IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 28, 2020, accepted May 4, 2020, date of publication May 25, 2020, date of current version June 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997337

Multi-Source Data Stream Online Frequent Episode Mining

TAO YOU, YAMIN LI, BINGKUN SUN, AND CHENGLIE DU
Department of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China

Corresponding author: Tao You (youtao@nwpu.edu.cn)

This work was supported in part by the National Science Foundation for Young Scientists of China under Grant 61801392, in part by the Equipment Pre Research Project under Grant 61400010207, in part by the National Key Research and Development Program of China under Grant 2017YFB1001900 and Grant 2018YFB2101304, in part by the Defense Industrial Technology Development Program under Grant JCKY2016607B006 and Grant JCKY2018205B001, and in part by the Special Civil Aircraft Research Program under Grant MJ-2017-S-39.

ABSTRACT Online frequent episode mining is more complicated than the traditional static frequent episode mining due to the continuous, unbounded and time-varying data stream. Especially in the multiple data streams, online frequent episode mining is more difficult than the single-source stream, due to the concurrency, global clock loss, and uncertainty of delay caused by the distributed environment. To cope with these problems, we propose a new algorithm. Firstly, the data stream with ''happen-before'' relationship among multiple sources is combined on the global data lattice. Next, the traversal on global data lattice generates effective parallel and serial candidate data streams, which guarantee the accuracy of subsequent mining and reduce the number of global sequences during searching process. Then, we use the frequent episode tree to detect the expanding online serial episodes and parallel episodes. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments.

INDEX TERMS Episode, global data lattice, multi-source data stream.

I. INTRODUCTION

Data stream frequent episode mining techniques are broadly used in network security monitoring analyzing [1], financial securities management [2], mobile communication services [3], and sensor data processing [4]. When there are multiple data sources in a system, data streams from different sources may often affect others [5]. In the distributed system, it is necessary to combine the data of each source to preserve the integrated environment information among the data streams. More importantly, the data sources may not have the global clock [6]–[8]. The heterogeneity and constrained resources of data sources may lead to diverse and unpredictable computation delay [9], [10]. Implementing a multi-source data stream can overcome the limitations of the single-source data stream, and combine to filter out interference information to form a comprehensive decision on the results, which can produce more accurate, reliable, and effective data compared with the single-source data.

Another application is the wireless sensor network scenario [11], [12]. The sensor nodes deployed in the monitoring area cooperate to perceive, collect and process the information of the perceived objects in the network coverage area and send them to the observer. By combining the data collected by each sensor node [13], we can realize the perception of the object and obtain the complete information. For example, the data streams, {ABADCCA...} and {HFFEGHH...} are based on the data of two nodes in the wireless sensor network system, where B occurs before E, G occurs before D, and the observer captures the data stream in one given order [14], {HFFABEGADCCAHH...}. Based on this scenario, we study the features that consider the order in which data items occur. The order feature is derived from the ''happen-before'' relation on the data streams. Therefore, we propose an online frequent episode mining algorithm for the multi-source data stream based on order features. In fact, to our knowledge, there is no published work defining order features in data mining.

However, for fast-growing sequence data, old episodes may become obsolete while new useful episodes keep emerging. The data stream contains an implicit data feature, the concept drift. The concept is the target concept hidden in the data stream. The phenomenon that the concept fundamentally changes with the flow of the data stream is called concept

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

drift. Additionally, a solution is needed to discover the latest frequent episodes from a growing data stream. We create the frequent episode tree by using a variable sliding window to adapt the concept drift, then we propose a streaming frequent episode mining solution to adapt dynamic changes in the data stream.

Faced with the significantly increasing number of data sources, we gradually tend to integrate the computing power of distributed data sources to maximize the use of existing resources. The combinations of the multi-source data streams filter out interference information to form a comprehensive decision on the result, which in turn can produce more accurate, reliable, and effective information that cannot be obtained from a single data source.

