Received December 25, 2019, accepted February 4, 2020, date of publication February 27, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976494

Fast Piecewise Polynomial Fitting of Time-Series Data for Streaming Computing

JIANHUA GAO, WEIXING JI, LULU ZHANG, SENHAO SHAO, YIZHUO WANG, AND FENG SHI
School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Weixing Ji (jwx@bit.edu.cn)

ABSTRACT Streaming computing attracts intense attention because of the demand for massive data analyzing in real-time. Due to unbounded and continuous input, the volume of streaming data is so high that all the data cannot be permanently stored. Piecewise polynomial fitting is a popular data compression method that approximately represents the raw data stream with multiple polynomials. The polynomial coefficients corresponding to the best-fitting curve can be calculated by the method of least squares, which minimizes the sum of the squared residuals between observed and fitted values. However, built on several matrix calculations, the method of least squares always leads to high time complexity and is difficult to be applied to streaming computing. This paper puts forward a fast piecewise polynomial fitting for time-series data in streaming computing. The input data stream is dynamically segmented according to a given residual bound. Meanwhile, the data points in each segment are fitted using an improved polynomial fitting method, which has less time overhead than general polynomial fitting by reusing the intermediate calculation results. Experimental results on four time-series datasets show that our algorithm can achieve the highest speedup to the general piecewise polynomial fitting of 2.82x for periodically sampled time-series data and 1.85x for aperiodically sampled time-series data, without affecting the compression ratio and fitting accuracy. Moreover, the event-time latency comparison in a streaming environment indicates that the improved method can endure higher throughput than general piecewise polynomial fitting with the same latency.

INDEX TERMS Least squares, piecewise polynomial fitting, streaming computing, time-series data.

I. INTRODUCTION

Streaming computing is gaining intense attention because of the increasing demand for massive data analyzing in real-time. On-the-fly processing of large datasets in a reasonable time is required in streaming computing, which is extensively applied in network monitoring systems, sensor networks, aerospace systems, and meteorological monitoring.

A time-series data stream is an unbounded sequence of data points taken successively in time. With the widespread use of sensors, smart devices, and other data collection devices, large-scale time-series data become ubiquitous. Due to the limited buffer space and expensive data communication, caching all time-series data in the memory is completely unpractical. An approximate representation of raw data points not only shows the general outline and developing tendency of the data but also reduces the storage overhead and facilitates subsequent data analysis and visualization. Therefore, it becomes important to compress time-series data in advance and return the data on the fly, which calls for online algorithms that take the data point one by one and construct the compressed representation of the time series as the data is streaming [1], [2].

In recent years, researchers have shown an intensive research in approximate representation or compression of large-scale data. The most popular methods include Singular Value Decomposition (SVD) [3]–[7], Discrete Fourier Transform (DFT) [8]–[11], Discrete Wavelet Transform (DWT) [5], [12]–[15], Piecewise Aggregation Approximation (PAA) [16]–[18], Piecewise Linear Approximation (PLA) [1], [2], [19]–[23], Adaptive Piecewise Constant Approximation (APCA) [24]–[26], Piecewise Curve Fitting [27]–[38], and some other methods [39]. Piecewise curve fitting fits discrete data utilizing different fitting functions at different intervals. These fitting functions include polynomial, exponential, power, and rational functions. Polynomial curve fitting is a typical fitting method. It constructs a polynomial that has the best approximate representation globally to a set of data points. The most common algorithm for computing the polynomial coefficients corresponding to

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

the best approximate representation is the method of least squares, which minimizes the sum of the squared residuals between the observed and fitted values. The method of least squares, built on several matrix calculations, leads to high time complexity, which is difficult to be directly applied to streaming computing on account of real-time processing constraints.

The temporal order of time-series data makes its analysis distinct from other data. This paper puts forward a fast piecewise polynomial fitting of time-series data for streaming computing. The input data stream is dynamically segmented according to a given residual bound. Meanwhile, the data points in each segment are fitted by a polynomial. In the process of polynomial fitting, we employ different acceleration methods for periodically and aperiodically sampled time-series data based on their respective temporal features. As for periodically sampled time-series data, we notice that in the calculation of the coefficient vector of the fitted polynomial, these data segments, which comprise of the same number of data points, have the same intermediate calculation results. Hence the hash table is used to cache the results to reduce redundant calculation. As for aperiodically sampled time-series data, we notice that in the same segment, the adjacent polynomial fittings have some overlapped calculations, thereby the idea of the incremental calculation is adopted to accelerate the fitting process. To evaluate the performance of our improved method, we conduct a series of experiments on four time-series datasets. The speedup to the general piecewise polynomial fitting is up to 2.82x for periodically sampled data and 1.85x for aperiodically sampled data. Moreover, the event-time latency comparison of two methods shows that the higher throughput is available to the improved method instead of general piecewise polynomial fitting with the same latency.

The main contributions of this paper are as follows:

- A dynamic segmentation algorithm for the time-series data stream is proposed;
- Different acceleration methods are proposed for the piecewise polynomial fitting of periodically and aperiodically sampled time-series data.
- The comparison of speedup and latency of two methods, as well as the presentation of compression ratio and fitted results, are of reference significance.

The rest of the paper is organized as follows. We investigate several compression algorithms for time-series data in Section II. Section III discusses the general polynomial fitting method. An improved piecewise polynomial fitting method for the time-series data stream is proposed in Section IV. Section V shows our experimental methodology and results. Section VI makes a conclusion and summary.

II. RELATED WORK

Constructing good approximation to time-series data is a fundamental problem in data compression, statistics, and databases. There has been extensive work in recent years on this problem, and they can be categorized into the following methods.

SVD (Singular Value Decomposition) was firstly used for indexing images and other multimedia objects by Wu *et al.* [3] and Kanth *et al.* [4], proposed for time series indexing by Chan and Fu [5], and first implemented by Keogh *et al.* [16]. Besides, Benes and Kruis [6] applied SVD to the compression of results from finite element solvers. Guo *et al.* [7] proposed a fast SVD method to compress nuclear magnetic resonance echo data. SVD is good at dealing with global transformation, which at the same time leads to high complexity and is not suitable for streaming computing.

DFT (Discrete Fourier Transform) was the first technique suggested for compression of time series by Agrawal *et al.* [8]. They used the DFT to map time sequences to the frequency domain and then had thus mapped sequences to a lower-dimensionality space by using only the first few Fourier coefficients. Besides, Nair *et al.* [10] applied DFT to approximate the bilateral filter, Kithulgoda *et al.* [11] presented a method for incremental maintenance of the Fourier spectrum to changes in concept that take place in streaming computing, also the schemes for feature selection and synopsis generation that enable the coefficient array to be refreshed efficiently on a periodic basis. In DFT, each coefficient represents a sine wave that is added along the entire length of the query, so it is not possible to utilize DFT methods for queries with ''don't care'' subsections [9] or the more general weighted Euclidean distance [16], [40].

DWT (Discrete Wavelet Transform) was first utilized for fast nearest-neighbor search in medical image databases by Korn *et al.* [41], then Chan and Fu [5] applied it for the compression of time series. Besides, Jain *et al.* [12] and Rajan and Fred [13] used 2D-DWT to compress image, and Xu *et al.* [14] presented a DWT-based fast and high-efficient intraframe compression algorithm. In contrast to DFT, DWT is a local transformation. In other words, some of the wavelet coefficients in DWT represent local information of data being studied.

