximate representation of the time series is a solution. Many scholars have proposed the method of the large segment representation PLR. The PLR methods can be divided into three categories according to different segmentation strategies as follows:

a) Piecewise Linear Representation based on Top-Down strategy (PLR-TD). For example, Keogh *et al*. [9] proposed the PLR-TP method based on a top-down linear piecewise strategy. For the original (unsegmented) sequence, this method will stand in the "global" perspective, introduce a segmenting point to the current time series at a time, and repeat this process until the current segmented linear representation result has been satisfied up to the pre-specified RS condition. Later, Ji *et al*. [8] combined the self-item-down segmentation strategy with finding Important Data Points (IDPs) in the time series to reduce the segmentation error while reducing the error of a single important point, which improves representation accuracy.
b) Piecewise Linear Representation based on Sliding Window Strategy (PLR-SW). PLR-SW uses SW to initialize the first data point of the original time series entering SW as the starting point of the linear segment and then searches for the current linear segment according to the pre-specified RS conditions during the sequential scan of the time series. This

process is repeated at the original sequence's termination point until the original sequence's piecewise linear representation is completed. The PLR method based on the Sliding Window (SW) strategy lacks time series feature representation from a global perspective. Liu *et al.* [10] proposed the Feasible Space Window (FSW) and Stepwise Feasible Space Window (SFSW) methods, and they have greatly improved the calculation efficiency of segmentation. Yin *et al.* [21] defined the local maximum and local minimum points as turning points.

GA is one of the earliest population-based random algorithms proposed in history [11]. It has been widely used. Sosiawan *et al.* [17] applied it in the hidden Markov model to obtain the best parameters of the model so that the model can obtain higher accuracy. Sheta *et al.* [16] uses GA to obtain the global optimal solution of the time series forecasting model, making the model more practical. Nayak used the global search ability of GA to get a better performance, which has been incorporated with the better generalization ability of a second order neural network [12]. Hence, how to make its solution more reliable is a problem we must pay attention to. What makes this algorithm reliable and able to estimate the global optimum for a given problem is the process of maintaining the best solutions in each generation and using them to improve other solutions. The GA used in our methods will be described in section 4. However, the previous methods cannot effectively obtain essential data points and some more realistic turning points of FTS. Moreover, the methods mentioned in the Relative work chapter are based on the traditional segmentation strategy. Furthermore, some transaction keys are not retained for the FTS, and the optimization strategy is not used in the final segment representation results. So, we propose a division criterion based on the quantile segmentation points and use the GA to optimize the segmentation representation of FTS.

3. Problem Description

To better describe the problem of FTS feature representation, we first present a set of conceptual definitions and quantitative measurement.

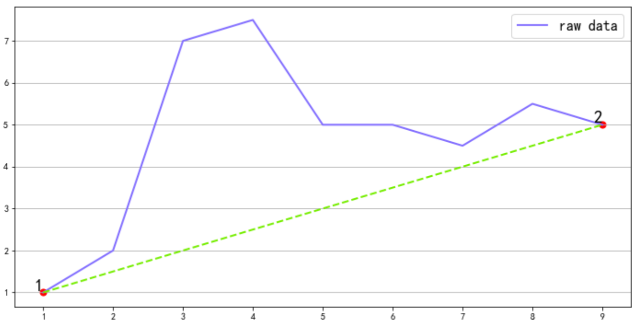
- **Definition 1**: Time series TS is a collection of data collected in chronological order, and it can be stated in the following form: for a given time series $T = (vt_1, vt_2, \dots, vt_i, vt_{i+1}, \dots, vt_n)$ of length n, the goal is to divide T into a sequence of segments $S_1S_2 \dots S_k (1 \le k \le n-1)$ and represent each segment with a straight line. Variable vt_i represents the time series data point at the time i in *TS*, which is referred to as the time series point. Variable n is generally a certain constant that represents the number of time series points in *TS*. In streaming time series data, n is a positive integer that can indefinitely increase.

In the FTS, the data characteristics of TS are reflected by all the data points in it. However, the impact of each data point on the overall data trend is not the same. When performing Linear Segmentation (LS) of TS, we need to pay special attention to the time series points in the *TS* that can reflect the trend of data changes not only to reduce the dimensionality of the original time series data but also to retain the basic data characteristics of the original data to the greatest extent. According to the analysis of FTS by Yin *et al.* [21] the local maximum and minimum points in the time series can be defined as turning points because these local extreme points indicate the beginning and ending moments of key transactions in related financial activities. These key transaction moments will guide users to choose the corresponding best transaction period. The turning point of time series also includes the diquantile segmentation points, which can reflect a dividing point of the current trading volume, and indicate the key time points in the related financial trading activities, to provide users accurate guidance at the key time. Next, we provide the basic definition of the data point of the time series.

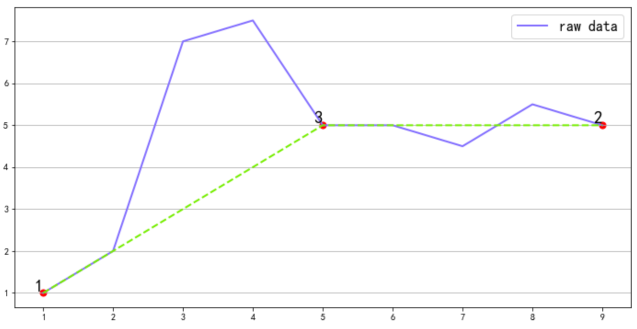
- **Definition 2**: The time series sorted by *TS* are marked as *TSO*=$\{vt_1, vt_2, \dots, vt_j, vt_{j+1}, \dots, vt_n\}$, where $1 < j < n$, *TSO* is obtained after the *TS* is sorted from small to large by default. Our diquantile segmentation points are recorded as *MSP* (median split point) to facilitate the subsequent presentation. When n is an odd number, the diquantile segmentation point is defined as $MSP_2 = vt_{\frac{n+1}{2}}$. When n is an even number, $MSP_2 = vt_{\lfloor\frac{n+1}{2}\rfloor}$, where it is expressed as rounding down to correspond to the time series points in the original time series. The original data points will not be lost in the result of our feature representation. We can effectively identify *MSP* from *TS* on the basis of the above definition.