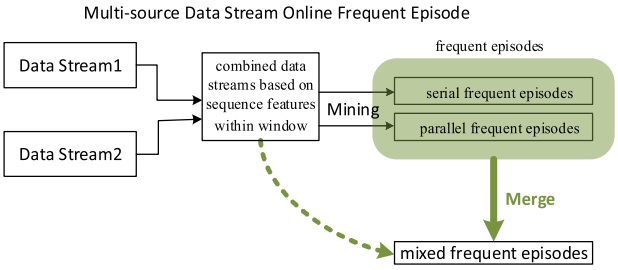

FIGURE 1. Multi-source Data Stream Online Frequent Episode.

In this paper, we jointly consider data combinations and frequent episodes mining that adjusts to the multi-source data stream [15], [16], as shown in Fig. 1. First, to save the data information among multiple sources, we present a system model and order features. Then, we propose combined data streams based on order features. Moreover, an online algorithm is designed to deal with the episode mining in a dynamic manner. Finally, the frequent episode mining is mainly divided into serial episodes and parallel episodes, then the mixed episodes merge of the existing serial and parallel frequent episodes.

To our best knowledge, it is the first time that the research of multi-source data stream online frequent episode mining is proposed. The main contributions of the paper can be summarized as follows:

1. The model of the multi-source data stream with ''happen-before'' relation is built, and the global data lattice of data streams is presented to maintain the global data information.

2. Taking into account the data lattice with order features, the concept of multi-sequential data lattice is proposed to reduce the space overhead.

3. Merge serial frequent episodes and parallel frequent episodes, we get the complete and effective frequent episodes.

The rest of the paper is organized as follows. In Section II, we review recent related work. Section III presents a system model. Section IV proposes order features, which describe data lattices contain single-sequential data lattice and multi-sequential data lattice in detail. Section V proposes a mining framework for the data stream frequent episode mining, Section VI describes the MDSOFE algorithm and analyzes the complexity. Then, the experiment results are discussed in Section VII. Finally, Section VIII concludes this paper and points out the application areas.

II. RELATED WORK

We study the problem of multi-source data stream online frequent episode mining, However, there are several kinds of research related to this work, including multi-source data stream combination and online frequent episode mining.

For multi-source data stream inputs, however, it is very difficult to construct a model to record the information from these sources. Typical data stream technology is related to the single-source data stream. Unfortunately, most existing works focus on data stream frequent episode mining, ignoring that the data sources are multi-level, multi-angle and multi-faceted, and the traditional single-source data stream mining [17], [18] cannot adapt with social progress. A new sampling system based on binary Bernoulli sampling to support a multi-source data stream environment was made by Wonhyeong Cho *et al*. [15]. The approach assumed each input site receives and transmits candidate data independently, but they did not consider the relation among distinct sources. In our work, we impose a multi-source data stream combination with order features.

Online frequent episode mining of data stream has been widely researched [19], [20], most stream processing model can be divided into the following three types: landmark window model [21], sliding window model [22], damped window model [23]. Leung proposed two sliding window-based algorithms, UF-streaming and SUF-Growth [23], of which each sliding window contains a fixed number of data when a window is full, it will first remove old data from the window, and then add the new data to the window. Manku [24] and Rajeev Motwani proposed the Sticky Sampling algorithm and the Lossy Counting algorithm to mine frequent episodes in the data stream and extend the Lossy Counting [25] algorithm to mine frequent episode sets in the data stream. Graham Cormode and S.Muthukrishnan considered the Count-Min SKETCH data structure [26], which allowed for quick querying. However, these researches can be only applied to mine frequent episodes with a single-source data stream, which makes it limited in practical use.

The problem for multi-source data stream online frequent episode mining has high complexity and constraints. To find the useful and accurate frequent episodes, the algorithm should consider all possible multi-source data stream combination as well as the fast-growing data streams. Especially in applications with distributed requirements, it is not acceptable to solve this problem by using the existing methods.

III. SYSTEM MODEL

In this section, we first describe the data generated by the multi-source data stream with ''happen-before'' relation, then we discuss the global lattice of data streams. Finally,

TABLE 1. Notations used in the data.