Keogh et al. first introduced the PAA (Piecewise Aggregation Approximation) algorithm in [16], which approximated the data by segmenting the sequences into equi-length sections and recoding the mean value of these sections. Then, these mean values can be indexed efficiently in a lower dimensionality space. This method may miss some important information and sometimes cause inaccurate results in time series mining, so Guo *et al.* [17] presented an improved PAA based on statistical features including a mean-based feature and variance-based feature. Besides, Fotso *et al.* [18] presented a heuristic for time series compression with PAA. Although PAA is simple to understand and to implement, it still has some disadvantages: (1) The size of section is a key factor; (2) Minimizing dimensionality by the mean value may miss some characteristic information.

The method of PLA (Piecewise Linear Approximation) first segments the time series and uses linear functions to fit each segment with certain error criterion. Gandhi *et al.* [1] presented an abstract framework for the online approximation of one-dimensional time-series data. Xie *et al.* [20] proposed two linear-time algorithms to construct error-bounded PLA for data stream based on the time domain. One generates a minimal number of line segments for the stream approximation, and the other is an alternative solution for the requirements of high efficiency and resource-constrained environment. Grützmacher *et al.* [22] proposed a novel PLA technique with a constant computational complexity as well as a constant memory complexity.

In contrary to PAA, the method of APCA (Adaptive Piecewise Constant Approximation) [24]–[26] segments the time series into a series of variable length segments, each of which is represented by a pair of the mean value and rightmost time-scale value of the segment. Because of the great storage overhead of APCA technique, Wang [26] proposed a new APCA-enhanced compression and query method, maintaining several times improvement of compression ratio compared with original APCA algorithm.

Piecewise curve fitting [27]–[38] segments the time-series data sequence into a series of variable-length segments, each segment is fitted by a non-linear function aiming to minimize the error between fitted value and observed value. These non-linear functions include polynomial, exponential, power and rational functions.

When the polynomial function is selected as a fitting function, the fitting process is called piecewise polynomial fitting [30], [33], [34]. This method is easy to understand and implement, and we can control the fitting error by adjusting the polynomial order. More importantly, it maintains a high compression ratio and fast indexing. However, high computing complexity in high-order polynomial fitting usually limits the scale of the problem solved. Therefore, this paper focuses on the acceleration of piecewise polynomial fitting so that it can be used in streaming computing.

III. POLYNOMIAL CURVE FITTING

Polynomial curve fitting is a fitting method in which the relationship between the independent variable and the dependent variable is modeled as a polynomial expression. Time-series data is a series of discrete data points $\langle(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\rangle$ collected over time, in which $\vec{t} = (t_1, t_2, \ldots, t_n)$ is a sequence of timestamps, and $\vec{y} = (y_1, y_2, \ldots, y_n)$ is a sequence of values. In this paper, we take the piecewise polynomial fitting between the time series and certain attribute values as example to illustrate our method. Let $\vec{t}$ be the independent variable and $\vec{y}$ be the dependent variable, the relationship between $\vec{t}$ and $\vec{y}$ can be modeled as $\vec{y} = \phi_k(\vec{t}) + \vec{r}$ utilizing polynomial fitting, in which ϕ_k is a polynomial map of order k ($k < n$). Its mathematical expression is given by (1).

$$\phi_k(t_i) = \theta_0 + \theta_1 t_i + \theta_2 t_i^2 + \cdots + \theta_{k-1} t_i^{k-1} + \theta_k t_i^k, \quad (1)$$

where $i = 1, 2, \ldots, n$. Besides, $\vec{r} = (r_1, r_2, \ldots, r_n)$ is the residual vector of ϕ_k on the time-series data, and it satisfies $r_i = |y_i - \phi_k(t_i)|$, which denotes the residual between fitted value $\phi(t_i)$ and observed value y_i.

Obviously, for a given group of data points, the fitted polynomials are not unique generally, but a curve with minimal error is desired. The most commonly used criterion for evaluating error is the l_p-error for $p = 1, 2, \infty$. The l_p-error of a function f for approximating the time-series data is $(\sum_{i=1}^{n} |f(t_i) - y_i|^p)^{1/p}$. In particular, the l_∞-error is $\max_i |f(t_i) - y_i|$ [2]. The process that obtains the optimal fitting polynomial by minimizing the square of l_2-error is called the method of least squares [31], [42].

Assume that

$$\boldsymbol{X} = \begin{bmatrix} 1 & t_1 & \cdots & t_1^k \\ 1 & t_2 & \cdots & t_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \cdots & t_n^k \end{bmatrix}, \quad \vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_k \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (2)$$

then

$$\min_{\vec{\theta}} \|\vec{r}\|_2^2 = \min_{\vec{\theta}} \|\boldsymbol{X}\vec{\theta} - \vec{y}\|^2. \quad (3)$$

There are many methods to solve (3), including normal equation, QR decomposition, Cholesky decomposition, and singular value decomposition. The method of the normal equation is utilized in this paper. Expanding $\|\boldsymbol{X}\vec{\theta} - \vec{y}\|^2$ as follows.

$$\begin{aligned} \|\boldsymbol{X}\vec{\theta} - \vec{y}\|^2 &= \|\boldsymbol{X}\vec{\theta} - \vec{y}\|^T \|\boldsymbol{X}\vec{\theta} - \vec{y}\| \\ &= \vec{\theta}^T \boldsymbol{X}^T \boldsymbol{X}\vec{\theta} - 2\vec{\theta}^T \boldsymbol{X}^T \vec{y} + \vec{y}^T \vec{y} \quad (4) \end{aligned}$$

From the necessary condition for solving the extreme of a function, we derive

$$\frac{\partial \|\boldsymbol{X}\vec{\theta} - \vec{y}\|^2}{\partial \vec{\theta}} = 2\boldsymbol{X}^T\boldsymbol{X}\vec{\theta} - 2\boldsymbol{X}^T\vec{y} = 0. \quad (5)$$

Further, we derive

$$\vec{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\vec{y}. \quad (6)$$

In (6), there are two matrix-matrix multiplications, one matrix inversion, and one matrix-vector multiplication, which have high time complexity and are difficult to satisfy the real-time processing constraints in streaming computing.

IV. POLYNOMIAL FITTING ACCELERATION

In streaming computing, time-series data may change over time. Therefore, constant polynomial fitting to the time-series data stream usually leads to a high time complexity and a large margin of fitting errors. A general solution to this problem is to divide the data sequence into multiple segments and fit each segment with a polynomial. However, as mentioned in the previous section, the general polynomial fitting has high time complexity and is difficult and inefficient to compress time-series data in real-time. Therefore, a fast piecewise polynomial fitting algorithm is proposed in this paper to reduce the time overhead of polynomial fitting in streaming computing.

First, we dynamically segment time-series data stream according to a given upper bound ε of the sqrt of the residual square (abbreviated as residual bound below) between fitted value and observed value. Let current time-series data sequence be $S = \langle (t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n) \rangle$, then the fitted polynomial ϕ_k of S satisfies following inequality.

$$(\phi_k(t_i) - y_i)^2 \leq \varepsilon^2, \quad i = 1, 2, \ldots, n. \tag{7}$$

Further, we derive

$$\sum_{i=1}^{n} (\phi_k(t_i) - y_i)^2 \leq n * \varepsilon^2. \tag{8}$$

Then, when a new data point (t_{n+1}, y_{n+1}) arrives, we just need to check whether the new time-series data sequence $S' = \langle (t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n), (t_{n+1}, y_{n+1}) \rangle$ satisfies following inequality.

$$\sum_{i=1}^{n+1} (\phi_k(t_i) - y_i)^2 \leq (n+1) * \varepsilon^2. \tag{9}$$

If it's true, then the fitted polynomial of S' is still ϕ_k. Otherwise, fitting a new k-order polynomial ϕ'_k for S' to make sure that (10) holds.

$$\sum_{i=1}^{n+1} (\phi'_k(t_i) - y_i)^2 \leq (n+1) * \varepsilon^2. \tag{10}$$

If the inequality holds, then the polynomial function ϕ'_k is the approximate expression of time-series data sequence S'. Otherwise, it means that no k-order polynomial can fit the time-series data sequence S' with the residual bound ε. At this time, data segmenting is required, and the data point (t_{n+1}, y_{n+1}) will be moved to the subsequent fitting as the first point of the new data segment.