The identification process of the above median segment points is shown in Figure 1. Figure 1-a) shows the initial segmentation result. According to the above definition 2, a quantile point is selected between segmentation points 1 and 2, as shown in Figure 1-b). Then, continue to select the next binary point 3, as shown in Figure 1-c). The result is shown in Figure 1-d). In completing the above identification operations, we consider the stream data characteristics of *TS*. Thus we need to establish a corresponding SW to perform LS operation on TS flowing into Sliding Window (SW). Given that the above operations need to be processed according to the corresponding data standards, and we need to determine the segmentation metric in the segment representation, this work uses the Maximum Vertical Distance (MVD) between the actual data point and the best-fit straight line as the metric standard. In comparison with Linear

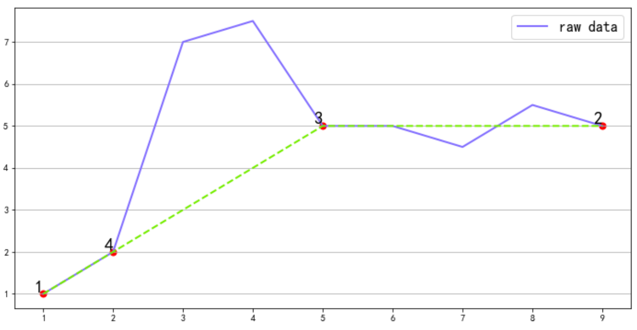
Regression (LR), Linear Interpolation (LI) is more efficient in processing data. Therefore, the subsequent segmentation operations in this work and related comparison experiments will use LI to complete the corresponding segmentation linear representation operations.



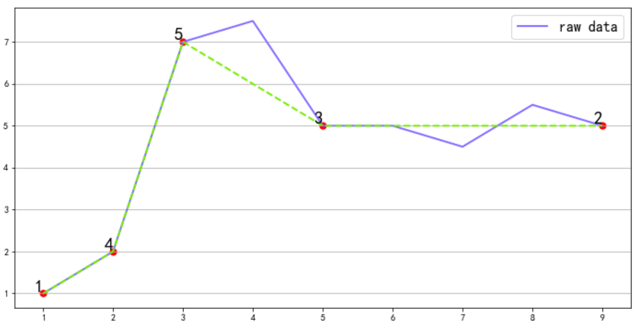
a) The first step of segment operation.



b) The second step of segment operation.



c) The third step of segment operation.



d) The fifth step of segment operation.

Figure 1. Process of identifying median segment points.

- **Definition 3**: Single Point Error (SPE) refers to the MVD between any time sequence point in TS and the fitted straight line. We suppose that the new time series $TS' = \{vt_1', vt_2', \dots, vt_i', vt_{i+1}', \dots, vt_n'\}$. After completing the segmentation operation on the time series TS, the SPE at time t_i in TS ep_i can be calculated as $ep_i = |v_i' - v_i|$.
- **Definition 4**: Maximum Error of Segmentation (MES) refers to the threshold value that the segmentation error of any segment obtained in *TS*. The MES cannot be exceed when LS is performed on the time series *TS*. After LS, the new segmented sequence can be expressed a $TS' = \{seg_1, seg_2, \dots, seg_m\}$, and seg_i is a segmented feature representation result of the time series. According to the definition of SPE in Definition 3, we can calculate the segmentation error of seg_i as $errorSeg_i = \sum_{n=p}^{q} ep_n$. Thus the *MES* in the time series *TS* can be expressed as $MES = \max_{i=1\dots m} errorSeg_i$.

The definitions of SPE, Segment Error (ES), and MES play important roles in the PLR and are the evaluation indicators for the subsequent comparison experiment of PLR. The feature representation method proposed in this work is divided into three stages on the basis of the above definition. In the first stage, we use diquantile segmentation point as the segmentation point to perform segmentation of TS and use the MES to constrain the segmentation point selection. If the ES of the current segment exceeds the pre-given MES, the diquantile segmentation points are selected again as the segmentation point on the current segment. The above process is repeated until all segments meet the requirements of MES. In the second stage, we combine the idea of bottom-up merging and continuously merge adjacent segments under the constraint of preset segment merging errors. If the segment of the new segment indicates that the error exceeds the given error, we begin from the next segment start to continue the merge operation until all segments have completed the merge operation. In stage three, introducing the GA idea can let us use the segmentation result obtained in stage two as the parent group, and generate new offspring through selection, crossover, and mutation operators. We select the optimal solution from these offspring as the final result. The detailed work and related algorithms of each stage will be provided below.