Notations	Explanation
$W^{(n)}$	Data source
$d_i^{(k)}$	Data on $W^{(k)}$
G_{ij}	Global data point in the global data lattice
$G_{ij}^{(k)}$	Global data point on $W^{(k)}$
m,n	The number of data points in the data source
$S_{(n-1)}$	Single-sequential data lattice
$M_{(m-1)(n-1)}$	Multi-sequential data lattice

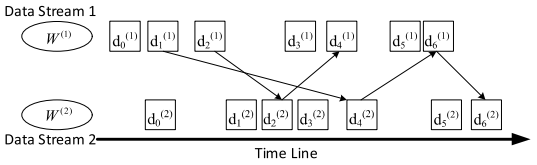

FIGURE 2. Data streams with interactions.

we introduce the 3-dimensional sliding window over data streams. Notations used in this section are listed in Table 1.

A. DATA WITH "HAPPEN-BEFORE" RELATION

When monitoring a distributed system, we are faced with multiple distributed data sources, which generate data streams at runtime. The data sources do not necessarily have the global clock [27], [28]. They are modeled as n sources with message passing among each other, e.g., $W^{(1)}$, $W^{(2)}, \ldots, W^{(n)}$. Each data source $W^{(n)}$ produces the data stream $d_1^{(k)}$, $d_2^{(k)}$, $d_3^{(k)}$, …, $d_n^{(k)}$, as shown in Fig. 2. The arrival of data means the growth of data stream, and we cannot ignore that the communication via sending/receiving messages among data streams. For example, in an intelligent transportation system, a car goes many road sections to its destination, according to the information of its passed road collected by traffic sensors on different vehicles, we can find the road sections that multiple vehicles may pass and regard them as the data episodes which occur frequently in the data streams. Sensors collect interactive information to facilitate further data causality reasoning.

We are now ready to give a precise definition of the "happen-before" relation [9] (denoted by '→') resulting from data causality.

Definition 1 ("Happen-Before" Relation): Given two data $d_i^{(k1)}$ and $d_j^{(k2)}$, we have $d_i^{(k1)} \rightarrow d_j^{(k2)}$ [29] if and only if one of the following three conditions is satisfied:

1. $d_i^{(k1)}$ and $d_j^{(k2)}$ are on the same process (k1 = k2) and $d_i^{(k1)}$ is generated before $d_j^{(k2)}$, or 2. $d_i^{(k1)}$ and $d_j^{(k2)}$ are on the different process (k1≠k2), meanwhile, $d_i^{(k1)}$ and $d_j^{(k2)}$ are the corresponding sending and receiving of the data message, or 3. there exists $d_r^{(k3)}$, $d_i^{(k1)} \rightarrow d_r^{(k3)}$, $d_r^{(k3)} \rightarrow d_j^{(k2)}$ such that $d_i^{(k1)} \rightarrow d_r^{(k3)} \rightarrow d_j^{(k2)}$.

Note that this definition has $d \rightarrow d$ for any data, that is, the "happen-before" relation satisfies reflexivity, anti-symmetry and transitivity. In Fig. 2, data streams extra have the relations, $d_1^{(1)} \rightarrow d_4^{(2)}$, $d_2^{(1)} \rightarrow d_2^{(2)}$, $d_2^{(2)} \rightarrow d_4^{(1)}$, $d_4^{(2)} \rightarrow d_6^{(1)}$, $d_6^{(1)} \rightarrow d_6^{(2)}$, which can represent interactions of the different data sources.

In this paper, the relation in the order features contains the order of the events in time and the causal relationship of events in logical. There are uncertainties in the "happen-before" relation of the concurrent data due to the multi-source data stream. Therefore, when the multi-source data streams are synthesized, there are many possible scenarios for the data status of the entire data stream, which are observed in the traditional model, that is, there are multiple global sequences. Global sequences will be described below in more detail.

B. GLOBAL DATA LATTICE OF DATA STREAMS

The construction of a lattice [30] is implemented by multi-source data stream sources. For a system of data streams, we are thus concerned with global data points of the system. A global data point may be either non-happen-before or happen-before. The notion of the non-happen-before data point is crucial for data stream processing. Intuitively, a global data point with two data $d_i^{(k1)}$ and $d_j^{(k2)}$ is non-happen-before, if $d_i^{(k1)}$ and $d_j^{(k2)}$ do not have a strict "happen-before" relation with each other, for example, data $d_0^{(1)}$ and $d_0^{(2)}$ from two sources form a non-happen-before data point means that sequence $\{d_0^{(1)}d_0^{(2)}\}$ and $\{d_0^{(2)}d_0^{(1)}\}$ both could occur in the global sequence. In the Global Data Lattice (GDL), black dots '·' denote the non-happen-before data points but the edges between them depict the "happen-before" relation (denoted by '→'), the crosses '×' denote the happen-before data points. The lattice structure serves as a key notion for the detection of global predicates over data streams. The generation of the sequence has the following three steps:

Step 1: Build a Single Data Status Table:

We use the following principles to build a single data status table:

1. Initially set all the local data status in the process to '-1'.
2. If there is a "happen-before" relation between two points in different sources, e.g., $d_i^{(k1)} \rightarrow d_j^{(k2)}$, we use the subscript i of the data $d_i^{(k1)}$ to describe the corresponding status of $d_j^{(k2)}$ in order that the interaction between these data can be described.
3. If there is a "happen-before" relation between two points in the same sources, i.e., k1=k2 for $d_i^{(k1)} \rightarrow d_j^{(k2)}$ there exist two consecutive local status in the same process, the last status inherits the value of the previous status.

As shown in Fig. 2, the source $W^{(1)}$ has the local data status $d_4^{(1)}$ in the process of $d_3^{(1)} \rightarrow d_4^{(1)}$, the process of $d_2^{(2)} \rightarrow d_4^{(1)}$, the value of $d_3^{(1)}$ is '-1', the value of $d_2^{(2)}$ is 2, so the value of $d_4^{(1)}$ is expressed as 2. We record the status table of the sources in Fig. 3. Indicates that the local status $d_0^{(2)}$, $d_1^{(2)}$, $d_2^{(2)}$ of $W^{(2)}$ and $d_4^{(1)}$ have the "happen-before" relation, and the global data points G_{40}, G_{41}, G_{42} are happen-before data points, denoted by '×'.

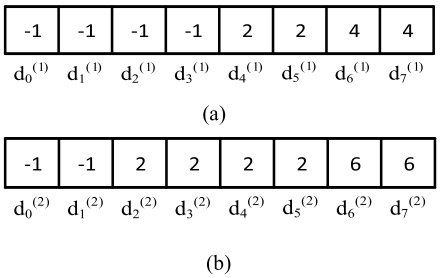

FIGURE 3. The status table of W(1), W(2). (a) the status table of data in source W(1). (b) the status table of data in source W(2).

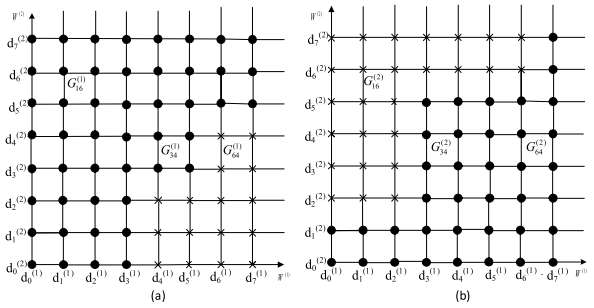

FIGURE 4. Diagram of $W^{(1)}$, $W^{(2)}$. (a) diagram generated from source $W^{(1)}$. (b) diagram generated from source $W^{(2)}$.

Step 2: Build a Single Process Data Diagram: On the basis of the status table constructed in the first step, the data diagram of each source can be made, and the single data diagram is established on the existed relation between the sources to record happen-before data points and non-happen-before data points. In Fig. 4(a), the status value of $d_1^{(1)}$ in the source $W^{(1)}$ is '-1', which indicates that there does not exist the "happen-before" relation between $d_1^{(1)}$ and $W^{(2)}$ at the moment, and all the dots of the column $d_1^{(1)}$ are initialized as non-happen-before data points, e.g., $G_{16}^{(1)}$ is '•' in Fig 4a. In Fig 4b, the status value of $d_6^{(2)}$ in $W^{(2)}$ is '6', which indicates that there exists the "happen-before" relation between $d_6^{(2)}$ and the first seven data in $W^{(1)}$. $d_0^{(1)}$, $d_1^{(1)}$, $d_2^{(1)}$, $d_3^{(1)}$, $d_4^{(1)}$, $d_5^{(1)}$, $d_6^{(1)}$ in row $d_6^{(2)}$ are denoted as the happen-before data points, their corresponding points is denoted by '×', e.g., $G_{16}^{(2)}$ is denoted by '×' in Fig. 4(b).