The dynamic segmentation algorithm of time-series data in streaming computing using the given residual bound is presented in Algorithm 1, where the *temp_seg* and *temp_θ* temporarily store data values and coefficients of the current data segment respectively, *seg* and θ store segmented data values and fitted polynomial coefficients respectively, and the fitted polynomial coefficient vectors of all data segments are stored in S_θ.

Algorithm 1 Dynamic Segmentation Algorithm of Time-Series Data in Streaming Computing Using a Given Residual Bound

```
Input: time-series data stream $S = \langle \ldots, (t, y), \ldots \rangle$
Output: the set $S_\theta$ of coefficient vectors of the polynomials
1  Let $k$ be the order of polynomials.
2  Let $\varepsilon$ be the given residual bound.
3  $temp\_seg = \emptyset$; $S_\theta = \emptyset$;
4  while $input(t, y)$ do
5      $temp\_seg = temp\_seg \cup (t, y)$;
6      if length($temp\_seg$)== $k + 1$ then
7          $temp\_\theta$ = calPolynomial($temp\_seg, k$);
8          $\theta = temp\_\theta$;
9          $seg = temp\_seg$;
10     else if length($temp\_seg$) $> k + 1$ then
11         if calError($temp\_seg, temp\_\theta$) $>$ length($temp\_seg$) $* \varepsilon$ then
12             $temp\_\theta$ = calPolynomial($temp\_seg, k$);
13             if calError($temp\_seg, temp\_\theta$) $>$ length($temp\_seg$) $* \varepsilon$ then
14                 $S_\theta = S_\theta \cup \theta$;
15                 $temp\_seg = \{(t, y)\}$;
16             else
17                 $\theta = temp\_\theta$;
18                 $seg = temp\_seg$;
19             end
20         end
21     else
22         continue;
23     end
24 end
```

A. PERIODICALLY SAMPLED TIME-SERIES DATA FITTING

Periodically sampled time-series data are collected at equal time intervals. Let δ denote the time interval, and the number of data points in current segment is n, then the time sequence of current segment is $\vec{t_0} = (\delta, 2\delta, \ldots, (n-1)\delta, n\delta)$. In order to simplify the calculation, we abbreviate the time sequence $\vec{t_0}$ as $\vec{t} = (1, 2, \ldots, n-1, n)$. Then these segments, which comprise of the same number of data points, have the same simplified time sequences. Hence they have the same matrix X according to (2) and the same intermediate result $(X^TX)^{-1}X^T$ according to (6).

Based on the above observation, we propose an improved piecewise polynomial fitting method, which caches the intermediate results $(X^TX)^{-1}X^T$ of sequences with different segment lengths to accelerate the calculation of coefficient vectors. The intermediate results are stored in a hash table and indexed by the length of the corresponding data sequence. Whenever the calculation of (6) is required, we query the hash table to find the cached intermediate results. If the cache hits, we do not need to take a further calculation of $(X^TX)^{-1}X^T$. Instead, we just need to multiply the intermediate result stored in the hash table by the vector $\vec{y}$. If the cache misses, the new intermediate result will be calculated and stored in the hash table. For time-series data with the same segment length, the data size of the cached intermediate result is $(k+1) * n$, in which k represents the order of the fitted polynomial, and n represents the length of the fitted segment. Compared with the timely calculating of general piecewise polynomial fitting, the improved method requires a little more memory space but significantly reduces time overhead. With the increasing memory space of computers nowadays, it is reasonable to sacrifice space appropriately for time.

B. APERIODICALLY SAMPLED TIME-SERIES DATA FITTING

Aperiodically sampled time-series data have variable sampling intervals, so the segments with the same length no longer have the same time sequence. Then the method mentioned above is not feasible anymore. Therefore, we propose a different acceleration method for fitting aperiodically sampled time-series data.

Given a group of aperiodically sampled time-series data points $S = \langle (t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n) \rangle$, and the coefficient vector of the fitted polynomial with a given residual bound is $\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$, in which $X^T X$ and $X^T \vec{y}$ are given by (11).

$$X^T X = \begin{bmatrix} n & \sum_{i=1}^{n} t_i & \cdots & \sum_{i=1}^{n} t_i^k \\ \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 & \cdots & \sum_{i=1}^{n} t_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} t_i^k & \sum_{i=1}^{n} t_i^{k+1} & \cdots & \sum_{i=1}^{n} t_i^{2*k} \end{bmatrix},$$

$$X^T \vec{y} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} t_i y_i \\ \vdots \\ \sum_{i=1}^{n} t_i^k y_i \end{bmatrix}. \tag{11}$$

When a new data point (t_{n+1}, y_{n+1}) arrives, then the new time-series sequence $S' = S \cup (t_{n+1}, y_{n+1})$ requires to be refitted. According to the previous deduction, the coefficient vector of fitted polynomial of S' is given by $\vec{\theta'} = (X'^T X')^{-1} X'^T \vec{y'}$, in which $X'^T X'$ is given by (12).

$$X'^T X' = \begin{bmatrix} n+1 & \sum_{i=1}^{n+1} t_i & \cdots & \sum_{i=1}^{n+1} t_i^k \\ \sum_{i=1}^{n+1} t_i & \sum_{i=1}^{n+1} t_i^2 & \cdots & \sum_{i=1}^{n+1} t_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n+1} t_i^k & \sum_{i=1}^{n+1} t_i^{k+1} & \cdots & \sum_{i=1}^{n+1} t_i^{2*k} \end{bmatrix}$$

$$= X^T X + \begin{bmatrix} 1 & t_{n+1} & \cdots & t_{n+1}^k \\ t_{n+1} & t_{n+1}^2 & \cdots & t_{n+1}^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n+1}^k & t_{n+1}^{k+1} & \cdots & t_{n+1}^{2*k} \end{bmatrix}. \tag{12}$$

As we can see from (12), $X'^T X'$ can be expressed as the sum of two addends. The first addend $X^T X$ is the intermediate result of polynomial fitting of previous n data points, and the second addend is the difference caused by new data point (t_{n+1}, y_{n+1}). Similarly, $X'^T \vec{y'}$ can be expressed as (13), in which $X^T \vec{y}$ is the intermediate result of last polynomial fitting, and the second addend is the change caused by the arrival of the new data point.

$$X'^T \vec{y'} = \begin{bmatrix} \sum_{i=1}^{n+1} y_i \\ \sum_{i=1}^{n+1} t_i y_i \\ \vdots \\ \sum_{i=1}^{n+1} t_i^k y_i \end{bmatrix} = X^T \vec{y} + \begin{bmatrix} y_{n+1} \\ t_{n+1} y_{n+1} \\ \vdots \\ t_{n+1}^k y_{n+1} \end{bmatrix}. \tag{13}$$

According to the above analysis, we know that there are redundant calculations between adjacent polynomial fitting. To accelerate the computation of polynomial fitting for aperiodically sampled time-series data, we cache the intermediate results $X^T X$ and $X^T \vec{y}$ of last polynomial fitting. In the next polynomial fitting, the cached data is accessed to participate in the calculation of the new coefficient vector. In other words, the calculation of polynomial fitting for the current n data points is based on the intermediate result of the polynomial fitting of the first $n - 1$ data points. Similarly, the intermediate result of the polynomial fitting of current n data points will also be reused in the polynomial fitting of the first $n + 1$ data points. Therefore, each time a new data point is added, the intermediate calculation result of the polynomial fitting of the new data sequence can be calculated by applying the changes brought by the new data point to the previous intermediate results.