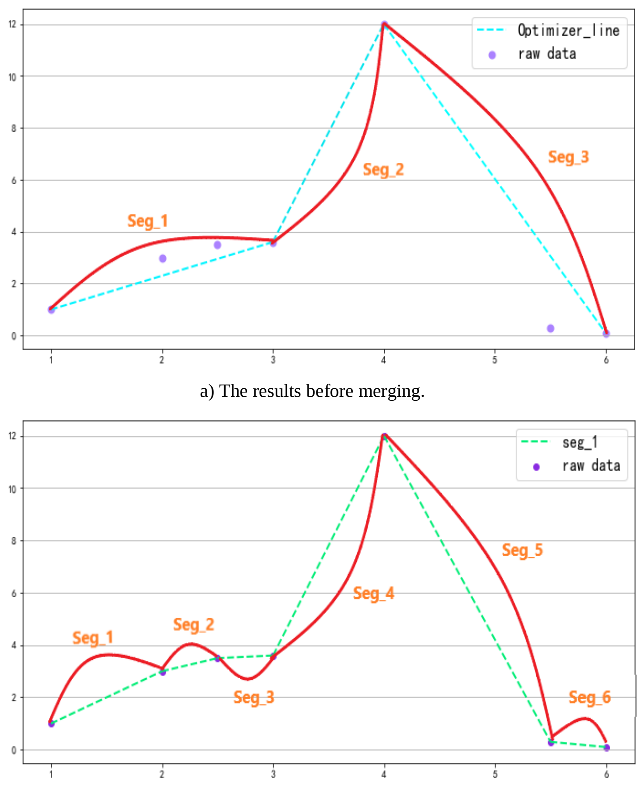
4. Algorithm Description

This work proposes a new time series segmentation algorithm on the basis of the segmentation criteria and related definitions proposed in section 3. We take the median in statistics as the dividing point, which is the representative value of the overall unit flag value determined by its position. The median is the number in the middle that can measure the central tendency of the data. When the median is applied to the FTS, it can dig out its hidden internal information and trends. Based on previous research, we selected SPE, ES, and MES evaluation indicators in the PLR. An integrated factor of the number of segments and the error of segment representation has been added. Choosing these indicators can help us describe the accuracy of the algorithm in PLR. The method also combines the traditional bottom-up strategy to achieve a concise representation result and, greatly improve the segmentation efficiency of the algorithm. Finally, we added the GA on the basis of the above-mentioned segmentation merge. We use the idea of a greedy method when merging adjacent segments. When the current segmentation error is less than MES, we continue to merge adjacent segments until the requirements of the MES are satisfied. Then, we merge from the next segment. The optimization strategy of the greedy method can easily fall into the local optimal solution and may not be able to obtain the global optimal solution from the global aspect; however, the GA can do that. We will further optimize the results of the previous step after segmentation through GA. After the selection, crossover, mutation, and combination operators are chosen, we have a low segmentation representation error based on the optimization of the previous step. A better performance on the integrated factor of segment number and error is obtained. Detailed algorithm description and experimental analysis will be provided in sections 5 and 6.

Algorithm 1: MS-BU-GA Algorithm phase 1: LS algorithm based on diquantile segmentation points

```
Input: time series ($TS = \{v_0, \dots, v_i, \dots v_j\}$) , MEP =δ, MES=ε
Output: segmenting points($s_0, s_2 \dots, s_i$).
1    Initial: start = $v_0$,end = $v_j$, segList = [$v_0$, $v_j$]
2    midnum = getMedian(TS, start, end)
     // obtain the median index of the current segment
3    segList.add(midnum)
     // adds the resulting median to the segmentation list
4    Ts_left = TS[start, midnum]
     // obtain the segment to the left of the median segment
     point
5    Ts_right = TS[midnum, end]
     // obtain the segment to the right of the median segment
     point
6    error_left = calculateError(Ts_left)
     // the left segmentation error is obtained
7    error_left = calculateError(Ts_right)
     // the right segmentation error is obtained
8    if (error_left >ε):
9        segList.add(getMMP(Ts_left, start, midnum, δ, ε))
10   end If
11   if (error_right >δ):
12       getMMP(Ts_right, midnum, end, δ, ε))
13   end If
14   return segList
```

The above description is the segmentation part. The time series TS is divided into several segments on the basis of the given MEP and MES. Each segment satisfies the constraint of MES. First, all the diquantile segmentation points (median) of the time series are searched by the second row of the above algorithm. In the current segList (segment point list), we can obtain the segments composed of the current segment point, the initial point, and the endpoint, as Ts_left and Ts_right according to the index of the segment point, as shown in line 4. In the fifth line of the algorithm, we can repeat the above operations on Ts_left and Ts_right through recursion until we find all the diquantile segmentation points that meet the conditions.



a) The results before merging.

b) The results after merging.

Figure 2. Process of piecewise merge.

In Figure 2, we can combine the adjacent segments according to the requirements of the overall representation error combined with the bottom-up strategy, to reduce the number of segments. For example, the SWAB method combined with the bottom-up strategy is better than the SW method in the segment representation results. Figure 2-a) demonstrates the segment representation result obtained by the above segmentation stage. Every two consecutive segment points form a segment with each other, which will produce too many numbers of segment points and fail in simplifying the original time series. Segments Seg_1, Seg_2, and Seg_3 in Figure 2-a) become Seg_1 in Figure 2-b) after merging. Seg_5 and Seg_6 become Seg_3 in Figure 2-b) after merging. Based on the segmentation representation result of phase 1, we carry out the merging operation of adjacent segments to ensure that the segment representation result more concise under the constraints of the MEM (maximum error of merge segment). In combination with the bottom-up strategy, we have simplified the 6-segment segmentation result from the first segmentation to the 3-segment segmentation result shown in Figure 2-b). We will describe the merge operation in phase 2.