Step 3: Build the Global Data Lattice: When the single process data diagram form multi-source data stream global status with combination, they meet the "OR" operator, boolean form of the non-happen-before data point is defined as false, happen-before data point is defined true, e.g., $G_{16}^{(1)}$ is false, $G_{16}^{(2)}$ is true, so $G_{16} = G_{16}^{(1)} \| G_{16}^{(2)}$ is true, that means G_{16} in the GDL is happen-before data point. $G_{34}^{(1)}$ is false, $G_{34}^{(2)}$ is false, so $G_{34} = G_{34}^{(1)} \| G_{34}^{(2)}$ is false; $G_{64}^{(1)}$ is true, $G_{64}^{(2)}$ is false, so $G_{64} = G_{64}^{(1)} \| G_{64}^{(2)}$ is true. And so on, we connect the non-happen-before data points, it becomes the GDL, which can characterize the concurrency and "happen-before" relation in the data stream. Fig. 5 shows the GDL of the above example.

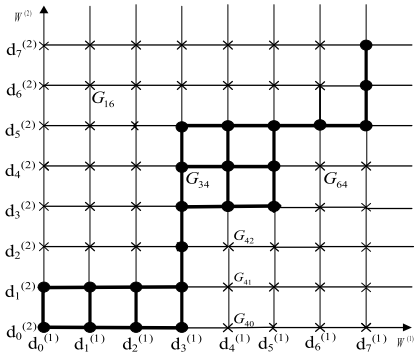

FIGURE 5. Global Data Lattice.

Algorithm 1 BuildGDL

Input: table[i]: the single status value table[i] of $d_i^{(k)}$ in data status table, k: the number k of the data streams, n: the length of the array table
Output: GDL: a binary file constituted by 0 and
1 for i ←0 to n
2 if (table[i] = −1)
3 Initialize the global status;
4 else
5 for j ←0 to table[i]
6 draw '×' in the diagram;
7 table[j] ← false;
8 for j ← table[i]+1 to n
9 draw '•' in the diagram;
10 table[j] ← true;
11 for i ←0 to n
12 for t ←0 to k
13 table[i] ← table[i] || table[j];
14 return GDL;

The three steps above introduce the construction framework of the lattice. As is shown in Algorithm 1, we transform the single status value into a diagram, line 3 initializes the status value to '−1'. Then we judge the status of the process is "non-happen-before" relation or "happen-before" relation in lines 5 to 10. According to Algorithm 1, the global data diagram of the single process will be operated to do "logical OR" merger in lines 11 to 13, so that we can get the GDL under the multi-source data stream environment. In short, lines 1 to 10 constructed the whole data diagram of all the sources, and lines 11 to 13 will use the "OR" operator to merge data points to get the final GDL we need.

C. THE N-SOURCE DATA STREAMS

It is similar to the synthesis of two-source data streams, the n-source data streams synthesis is also divided into the construction of a single data status table, the construction of a single process data diagram and the construction of a GDL, e.g., in the process of building the GDL, for the n sources $W^{(1)}$, $W^{(2)}$, $W^{(3)}$, $\cdots$, $W^{(n)}$, the global data lattice G

has $G=G^{(1)}||G^{(2)}||\ G^{(3)}||\ldots||G^{(n)}$. Then, we can search the data stream to get a global sequence in the GDL.

D. GDL COMPLEXITY ANALYSIS

Assume that the space for storing a GDL is one unit, the time of sequence conversion process is negligible, the number of the data stream is d, and we set m to describe the number of data.

For the original GDL, the worst-case space cost is $O(d^n)$, where n is the number of the sources in a multi-source environment. Each node of the lattice stores global data information and a predecessor node and a subsequent node of data information. Among them, the global status of the n-source data streams is n-dimensional, and the upper bound of the length of the linked list is $O(n)$, so the lattice storage space cost is $O(n^2d^n)$. We compare the new data with other n-1 data to see if there is the ''happen-before'' relation between them, so the time complexity of constructing new global data points is $O(n^2)$. The worst-case time of GDL is $O(n^3d^n)$, where d^n represents the number of GDL in the window and n is the number of the data source.