Besides, at any time, we only need to cache the intermediate results $X^T X$ and $X^T \vec{y}$ of a sequence, where the data sizes of $X^T X$ and $X^T \vec{y}$ are $(k + 1) \times (k + 1)$ and $(k + 1) \times 1$, respectively. Obviously, they vary with the order of polynomial instead of the number of data points in the current segment. As a result, the memory space can be reused, and less memory space is required.

V. EVALUATION

In this section, we present the experimental evaluation of our improved method to illustrate its performance on four time-series datasets. Detailed information of four datasets is presented in part A. Then, in part B, we present the speedup of improved method to general piecewise polynomial fitting and compression ratio of two methods with the different residual bounds and polynomial orders, as well as the accuracy comparison. In part C, we evaluate the event-time latency performance of two methods in a realtime computation system, namely Apache Storm 1.2.2 [43]. All experiments in this section are conducted on a computer with a CPU of Intel(R) Xeon(R) 3.40GHz and 16.00GB memory.

A. DATASET DESCRIPTION

The experiments are conducted on both periodically and aperiodically sampled time-series datasets.

1) GAS SENSOR ARRAY UNDER DYNAMIC GAS MIXTURES DATASET

[44], abbreviated as the gas dataset below, is sampled periodically. It contains the recordings of 16 chemical sensors exposed to two dynamic gas mixtures (CO and methane) at varying concentrations. For each mixture, signals were acquired continuously during 12 hours. In our experiments, 4,177,648 recordings altogether of the first chemical sensor exposed to methane are utilized.

2) INDIVIDUAL HOUSEHOLD ELECTRIC POWER CONSUMPTION DATASET

[45], abbreviated as the power consumption dataset below, is sampled periodically. The electric power consumption in one household with a one-minute sampling rate over almost four years is collected in this dataset. In our experiments, 2,075,259 recordings altogether of household global minute-averaged active power are utilized.

3) DAPHNET FREEZING OF GAIT (FoG) DATASET

[46], abbreviated as the FoG dataset, is sampled aperiodically. It contains the annotated readings of 3 acceleration sensors at the hip and leg of Parkinson's disease patients that experience FoG during walking tasks. In our experiments, 195,737 recordings altogether of the horizontal forward acceleration attribute are utilized.

4) GAS SENSOR ARRAY TEMPERATURE MODULATION DATASET

[47], [48] abbreviated as the temperature dataset, is sampled aperiodically. A chemical detection platform composed of 14 temperature-modulated metal oxide (MOX) gas sensors was exposed during three weeks to mixtures of carbon monoxide and humid synthetic air in a gas chamber. This dataset provides the acquired time series of the sensors and the measured values of CO concentration, humidity, and temperature inside the gas chamber. In our experiments, 295,653 recordings altogether of the R1(MOhm) attribute are utilized.

B. PERFORMANCE COMPARISON IN STATIC ENVIRONMENT

Before performing the experiments, the residual bound ε and polynomial order k are required to be given. To observe the overall approximation of polynomial fitting on the four datasets, we determine the value of ε according to the difference between maximum and minimum values of most data points in each dataset. The differences of the gas, power consumption, FoG, and temperature datasets are about 3403, 12, 1952, 120, respectively. Then, we set the error bound ε to be less than 10% of the difference for each dataset. In the real streaming computing scenario, however, the difference is unknown in advance, our suggestion is to determine the residual bound according to the prior knowledge of the specific attribute, or a variable residual bound is also recommended. As for the value of k, we change it from 2 to 10, and it will be explained later.

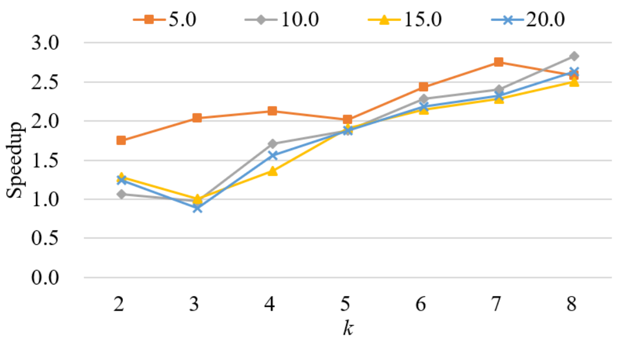


FIGURE 1. The speedup of the improved method to general piecewise polynomial fitting on the gas dataset with different residual bounds and polynomial orders. Four curves correspond to four different values of error bound ε.

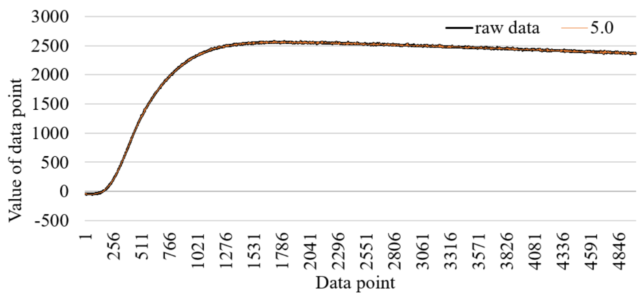


FIGURE 2. Comparison between raw data and fitted results on the gas dataset with $k = 2$ and $\varepsilon = 5.0$.

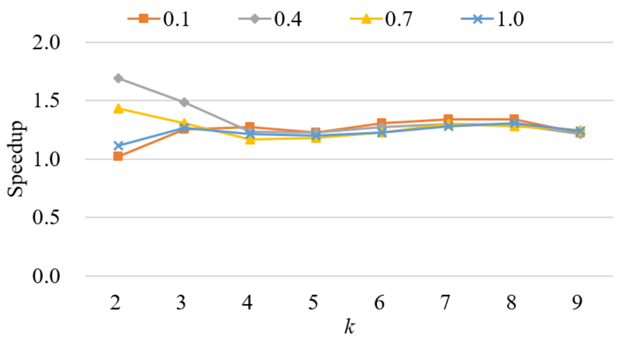


FIGURE 3. The speedup of the improved method to general piecewise polynomial fitting on the power consumption dataset with different residual bounds and polynomial orders. Four curves correspond to four different values of error bound ε.

1) PERIODICALLY SAMPLED TIME-SERIES DATA FITTING

The resulting speedup of the improved method to general piecewise polynomial fitting on the gas dataset is shown in Fig. 1. From the figure, we see the following trends. First, the speedup generally increases with increasing k.