Algorithm 2: MS-BU-GA Algorithm phase 2: Merging stag based on bottom-up strategy

```
    Input: segmenting points($s_0, ..., s_i, ... s_j$), TS, MEM=β,
    Output: segList ($v_0, ..., v_i, ... v_j$)
    1   Initial: i = 1, start = $s_0$, segList_merge=null
    2   segList_merge.add(start) //add segment starting point
    3   while i < segList.length: //run the merging operatio
    4       error_merge =
            mergeSeg_error(start,segList[i+1])
            // calculate the combined segmentation error
    5       if error_merge < β:
    6           end = TS[TS.index(end) + 1]
    7       else:
    8           seg_merge.add(end) //
    9           start=TS[i+1]
                // change the merge operation starting point
    10      end if
    11      i += 2
    12  end while
    13  return segList_merge
```

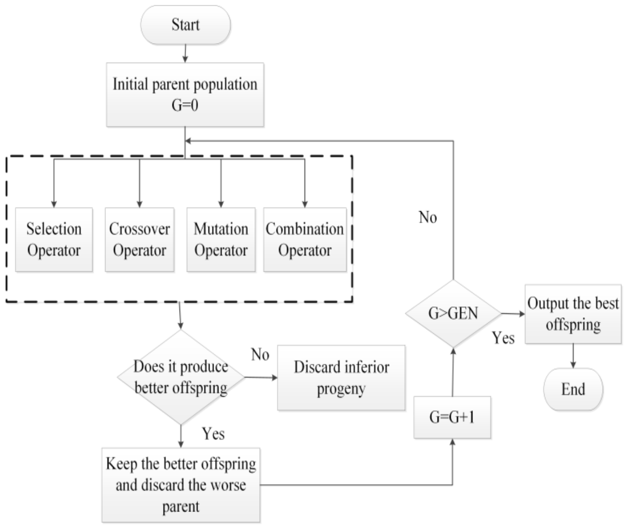
Phase 2 takes the segmentation point list after the segmentation operation in Table 1 as input. Under the constraints of the MEM specified by the user, we perform segmentation merge optimization operations. The third row calculates the segmentation error of the new segment after merging two adjacent segments. Then, we conduct error judgment. If the ES of the new segment is less than the Maximum Error of Merged segment (MEM), then continue to merge forward, otherwise, as shown in line 5. We continue to find segment points in segList until the ES of current segment is greater than the MEM, as shown in lines 7 and 8. The above process is repeated until all segments are merged. In comparison with the traditional method Sliding Window and Bottom-up (SWAB), the method starts to merge operation from adjacent segments. Nevertheless, the merge operation of SWAB starts from adjacent timing points. Therefore, this method can not only retain the segmentation results after the segmentation operation of phase 1 but also improve the feature representation efficiency of the overall algorithm.

Based on the segmentation results in phase 2, we use the evolvable nature of GA to optimize the segmentation results. The specific algorithm description will be provided in phase 3.

Algorithm 3: MS-BU-GA Algorithm phase 3: GA optimization stage

```
Input: segmenting points ($s_0, s_1 ..., s_i, ... s_j, ...$) , GEN = γ
// γ is the max iteration
Output: segList_GA ($v_0, v_1 ..., v_i, ... v_j, ...$)
1   geneticAlgorithm(segList, G):
2       num = getParentNum(segList.length)
        // obtain the number of parent generations
3       ParentList = getVector(segList,num)
        // obtain the parent solution vector
4       parent1 = random.randsample(ParentList,num))
        //one half of the parent solution vector is chosen at
        //random as parent 1
5       parent2 = ParentList.remove(parent1)
        //the remaining vector of parentList is chosen as
        //parent2
6       random = (selection,cross,mutation,combine)
        //randomly select one from the four operator
7       segList_GA = random(parent1,parent2)
8       if(segList.length == 1 or G > γ): // obtain the best
        //child generation
9           return segList
10      G == G + 1
11      geneticAlgorihtm(segList, G)
```

The input of the above-mentioned method is the segmented point list from the phase 2. First, the length of the currently segmented point list segList is used to determine the number of initial solution vectors of the initial population, as shown in the second row of Table 3. Then, we divide the above-mentioned initial parents into two equally for genetic operator operations as shown in the 4th and 5th lines. The genetic operators we choose include natural selection, crossover, mutation, and combination operators.



Start
Initial parent population G=0
Selection Operator
Crossover Operator
Mutation Operator
Combination Operator
No
G>GEN
Output the best offspring
Yes
End
Does it produce better offspring
No
Discard inferior progeny
Yes
G=G+1
Keep the better offspring and discard the worse parent

Figure 3. The flow chart of GA.

We select operators from the four operators based on random probability to directly perform genetic operations on the two parents to produce better offspring. The selection strategy is to select the offspring that meets the requirements according to the segmentation error standard. If the offspring produced is better than the parent, then the parent is discarded, otherwise, the current parent is retained for the next round of inheritance. The above operations are repeated until one offspring is left. Then, the current offspring is the optimal offspring that meets the requirements. The detailed flowchart of the algorithm is shown in Figure 3.

5. Experiment and Analysis

In this section, we will conduct a comprehensive experimental analysis of our methods, MS-BU, MS-BU-GA, and FSW methods, SFSW methods, and

OPLR-TP methods. In order to make more detailed comparison, MS-BU-GA phase 2 is named MS-BU (median segmentation based bottom-up). The FSW, SFSW, and OPLR-TP methods are online methods. The FSW and SFSW algorithms are improved methods in terms of the global evaluation of the entire time series of the SW algorithm, and they are faster than the SW method and more reliable. In the subsequent experiments, we consider three indicators, the number of segments, the representation error, and the product of their standardized results, as the experimental standards to compare other methods with the MS-BU-GA method to evaluate the LS efficiency.