IV. DATA LATTICE WITH ORDER FEATURES

Before we describe our research on single-sequential data lattice, multi-sequential data lattice and complex data lattice formed by non-happen-before data points, we begin this section by introducing the basic concepts in this paper.

Definition 2 (Single-Sequential Data Lattice): Given data sources $W^{(i)}$, $W^{(j)}$ with generated data, we assume that $d_{k1}^{(i)} \in W^{(i)}$, $d_{k1}^{(j)}$, $d_{k2}^{(j)}$, $d_{k3}^{(j)}$,…,$d_{kn}^{(j)} \in W^{(j)}$, if and only if the uniquely deterministic sequence $\{d_{k1}^{(i)}d_{k1}^{(j)}d_{k2}^{(j)}d_{k3}^{(j)}\ldots d_{kn}^{(j)}\}$ exits among the data lattice, it is said that the data lattice formed by this sequence is the single-sequential data lattice, denoted by $S_{(n-1)}$, where n is the number of data points in the single-sequential data lattice.

Definition 3 (Multi-Sequential Data Lattice): Given data sources $W^{(i)}$, $W^{(j)}$ with generated data, we assume that $d_{k1}^{(i)}$, $d_{k2}^{(i)}$, $d_{k3}^{(i)}$, …, $d_{km}^{(i)} \in W^{(i)}$, $d_{k1}^{(j)}$, $d_{k2}^{(j)}$, $d_{k3}^{(j)}$, …, $d_{kn}^{(j)} \in W^{(j)}$, when many uncertain sequences exits among the data lattice, and the data corresponding to different sources have the ''happen-before'' relation with each other, denoted by $M_{(m-1)(n-1)}$, where m, n is the number of data points corresponding to the sources in the multi-sequential data lattice.

We reinterpret the notion of time based on lattice's definition of the ''happen-before'' relation among the data items, corresponding to the happen-before data points. The data arrives at runtime deterministically and irrevocably, the GDL is further divided into single-sequential data lattice and multi-sequential data lattice. Single-sequential data lattice which has the feature of small computational memory space overhead can generate the unique sequence, while the number of sequences in the multi-sequential data lattice is uncertainty under the condition of the ''happen-before'' relation. In this paper, we propose a method for constructing binary sequences which can reduce the space overhead by generating simple binary sequences to represent various possible sequences in the GDL. Note that there is a situation where the GDL overlap, we extend the definition of the multi-sequential data lattice to handle all these cases, and call it as the complex data lattice. Our next step is to define the complex data lattice we are interested in.

Definition 4 (Complex Data Lattice): When two groups of multi-sequential data lattices are confirmed in the non-happen-before data stream, and at least one adjacent side exists between two groups of multi-sequential data lattices, we define the lattice with the above characteristics as the complex data lattice.

It is known from the GDL model of two source data streams that with the combination of data streams, the GDL has single-sequential, multi-sequential and complex data lattice. Single-sequential data lattice generates the uniquely deterministic sequence $\{d_{k1}^{(i)}d_{k1}^{(j)}d_{k2}^{(j)}d_{k3}^{(j)}\ldots d_{kn}^{(j)}\}$ to reduce computational memory space, we can easily demonstrate a single sequence. In further text, we consider multiple possible sequences for the multi-sequential and complex data lattice. To counter this problem, we introduce binary sequences to reserve all the possibilities in different sources.

Definition 5 (Binary Sequences): Given two data streams from data sources, we define the structure

$$\begin{Bmatrix} d_{k1}^{(i)}, d_{k2}^{(i)}, d_{k3}^{(i)}, \cdots, d_{km}^{(i)} \\ d_{k1}^{(j)}, d_{k2}^{(j)}, d_{k3}^{(j)}, \cdots, d_{kn}^{(j)} \end{Bmatrix}$$

containing all data items occurring from data sources, line 1 represents the data stream from $W^{(1)}$, and line 2 represents the data stream from $W^{(2)}$, note that we can reverse the first and second rows in curly braces.