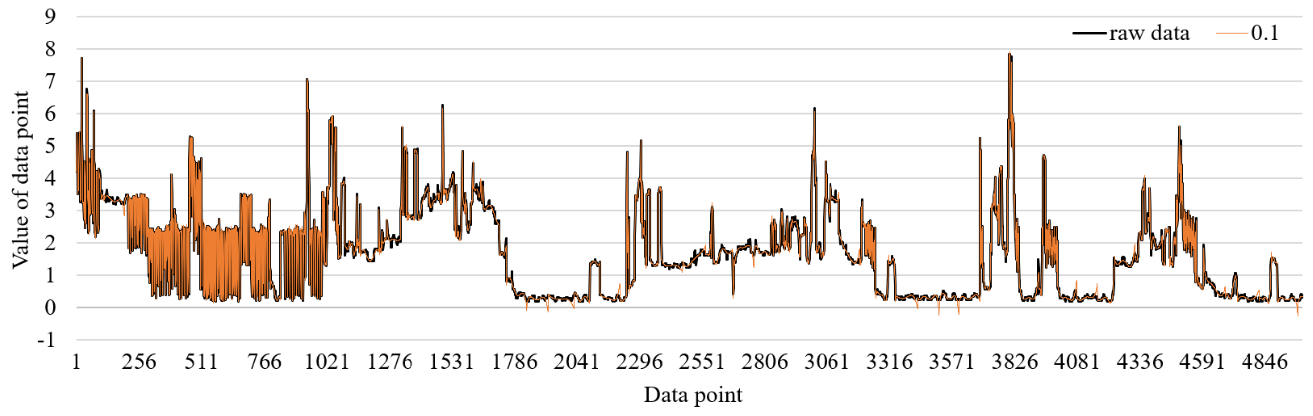


FIGURE 4. Comparison between raw data and fitted results on the power consumption dataset with $k = 5$ and $\varepsilon = 0.1$.

TABLE 1. The compression ratio of two methods on the gas dataset with different residual bounds and polynomial orders.

ε \ k	2	3	4	5	6	7	8
5.0	3.55	3.38	3.25	3.11	2.79	2.19	1.53
10.0	3878.97	888.86	94.44	18.85	6.56	3.03	1.64
15.0	5615.12	1106.37	119.17	22.16	7.32	3.25	1.70
20.0	6859.85	1193.61	136.93	24.56	7.90	3.43	1.74

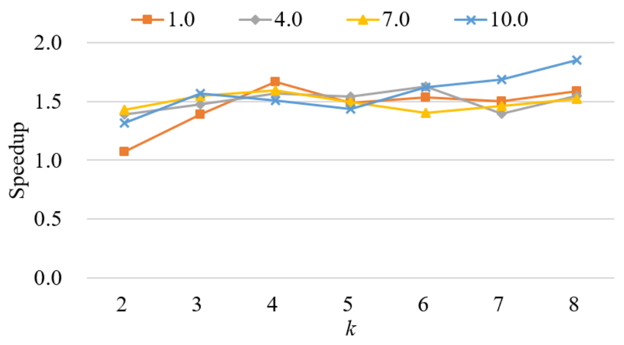


FIGURE 5. The speedup of the improved method to general piecewise polynomial fitting on the temperature dataset with different residual bounds and polynomial orders. Four curves correspond to four different values of error bound ε.

Specifically, the speedup is up to 2.82x at $k = 8$, $\varepsilon = 10$. The reason is that higher k generally means higher computation overhead, thus the superiority of the improved method is more obvious. Second, no significant trend in speedup is found as ε varies, but it seems that a higher speedup is received at the $\varepsilon = 5.0$ than $\varepsilon = 10.0, 15.0, 20.0$.

No matter periodically sampled or aperiodically sampled time-series data fitting, caching intermediate calculation results does not affect the fitting result, so these two methods have the same compression ratio and fitting accuracy.

Table 1 presents the compression ratio of two methods on the gas dataset with different residual bounds and polynomial orders, where the compression ratio is computed as the ratio of the amount of data to the number of polynomial coefficients of all segments. It can be seen from the data in the table that the compression ratio decreases with the increase of k. Obviously, increasing k means more polynomial coefficients. At the same value of k, we find that the compression ratio increases with increasing ε. The result is not surprising, because the piecewise polynomial fitting with the larger residual bound always generates fewer segments.

The comparison between raw data and fitted results on the gas dataset, with $k = 2$ and $\varepsilon = 5.0$, is presented in Fig. 2. It can be seen that the fitted curve is very close to the raw curve. Also, we record the root-mean-square error (RMSE) of each segment and find that they are all smaller than the given residual bound when $k \leq 8$. However, when continuing to increase the value of k, the RMSE of some segments will exceed the given residual bound ε because of Runge's phenomenon [49]. In other words, high-order polynomials are not always better than the low-order polynomials. Besides, different datasets or attributes, holding different data evolvements, have the variable optimal values of k. There is no specific way to determine the optimal value of k, the determining based on the prior knowledge is recommended.

The speedup of the improved method to general piecewise polynomial fitting on the power consumption dataset with different k and ε is presented in Fig. 3. Unlike the result on the gas dataset, the speedup on the power consumption dataset does not increase with the increase of k. In fact, with the increase of k, the corresponding computation cost will increase, and the speedup of the improved method to general piecewise polynomial fitting should also increase. However, it is non-negligible that the number of segments will decrease with the increase of k, which will weaken the acceleration effect.

The compression ratio on the power consumption dataset is presented in Table 2. Comparing it with Table 1, it can be seen that the compression ratio on the power consumption dataset is smaller than that on the gas dataset, and the reason can be found in Fig. 2 and Fig. 4. Obviously, the power consumption

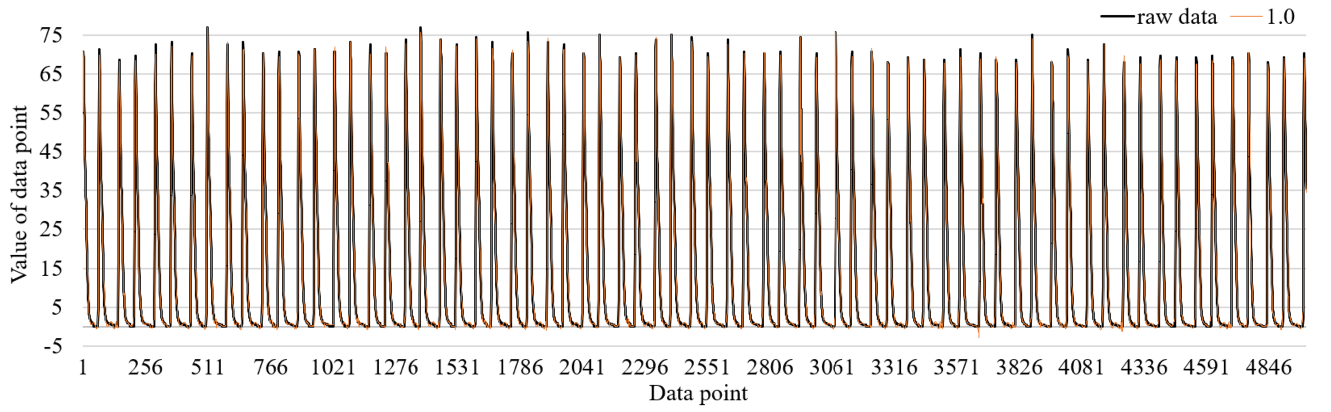


FIGURE 6. Comparison between raw data and fitted results on the temperature dataset with $k = 5$ and $\varepsilon = 1.0$.

TABLE 2. The compression ratio on the power consumption dataset with different residual bounds and polynomial orders.

ε \ k	2	3	4	5	6	7	8
0.1	6.98	6.41	6.10	5.16	3.84	2.64	1.71
0.4	29.54	28.11	22.35	12.63	6.55	3.60	2.11
0.7	120.81	104.58	46.37	17.34	7.73	4.00	2.29
1.0	1124.80	366.14	73.27	20.82	8.53	4.32	2.45

dataset has more complex and more frequent data changes than the gas dataset. Then, with the limitation of residual bound, more segmenting is required, thus the compression ratio is smaller. Fig. 4 shows the comparison between raw data and fitted results on the power consumption dataset with $k = 5$ and $\varepsilon = 0.1$. Although the dataset has complex data evolvement, it can be seen from the figure that the fitted curve is still very close to the raw curve.