5.1. Experimental Setting

In the experiment, the experimental data set we selected includes the historical data of the US S&P 500 index and the Shanghai and Shenzhen 300. In the subsequent comparative experimental analysis, SAP500 will be used to represent the closing price in the historical data of the US S&P 500, and the CSI300 is used to represent the closing price in the historical data of CSI 300. This initiative is carried out to facilitate the description of the data set. The detailed description is provided in Table 1. The setting of MES will affect the performance of the algorithm. For example, all algorithms will have n-1 segments when we give a small MES error, where n is the overall length of the current data. When MES is provided with a large value, all algorithms will have a segment. If we use the same fixed MES value for the different data sets for the subsequent experiments, it may not be possible for the LS to be reasonably evaluated and analyzed, considering the certain differences in the different data sets in our experiments. For example, the maximum error segmentation of two is acceptable for a time series with a value from 1 to 100. However, such an MES is not acceptable for a time series with a value ranging from 50 to 70. If the MES is set to 30% for a time series with values ranging from 50 to 70, then the maximum percentage segmentation error MEPS=(70-50)*30%=6 in our algorithm. Accordingly, we use the relative MES in our experiment which is the percentage of MES in the range of time series values, named the Maximum Error Percentage (MEPS) rather than absolute MES because the absolute MES is invariable. The error of time series representation in different data sets cannot be effectively calculated by using the same MES for different data sets.

Table 1. Data set description.

Name	Length	Description
SAP500	7603	Closing price in the historical data of the U.S. S&P 500 Index
CSI300	4515	Closing price in the historical data of CSI 300
SAPO	7603	Opening price in the historical data of the US S&P 500 Index
CSIO	4515	Opening price in the historical data of CSI 300

Given that the time series is sensitive to errors, we must select the appropriate MEPS for each different data set. This operation is necessary to ensure that the number of segments is within a reasonable range, thereby eliminating the influence of other unrelated parameters on the result of the linear representation. We illustrate the parameter MEM (maximum error of merge segment) in phase 2. We will set the numerical value of MEM to an integer multiple of MEPS to ensure the reasonableness of the subsequent experimental results of MS-BU, where N is set to a positive integer not less than one. Hence MEM is defined as $MEM=N\times MEPS$. The MES percentage in the comparison method OPLR-TP is defined as $MEM=N\times MEPS$, where n={1, 2, 3,…N} and N is a positive integer. We will conduct the corresponding comparative experiments on the MS-BU method and the three comparative methods on the above four data sets.

5.2. Segmentation Comparison Experiment Combined with Bottom-Up Merge Strategy

In this section, we will use the FSW, SFSW, and OPLR-TP methods as comparison methods in the data set shown in Table 1 to conduct comparative experiments. Then we calculate the Normalized Representation Errors (NREs) corresponding to different methods. We set MEPS as {3*10%, 3*20%, 3*30%, 3*40%, 3*50%} to objectively evaluate the representation errors (RE) of different methods. We set the MEM of the MS-BU method in the second stage of the merge operation as {3*10%, 3*20%, 3*30%, 3*40%, 3*50%} which constitutes five sets of PLR thresholds based on different MESP and MEPS and MEM{(10%,3*10%,8*10%),(20%,3*20%,8*20%),(30%,3*30%,8*30%),(40%,3*40%,8*40%),(50%,3*50%, 8*50%)}. Then, the representation error comparison experiments based on the threshold conditions set above are carried out on the data set shown in Table 1.

5.3. Representation Error and Segment Number Comparison Experiment

On the basis of the data set in Table 1, we conduct comparative experiments based on the experimental parameters provided in section 5.2. In the comparative experiment, we use RE to measure the representation accuracy of different representation methods. In the comparative experiment, we use RE to measure the representation accuracy of different representation methods. We set the NARE of FSW method as one in condition on MESP=10% to have clear results. We standardize the representation errors on the basis of the five groups of different thresholds, and obtain the NRE of each method in the current data set by using FSW, SFSW, and OPLR-TP as the comparison methods. The error comparison results are shown in Table 2.

Table 2. Representation error based on different thresholds.

SM	MESP	MEPS	MEM	ARE	NARE
FSW	10%			247.1067	1
SFSW	10%			198.3730	0.7798
OPLR-TP	10%	3×10%		231.0596	0.9083
MS-BU	10%		8×10%	209.1565	0.8222
FSW	20%			483.7300	1.9897
SFSW	20%			442.4271	1.7392
OPLR-TP	20%	3×20%		362.7395	1.4260
MS-BU	20%		8×20%	283.8535	1.1159
FSW	30%			850.5825	3.4946
SFSW	30%			677.4146	2.6630
OPLR-TP	30%	3×30%		421.5392	1.6571
MS-BU	30%		8×30%	340.1663	1.3372
FSW	40%			1024.3371	4.3726
SFSW	40%			858.8847	3.3764
OPLR-TP	40%	3×40%		508.1452	1.9976
MS-BU	40%		8×40%	391.7257	1.5399
FSW	50%			1289.6974	5.2916
SFSW	50%			1071.4184	4.2119
OPLR-TP	50%	3×50%		544.2596	2.1395
MS-BU	50%		8×50%	420.3861	1.7012

The ARE column in Table 1 is the average representation error of the above-mentioned four methods on the four different data sets. We set the NARE of the FSW method as one in condition on MESP=10% and standardize the ARE of the remaining methods to intuitively reflect the changes in the ARE under different threshold limits. According to the results in Table 1, the NARE of the MS-BU method slightly increases with the increase in MESP under different MEM constraints. The comparison results of the above-mentioned four methods under different MESP are shown in Figure 4. The MS-BU method has the smallest NARE under different MESP conditions. When MESP=50%, the NARE gap with other methods is the largest. In comparison with the FSW and SFSW methods, the MS-BU method uses MEPS to constrain the segmentation in the linear segment representation. In addition, the bottom-up merge operation is added on the basis of the segmentation result, and the predetermined MEM is used. The initial segmentation results are optimized. The number of segments is reduced and better accuracy can be maintained.