Overall, the results presented in this part reveal the superiority of our improved method for fitting periodically sampled time-series data than general piecewise polynomial fitting, with no compression ratio and accuracy affected. The next subsection, therefore, moves on to discuss the performance of two methods for fitting aperiodically sampled time-series data.

2) APERIODICALLY SAMPLED TIME-SERIES DATA FITTING

The speedup on the temperature dataset with different residual bounds and polynomial orders is presented in Fig. 5. It can be seen that the speedup is stable at about 1.5x, and the maximum speedup is 1.85x when $k = 8$ and $\varepsilon = 10.0$. Table 3 shows the compression ratio on the temperature dataset with different k and ε. Similarly, the compression ratio increases with the increase of error bound and decreases with the increase of k. Besides, maximum compression ratio is 23.84 when $k = 2$ and $\varepsilon = 10.0$. The comparison between raw data and fitted results with $k = 5$ and $\varepsilon = 1.0$ on the temperature dataset is presented in Fig. 6. As we can see that

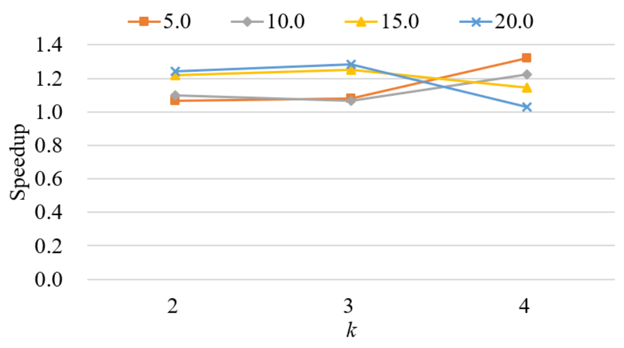


FIGURE 7. The speedup of the improved method to general piecewise polynomial fitting on the FoG dataset with different residual bounds and polynomial orders. Four curves correspond to four different values of error bound ε.

the method of piecewise polynomial fitting can fit the value trend of raw data well.

TABLE 3. The compression ratio on the temperature dataset with different residual bounds and polynomial orders.

ε \ k	2	3	4	5	6	7	8
1.0	4.45	4.32	4.11	4.13	4.09	4.00	3.74
4.0	10.46	9.02	8.08	9.17	8.97	8.85	7.94
7.0	19.14	16.88	14.21	12.11	10.42	9.13	8.12
10.0	23.84	18.19	14.63	12.19	10.46	9.70	9.69

Fig. 7 shows the speedup of the improved method to general piecewise polynomial fitting on the FoG dataset with different k and ε. The reason for only three groups of k selected is that it will be unable to fit when the value of k is greater than 4 because of the singularity of the intermediate result. In fact, from the data in Table 4, it can be seen that the polynomial fitting with $k = 2$ has achieved the highest compression ratio under the limitation of four groups of residual bounds. Fig. 8 presents the comparison between raw data and fitted result on

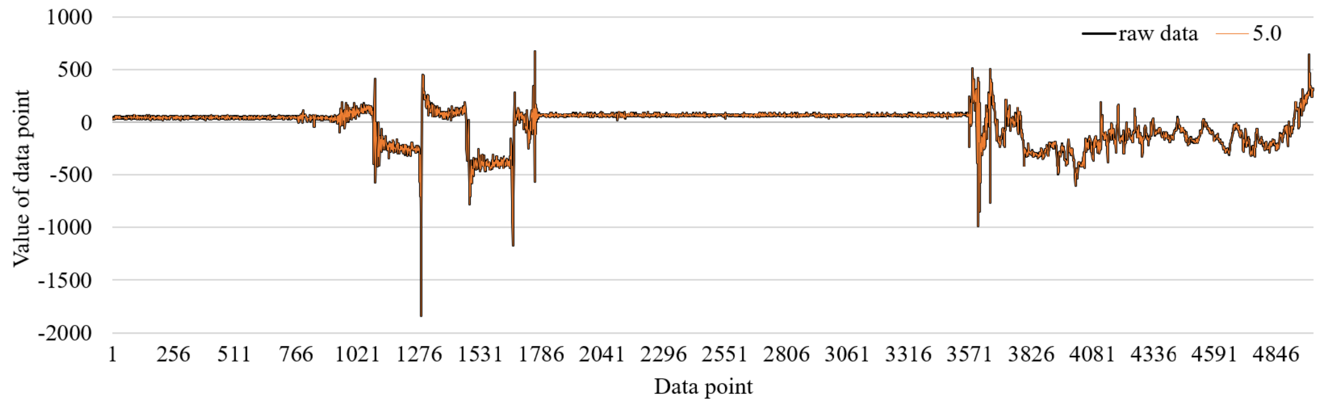


FIGURE 8. **Comparison between raw data and fitted results on the FoG dataset with $k = 2$ and $\varepsilon = 5.0$.**

TABLE 4. **The compression ratio on the FoG dataset with different residual bounds and polynomial orders.**

k \ ε	5.0	10.0	15.0	20.0
2	1.56	3.67	4.47	5.15
3	1.50	3.62	4.39	5.08
4	1.08	2.90	3.57	4.15

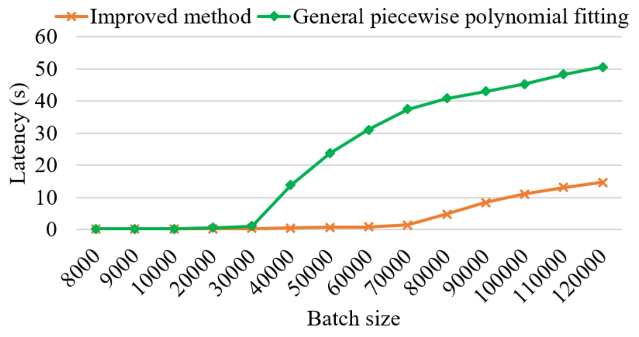


FIGURE 9. **The event-time latency comparison of two methods on the gas dataset with $k = 8$ and $\varepsilon = 5$.**

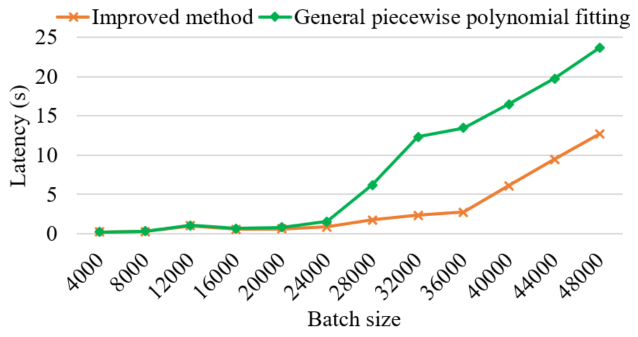


FIGURE 10. **The event-time latency comparison of two methods on the power consumption dataset with $k = 8$ and $\varepsilon = 0.4$.**

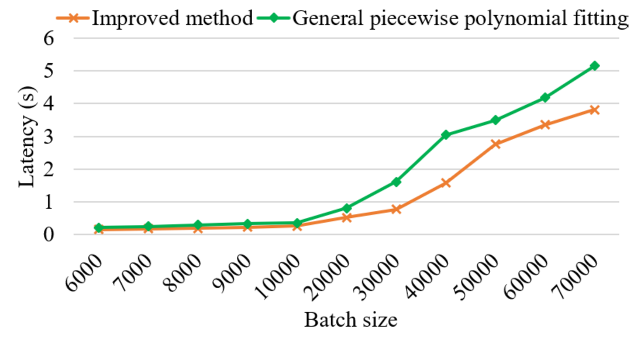


FIGURE 11. **The event-time latency comparison of two methods on the temperature dataset with $k = 8$ and $\varepsilon = 5$.**

the FoG dataset with $k = 2$ and $\varepsilon = 5.0$. Obviously, the fitted results are consistent with raw data.