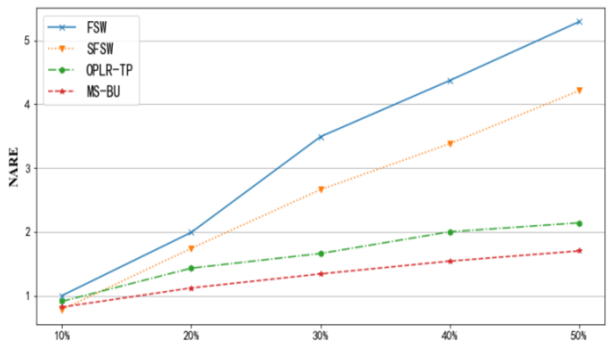


Figure 4. ARE under different segmentation error conditions.

5.4. Comparative Experiment Based On GA Optimization Stage

In this part, we will use the GA to further optimize our results on the basis of the above-mentioned segmented representation results that combine with the bottom-up merging strategy. A detailed description of the algorithm is provided in phase 3. In section 4, the MS-BU method uses the greedy strategy. We added a GA operation that can search for the global optimal solution in the overall segmentation result to ensure that the MS-BU method achieves better performance in segmented representation. In particular, we introduced the important perception point based on the turning point [8] as the segmentation point and added it to the optimization goal in the mutation operation of the genetic operator in GA. When adding such points, the reduction percentage in the ES is greater than the increase in the number of segments. The crossover and selection operators in the genetic operator can ensure that the number of segments selected after the parent crossover operation is not higher than the parent. The overall segmentation error of the offspring is better than that of the parent. The MS-BU-GA method can do better in the overall performance of segment representation error and the number of segments. Based on this optimization goal, the results of our detailed comparison experiments on the different data sets illustrated in Table 1 are showed in Figure 5.

In the results shown in Figure 5, panel (e) is based on the average representation error of different methods on various data sets, and panel (f) is the product of the standardized segment number and error. The NRE results of different methods on four different data sets are shown in Figures 5-a) to 5-d). With the continuous increase in the maximum error percentage, the representation errors of the above four methods will increase. The changes between the FSW and the SFSW methods are more obvious. When the MESP is =30% on the SAP500 closing price data set, the representation error of the SFSW and FSW methods has significantly exceeded that of Online PLR based on Turning Points (OPLR-TP) and MS-BU. When the MESP is=50% in Figures 5-c) and 5-d), it means that the error is better than OPLR-TP and MS-BU. When the MESP is=10%, the representation error of MS-BU-GA is better than those of OPLR-TP and MS- BU on the CSI300 closing price and CSI300 opening price data sets as shown in Figures 5-c) and 5-d). Under the constraint of small error, the two methods of FSW and SFSW have better segmentation performance; however, the representation error increases more than the other two methods with the increases in the maximum error percentage. Therefore, the MS-BU, MS-BU-GA, and OPLR-TP methods have good adaptability to errors. Meanwhile, the MS-BU-GA method shows the smallest increase in error under the relatively relaxed maximum error percentage limit, which is better than the other methods.

To evaluate the overall performance of the above-mentioned four methods on different datasets and intuitively reflect the overall situation of the four methods, we summarize the representation errors of the above four methods in the last four data sets according to the different threshold conditions. Then, we obtain the average representation errors on different data sets. We set the representation error of the FSW method in different data sets to one, and standardize other methods to obtain the results shown in Table 3. Moreover, we take the same operation on the segment numbers, and the results are shown in Table 4.

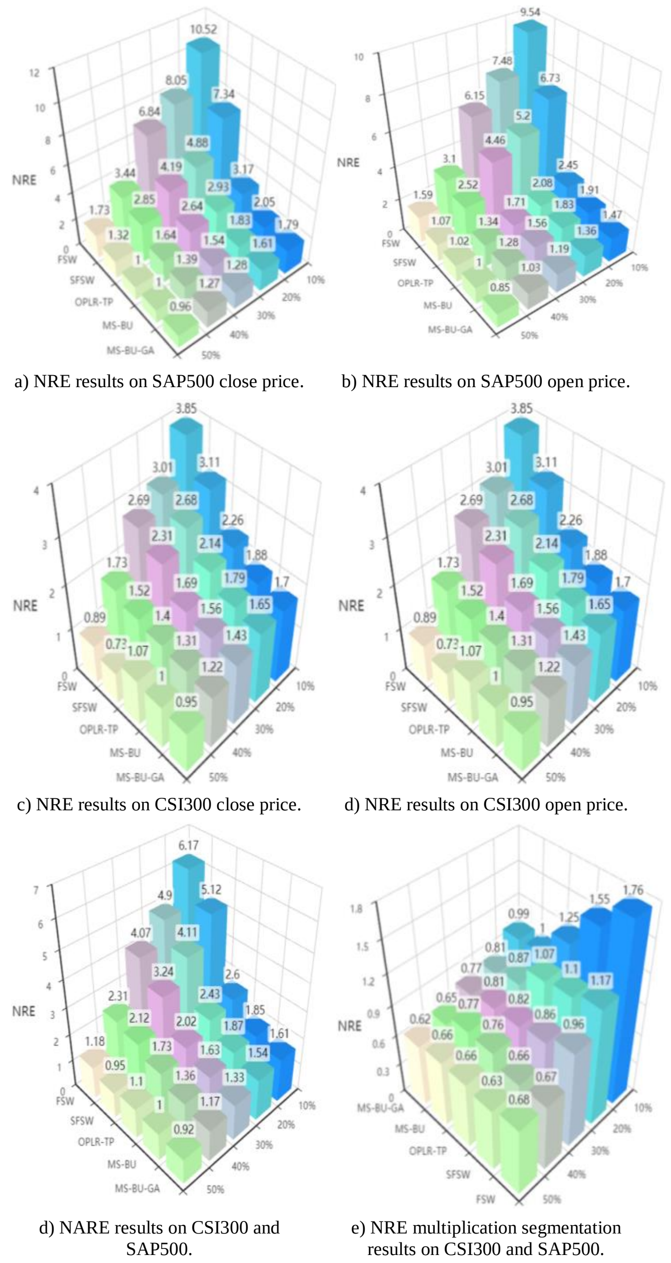

a) NRE results on SAP500 close price. b) NRE results on SAP500 open price.
c) NRE results on CSI300 close price. d) NRE results on CSI300 open price.
d) NARE results on CSI300 and SAP500. e) NRE multiplication segmentation results on CSI300 and SAP500.

Figure 5. Comparison of NRE and comprehensive performance on different data sets.

Table 3. Average normalized representation error of different methods.

SM	FSW	SFSW	OPLR-TP	MS-BU	MS-BU-GA
SAP500	1	0.67	0.37	0.26	0.19
CSI300	1	0.85	0.74	0.63	0.46
SAPO	1	0.72	0.30	0.27	0.27
CSIO	1	0.85	0.70	0.62	0.59
Average	1	0.77	0.53	0.44	0.38

Table 4. Average normalized number of segments of different methods.

SM	FSW	SFSW	OPLR-TP	MS-BU	MS-BU-GA
SAP500	1	1.45	2.35	2.91	3.01
CSI300	1	1.45	1.24	1.29	1.39
SAPO	1	1.50	2.50	2.59	2.67
CSIO	1	1.35	1.10	1.13	1.22
Average	1	1.44	1.80	1.98	2.07

According to the results in Tables 3 and 4, the MS-BU-GA method has the lowest number of segments, and the average representation error is reduced by 62 percent compared with the FSW. Meanwhile, the average normalized number of segments is increased by 107 percent. When performing the crossover operation, some important turning and perception points are added to the current parent to improve the accuracy of the representation. In terms of the overall number of segments, the FSW method has the lowest average number of segments, while the MS-BU-GA and OPLR-TP methods have more segments under the condition of the same segmentation constraints. By contrast, the MS-BU-GA method has the largest number of segments. In comparison with the MS-BU method, the number of segments is increased by 4.55%, which is less than the error reduction percentage of 13.63%, especially on the CSI300 data set. The overall average representation error combined with the GA optimization is 26.98% lower than the bottom-up merge result. However, the number of segments only increases by 7.75%. The GA can optimize the overall performance of the MS-BU method in the number of segments and the segment representation error. MS-BU-GA is 21.73% higher than the FSW in terms of the product result of the overall standardization error and the overall standardization segment number. A great improvement can be observed in the overall indicator of the error during the production as well as the error and the number of segments.

6. Conclusions

In this work, we first provide a basic segmentation criterion based on the diquantile point in the FTS. On the basis of this segmentation criterion, we propose an improved quantile-point-based evolutionary segmentation online PLR method MS-BU-GA based on a bottom-up mechanism. Experiments show that this method can perform efficiently PLR of FTS and make the result have higher representation accuracy. We conducted the corresponding comparative experiments based on a large number of different data sets under the condition of different representation errors that prove the advantages of diverse algorithms in the data set. The experimental results show that the FSW and SFSW methods always produce fewer segments and higher representation efficiency when given the same constraint MESP. By contrast, MS-BU-GA has higher representation accuracy when it has more segments while retaining the special points of the FTS such as the diquantile points. The OPLR-TP and MS-BU-GA methods have better stability and good robustness when given a relatively loose error. The MS-BU-GA method has better performance in the two standards of segment representation error and segment number. In future work, we will consider tertiles or quartiles as split points, not just diquantile. Moreover, we will also consider extending the segmentation method to other types of time series. Furthermore, we will pay more attention to the calculation time of the algorithm in the future. Finally, we will do some data mining work based on the time series feature representation results, such as classification and prediction.

Acknowledgments

This work has been partially supported by Xihua University Graduate Innovation Fund Research Project Grant YCJJ2021026. Science and Technology Program of Sichuan Province under Grant 2020JDRC0067 and 2020YFG0326. China Scholarship Council. Talent Program of Xihua University under Grant Z202047.

References

[1] Aldabbas H., Albashish D., Khatatneh K., and Amin R., "An Architecture of Iot-Aware Healthcare Smart System By Leveraging Machine Learning," *The International Arab Journal of Information Technology*, vol. 19, no. 2, pp. 160-172, 2022.

[2] Bernardini A. and Sarti A., "Canonical Piecewise-Linear Representation of Curves in the Wave Digital Domain," *in Proceedings of 25th European Signal Processing Conference*, Kos, pp. 1125-1129, 2017.

[3] Buhler J. and Tompa M., "Finding Motifs Using Random Projections," *Journal of Computational Biology*, vol. 9, no. 2, pp. 225-242, 2002.

[4] Chang M., Lou Y., and Qiu L., "An Approach for Time Series Similarity Search Based on Lucene," *in Proceedings of 4th International Conference on Cloud Computing and Intelligence Systems*, Beijing, pp. 210-214, 2016.

[5] Esling P. and Agon C., "Time-Series Data Mining," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1-34, 2012.