Taken together, these results presented in part B suggest that the improved method has less time overhead than general piecewise polynomial fitting without affecting the compression ratio and accuracy. Moreover, stable speedup with the variation of residual bound and polynomial order indicates the good scalability of our method.

C. PERFORMANCE COMPARISON IN STREAMING ENVIRONMENT

In this subsection, we run tasks with the local mode of Apache Storm system to observe the performance of two methods in a streaming environment. Fig. 9, 10, 11 present the change of event-time latency [50] with the increase of batch size on the gas, power consumption, and temperature datasets, respectively. The event-time latency is computed as the interval between the moment that a tuple will be spouted and the moment that the streaming system has fully processed the tuple, and the batch size is the data amount of one tuple sent each time.

As can be seen from Fig. 9, both methods have excellent performance when the system is under-utilized (small batch size), and their latency performance collapses at a large batch size as a result of limited processing capability. The gap between the two curves indicates that the improved method can deal with a higher throughput rate than general piecewise polynomial fitting. On the power consumption dataset, as shown in Fig. 10, the latency performance of general piecewise polynomial fitting starts collapsing when the batch size is larger than 24000. In comparison, the improved method still has excellent latency performance until the batch size is larger than 36000.

As for the aperiodically sampled time-series data, the superiority of the improved method is lower than that on the periodically sampled time-series data. From Fig. 11, we can see that both methods start collapsing almost at the same batch size, but the improved method still has more excellent latency performance than general piecewise polynomial fitting. Besides, on the FoG dataset, we have experimentally found that the latency performance of the two methods is similar. This rather unexpected result might be explained by the fact that this dataset has a small number of data points. Polynomial fitting with low order takes low time cost. About 190,000 data points can not accurately simulate a stable streaming computing. Thus we have not presented the experimental result on the FoG dataset in this part.

VI. CONCLUSION

This paper sets out to fast piecewise polynomial fitting of time-series data for streaming computing. By analyzing the polynomial fitting of periodically and aperiodically sampled time-series data, we propose different improved methods. For the periodically sampled time-series data, the intermediate calculation results of polynomial fitting of different segment lengths are cached in a hash table indexed with the segment length to avoid redundant calculation. For the aperiodically sampled time-series data, the idea of the incremental calculation is adopted to accelerate the fitting process. The experimental results on four time-series datasets show that the highest speedup of our improved method to general piecewise polynomial fitting for fitting periodically sampled time-series data is up to 2.82x and aperiodically sampled time-series data is up to 1.85x, without affecting the compression ratio and fitting accuracy. Moreover, performance comparison on a streaming environment shows that the improved method can endure higher throughput than the general piecewise polynomial fitting with the same event-time latency.

REFERENCES

[1] S. Gandhi, L. Foschini, and S. Suri, "Space-efficient online approximation of time series data: Streams, amnesia, and out-of-order," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, Mar. 2010, pp. 924–935.

[2] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, and Y. Mu, "Piecewise linear approximation of streaming time series data with max-error guarantees," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 173–184.

[3] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith, "Efficient retrieval for browsing large image databases," in *Proc. 5th Int. Conf. Inf. Knowl. Manage. (CIKM)*. New York, NY, USA: ACM, 1996, pp. 11–18, doi: 10.1145/238355.238365.

[4] K. V. Ravi Kanth, D. Agrawal, A. E. Abbadi, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *Comput. Vis. Image Understand.*, vol. 75, nos. 1–2, pp. 59–72, Jul. 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314299907622

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

[6] Š. Beneš and J. Kruis, "Singular value decomposition used for compression of results from the finite element method," *Adv. Eng. Softw.*, vol. 117, pp. 8–17, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997817306592

[7] J. Guo, R. Xie, and G. Jin, "An efficient method for NMR data compression based on fast singular value decomposition," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 2, pp. 301–305, Feb. 2019.

[8] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Foundations of Data Organization and Algorithms*, D. B. Lomet, Ed. Berlin, Germany: Springer, 1993, pp. 69–84.

[9] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim, "Fast similarity search in the presence of noise, scaling, and translation in time-series databases," in *Proc. 21st Int. Conf. Very Large Data Bases (VLDB)*, Zurich, Switzerland, Sep. 1995, pp. 490–501. [Online]. Available: http://www.vldb.org/conf/1995/P490.PDF

[10] P. Nair, A. Popli, and K. N. Chaudhury, "A fast approximation of the bilateral filter using the discrete Fourier transform," *Image Process. Line*, vol. 7, pp. 115–130, May 2017.

[11] C. I. Kithulgoda, R. Pears, and M. A. Naeem, "The incremental Fourier classifier: Leveraging the discrete Fourier transform for classifying high speed data streams," *Expert Syst. Appl.*, vol. 97, pp. 1–17, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741730845X

[12] N. Jain, M. Singh, and B. Mishra, "Image compression using 2D-discrete wavelet transform on a light weight reconfigurable hardware," in *Proc. 31st Int. Conf. VLSI Des. 17th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2018, pp. 61–66.

[13] P. V. Sundara Rajan and A. L. Fred, "An efficient compound image compression using optimal discrete wavelet transform and run length encoding techniques," *J. Intell. Syst.*, vol. 28, no. 1, pp. 87–101, Jan. 2019.

[14] W. Xu, F. Fu, Y. Wang, and J. Wang, "Discrete wavelet transform-based fast and high-efficient lossless intraframe compression algorithm for high-efficiency video coding," *J. Electron. Imag.*, vol. 28, no. 1, pp. 1–17, Jan. 2019, doi: 10.1117/1.JEI.28.1.013017.

[15] G. Wang, W. Wang, Q. Fang, H. Jiang, Q. Xin, and B. Xue, "The application of discrete wavelet transform with improved partial least-squares method for the estimation of soil properties with visible and near-infrared spectral data," *Remote Sens.*, vol. 10, no. 6, p. 867, Jun. 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/6/867

[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, Aug. 2001, doi: 10.1007/PL00011669.

[17] C. Guo, H. Li, and D. Pan, "An improved piecewise aggregate approximation based on statistical features for time series mining," in *Knowledge Science, Engineering and Management*, Y. Bi and M.-A. Williams, Eds. Berlin, Germany: Springer, 2010, pp. 234–244.

[18] V. S. Siyou Fotso, E. Mephu Nguifo, and P. Vaslin, "Grasp heuristic for time series compression with piecewise aggregate approximation," *RAIRO–Oper. Res.*, vol. 53, no. 1, pp. 243–259, Feb. 2019, doi: 10.1051/ro/2018089.

[19] Y. Zhu, D. Wu, and S. Li, "A piecewise linear representation method of time series based on feature points," in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R. J. Howlett, and L. Jain, Eds. Berlin, Germany: Springer, 2007, pp. 1066–1072.

[20] Q. Xie, C. Pang, X. Zhou, X. Zhang, and K. Deng, "Maximum error-bounded piecewise linear representation for online stream approximation," *VLDB J.*, vol. 23, no. 6, pp. 915–937, Apr. 2014, doi: 10.1007/s00778-014-0355-0.

[21] 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.