[6] Hu Y., Ji C., Jing M., Ding Y., Kuai S., and X Li., "A Continuous Segmentation Algorithm For Streaming Time Series," *in Proceedings of International Conference on Collaborative Computing: Networking, Applications and Worksharing Springer*, Beijing, pp. 140-151, 2016.

[7] Idrees S., Alam M., and Agarwal P., "A Prediction Approach for Stock Market Volatility Based on Time Series Data," *IEEE Access*, vol. 7, pp. 17287-17298, 2019.

[8] Ji C., Liu S., Yang C., Wu L., Pan L., Meng X., "A Piecewise Linear Representation Method Based on Importance Data Points for Time Series Data," *in Proceedings of IEEE 20th International Conference on Computer Supported Cooperative Work in Design*, Nanchang, pp. 111-116, 2016.

[9] Keogh E., Chu S., Hart D., Pazzani M., "An Online Algorithm for Segmenting Time Series," *in Proceedings IEEE International Conference on Data Mining*, San Josepp, pp. 289-296, 2001.

[10] Liu X., Lin Z., and Wang H., "Novel Online Methods for Time Series Segmentation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 12, pp. 1616-1626, 2008.

[11] Mirjalili S., "Evolutionary Algorithms and Neural Networks," *in Studies in Computational Intelligence*, *Springer*, vol. 780, 2019.

[12] Nayak S., Misra B., and Behera H., "On Developing And Performance Evaluation of Adaptive Second Order Neural Network with Ga-Based Training (Asonn-Ga) for Financial Time Series Prediction," *in Advancements in Applied Metaheuristic Computing*, *IGI global*, pp. 231-263, 2018.

[13] Pavlidis T., "Waveform Segmentation through Functional Approximation," *IEEE Transactions on Computers*, vol. 100, no. 7, pp. 689-697, 1973.

[14] Reinert G., Schbath S., and Waterman M., "Probabilistic and Statistical Properties of Words: An Overview," *Journal of Computational Biology*, vol. 7, no. 1-2, pp. 1-46, 2000.

[15] Staden R., "Methods for Discovering Novel Motifs in Nucleic Acid Sequences," *Bioinformatics*, vol. 5, no. 4, pp. 293-298, 1989.

[16] Sheta A. and De Jong K., "Time-Series Forecasting Using GA-Tuned Radial Basis Functions," *Information Sciences*, vol. 133, no. 3, pp. 221-228, 2001.

[17] Sosiawan A., Nooraeni R., and Sari L., "Implementsation of using HMM-GA in Time Series Data," *Procedia Computer Science*, vol. 179, pp. 713-720, 2021.

[18] Temme C., Ebinghaus R., Einax J., Steffen A., Schroeder W., "Time Series Analysis of Long-Term Data Sets of Atmospheric Mercury Concentrations," *Analytical and Bioanalytical Chemistry*, vol. 380, no. 3, pp. 493-501, 2004.

[19] Xing Z., Pei J., and Keogh E., "A Brief Survey on Sequence Classification," *ACM Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 40-48, 2010.

[20] Yi B., and Faloutsos C., "Fast Time Sequence Indexing for Arbitrary LP norms," *in Proceedings of the 26th International Conference on Very Large Databases*, Cairo, pp. 297-306, 2000.

[21] Yin J., Si Y., and Gong Z., "Financial Time Series Segmentation Based on Turning Points," *in Proceedings International Conference on System Science and Engineering IEEE*, Macau, pp. 394-399, 2011.

[22] Yu P. and Yan X., "Stock Price Prediction Based on Deep Neural Networks," *Neural Computing and Applications*, vol. 32, no. 6, pp. 1609-1628, 2020.

[23] Zhan P., Hu Y., Luo W., Xu Y., Zhang Q., and Li X., "Feature-Based Online Segmentation Algorithm for Streaming Time Series (Short Paper)," *in Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Shanghai, pp. 477-487, 2018.

[24] Zhu Y., Wu D., and Li S., "A Piecewise Linear Representation Method of Time Series Based on Feature Points," *in Proceedings of the International Conference on Knowledge-Based and Intelligent Information And Engineering Systems*, Vietri sul Mare, pp. 1066-1072, 2007.

Lei Liu received the B.S. degree in Information and Computing Science from Sichuan Agricultural University, Ya'an, Sichuan in 2019. He is currently pursuing the master's degree with Xihua University, Chengdu, China. His research interests include machine learning and financial time series analysis.

Zheng Pei received the M.S. and Ph.D. degrees from Southwest Jiaotong University, Chengdu, China, in 1999 and 2002, respectively. He is currently a Professor with the School of Science, Xihua University, Chengdu. He has nearly 100 research articles published in academic journals or conference. His research interests include rough set theory, fuzzy set theory, logical reasoning, and linguistic information processing.

Peng Chen IEEE member, CCF member, received B.E. degree in computer science and technology from University of Electronic Science and Technology of China, M.Sc. degree in computer software and theory from Peking University and Ph.D. in computer science and technology from Sichuan University. He is currently a full professor of School of Computer and Software Engineering, Xihua University. His research interests include machine learning, service computing and time series analysis.

Zhisheng Gao is an professor at the Xihua University. He received his Ph.D. degree in computer science from Sichuan University in 2012. He is the author of more than 50 journal papers. His current research interests include machine learning, image processing, and computer vision.

Zhihao Gan received the B.E. degree in Internet of Things Engineering from Xihua University, Chengdu, China in 2020. He is currently pursuing the master's degree with Xihua University, Chengdu, China. His research interests include time series analysis and cloud computing.

Kang Feng received the B.E. degree in Communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China in 2019. He is currently pursuing the master's degree with Xihua University, Chengdu, China. His research interests include financial time series forecasting and decision-making.