[22] F. Grützmacher, B. Beichler, A. Hein, T. Kirste, and C. Haubelt, "Time and memory efficient online piecewise linear approximation of sensor signals," *Sensors*, vol. 18, no. 6, p. 1672, May 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/6/1672

[23] R. Duvignau, V. Gulisano, M. Papatriantafilou, and V. Savic, "Piecewise linear approximation in data streaming: Algorithmic implementations and experimental analysis," *CoRR*, vol. abs/1808.08877, 2018. [Online]. Available: http://arxiv.org/abs/1808.08877

[24] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, Jun. 2002, doi: 10.1145/568518.568520.

[25] L. Junkui and W. Yuanzhen, "APCAS: An approximate approach to adaptively segment time series stream," in *Advances in Data and Web Management*, G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, Eds. Berlin, Germany: Springer, 2007, pp. 554–565.

[26] H. Wang, "An APCA-enhanced compression method on large-scale time-series data," in *Proc. ACM Turing 50th Celebration Conf. (China-ACM TUR-C)*. New York, NY, USA: ACM, 2017, pp. 20:1–20:6, doi: 10.1145/3063955.3063975.

[27] J. Ferguson and P. A. Staley, "Least squares piecewise cubic curve fitting," *Commun. ACM*, vol. 16, no. 6, pp. 380–382, Jun. 1973, doi: 10.1145/362248.362276.

[28] C. L. Lawson, "Characteristic properties of the segmented rational minmax approximation problem," *Numerische Math.*, vol. 6, no. 1, pp. 293–301, Dec. 1964, doi: 10.1007/BF01386077.

[29] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comput.*, vol. C-23, no. 8, pp. 860–870, Aug. 1974.

[30] K. Ichida, F. Yoshimoto, and T. Kiyono, "Curve fitting by a piecewise cubic polynomial," *Computing*, vol. 16, no. 4, pp. 329–338, Dec. 1976, doi: 10.1007/BF02252081.

[31] G. Chen, Z. L. Ren, and H. Z. Sun, "Curve fitting in least-square method and its realization with MATLAB," *Ordnance Ind. Automat.*, vol. 3, p. 063, 2005.

[32] D. Li, J. Shen, and J. Xie, "Study on representation of time series based on subsection polynomial fitting," in *Proc. 4th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, 2007, pp. 16–20.

[33] C.-F. Yan, J.-F. Fang, L.-L. Zhou, and L.-X. Wu, "A novel approach of time series trend extraction based on global constrained multi-segment polynomial fitting," in *Proc. Mater., Manuf. Technol., Electron. Inf. Sci.*, Apr. 2016, pp. 358–370.

[34] M. Novosadová and P. Rajmic, "Piecewise-polynomial curve fitting using group sparsity," in *Proc. 8th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2016, pp. 320–325.

[35] M. Beitollahi and S. A. Hosseini, "Using curve fitting for spectral reflectance curves intervals in order to hyperspectral data compression," in *Proc. 10th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, Jul. 2016, pp. 1–5.

[36] M. Beitollahi and S. A. Hosseini, "Using Savitsky–Golay filter and interval curve fitting in order to hyperspectral data compression," in *Proc. Iranian Conf. Electr. Eng. (ICEE)*, May 2017, pp. 1967–1972.

[37] S. A. Hosseini and H. Ghassemian, "Rational function approximation for feature reduction in hyperspectral data," *Remote Sens. Lett.*, vol. 7, no. 2, pp. 101–110, Nov. 2015, doi: 10.1080/2150704X.2015.1101180.

[38] S. A. Hosseini and H. Ghassemian, "Hyperspectral data feature extraction using rational function curve fitting," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 1, 2016, Art. no. 1650001.

[39] K. Ose, K. Iwata, and N. Suematsu, "Sampling shape contours using optimization over a geometric graph," *IEICE Trans. Inf. Syst.*, vol. E102.D, no. 12, pp. 2547–2556, Dec. 2019.

[40] 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. 4th Int. Conf. Knowl. Discovery Data*, 1998, pp. 239–243.

[41] F. Korn, N. Sidiropoulos, C. Faloutsos, E. L. Siegel, and Z. Protopapas, "Fast nearest neighbor search in medical image databases," in *Proc. 22nd Int. Conf. Very Large Data Bases (VLDB)*, Mumbai, India, Sep. 1996, pp. 215–226. [Online]. Available: http://www.vldb.org/conf/1996/P215.PDF

[42] X. Liu and Y. Wang, "Research of automatically piecewise polynomial curve-fitting method based on least-square principle," *Sci. Technol. Eng.*, vol. 14, no. 3, pp. 55–58, 2014.

[43] A. Toshniwal, J. Donham, N. Bhagat, S. Mittal, D. Ryaboy, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, and M. Fu, "Storm@Twitter," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: ACM, 2014, pp. 147–156, doi: 10.1145/2588555.2595641.

[44] J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," *Sens. Actuators B, Chem.*, vol. 215, pp. 618–629, Aug. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925400515003524

[45] G. Hebrail and A. Berard. (Aug. 2012). *Individual Household Electric Power Consumption Data Set*. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

[46] M. Bachlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Troster, "Wearable assistant for Parkinson's disease patients with the freezing of gait symptom," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 436–446, Mar. 2010, doi: 10.1109/TITB.2009.2036165.

[47] J. Burgués, J. M. Jiménez-Soto, and S. Marco, "Estimation of the limit of detection in semiconductor gas sensors through linearized calibration models," *Anal. Chim. Acta*, vol. 1013, pp. 13–25, Jul. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003267018301673

[48] J. Burgués and S. Marco, "Multivariate estimation of the limit of detection by orthogonal partial least squares in temperature-modulated MOX sensors," *Analytica Chim. Acta*, vol. 1019, pp. 49–64, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003267018303702

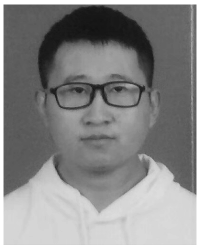
[49] C. Runge, "Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten," *Zeitschrift Math. Phys.*, vol. 46, nos. 224–243, p. 20, 1901.

[50] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl, "Benchmarking distributed stream data processing systems," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 1507–1518.

JIANHUA GAO was born in Shanxi, China, in 1995. She received the B.S. degree from the College of Mathematics, Taiyuan University of Technology, China, in 2017. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology. Her research interests mainly include parallel computing and heterogeneous computing.

WEIXING JI was born in Shanxi, China, in 1980. He received the B.S. and Ph.D. degrees from the School of Computer Science and Technology, Beijing Institute of Technology, in 2003 and 2008, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests mainly include big data analysis, parallel computing, and program analysis and optimization.

LULU ZHANG was born in Henan, China, in 1990. She received the B.S. and M.S. degrees from the School of Computer Science and Technology, Beijing Institute of Technology, China, in 2015 and 2018, respectively. Her specialty is big data analysis.



SENHAO SHAO was born in Shanxi, China, in 1997. He received the B.S. degree from the College of Data Science, Taiyuan University of Technology, China, in 2019. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, Beijing Institute of Technology. His specialty is high performance computing.

YIZHUO WANG was born in Shanxi, China, in 1979. He received the B.S. and Ph.D. degrees from the School of Computer Science and Technology, Beijing Institute of Technology, China, in 2000 and 2005, respectively. He is currently a Lecturer with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests mainly include computer architecture, parallel programming, and embedded systems.

FENG SHI was born in 1961. He received the B.S. degree from the School of Physics, Peking University, China, in 1983, and the Ph.D. degree from the School of Computer Science and Technology, Beijing Institute of Technology, China, in 1999. He is currently a Full Professor with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests mainly include computer architecture and embedded systems.