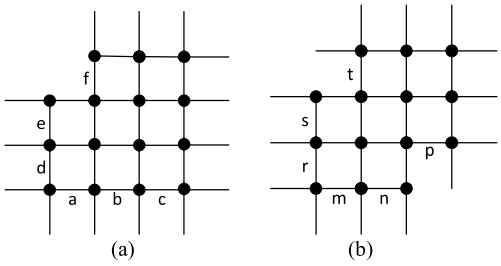

FIGURE 6. Complex data lattice. (a) multi-sequential data lattices connected by an edge. (b) multi-sequetial data lattices connected by two edges.

When there is one adjacent side that exists between two groups of multi-sequential data lattices, the complex data lattices are divided according to the multi-sequential data lattices. For example, in Fig. 6(a), the non-happen-before data points between event e and event f can be determined as the dividing points. Moreover, if there are two sides connected crossly, i.e., there exist multiple ways to get from one to another, for example, Fig. 6(b) can be determined by non-happen-before data points between event s and event t, as well as the non-happen-before data points between event n and p, then the binary sequences are listed by dividing points, and

the binary sequence groups can be used to represent complex data lattices.

Definition 6 (Binary Sequence Groups): Based on the above discussion, we list binary sequences by dividing points in the data lattice, then define the structure

$$\left\{\begin{matrix} d_{k1}^{(i)}, d_{k2}^{(i)}, \cdots, d_{kp}^{(i)} \\ d_{k1}^{(j)}, d_{k2}^{(j)}, \cdots, d_{kq}^{(j)} \end{matrix}\right\} \left\{\begin{matrix} d_{k(p+1)}^{(i)}, \cdots, d_{km}^{(i)} \\ d_{k(q+1)}^{(j)}, \cdots, d_{kn}^{(j)} \end{matrix}\right\},$$
$$\left\{\begin{matrix} d_{k1}^{(i)}, d_{k2}^{(i)}, \cdots, d_{kr}^{(i)} \\ d_{k1}^{(j)}, d_{k2}^{(j)}, \cdots, d_{ks}^{(j)} \end{matrix}\right\} \left\{\begin{matrix} d_{k(r+1)}^{(i)}, \cdots, d_{km}^{(i)} \\ d_{k(s+1)}^{(j)}, \cdots, d_{kn}^{(j)} \end{matrix}\right\}, \cdots$$

consisting of cases which are required by dividing points, each case corresponds to several possible data streams, when we need to get all sequences, the binary sequence groups with all cases can describe them.

The binary sequence in Fig. 6(a) has the following three cases when we determine the non-happen-before data points between event e and event f as the dividing point:

$$\left\{\begin{matrix} de \\ a \end{matrix}\right\} \left\{\begin{matrix} f \\ bc \end{matrix}\right\}, \left\{\begin{matrix} de \\ ab \end{matrix}\right\} \left\{\begin{matrix} f \\ c \end{matrix}\right\}, \left\{\begin{matrix} de \\ abc \end{matrix}\right\} f$$

The binary sequence in Fig. 6(b) has the following four cases when we determine the non-happen-before data points between event s and event t, data point between event n and event p as the dividing points:

$$\left\{\begin{matrix} rs \\ mn \end{matrix}\right\} \left\{\begin{matrix} t \\ p \end{matrix}\right\}, \left\{\begin{matrix} rs \\ m \end{matrix}\right\} \left\{\begin{matrix} t \\ np \end{matrix}\right\}, \left\{\begin{matrix} r \\ mn \end{matrix}\right\} \left\{\begin{matrix} st \\ p \end{matrix}\right\}, \left\{\begin{matrix} r \\ m \end{matrix}\right\} \left\{\begin{matrix} st \\ np \end{matrix}\right\}$$

In practice, data lattice composed of $\{d_{k1}^{(i)} d_{k2}^{(i)} d_{k3}^{(i)} \ldots d_{km}^{(i)}\}$ and $\{d_{k1}^{(j)} d_{k2}^{(j)} d_{k3}^{(j)} \ldots d_{kn}^{(j)}\}$ may generate many possible sequences due to the "happen-before" relation of the concurrent data. However, the traditional method can only observe one sequence in the multi-sequential data lattice and complex data lattice by a single perspective, ignoring a large amount of effective information between data items. Using binary sequence to record multi-sequential data lattice and complex data lattice, the original complete data items information can be retained.

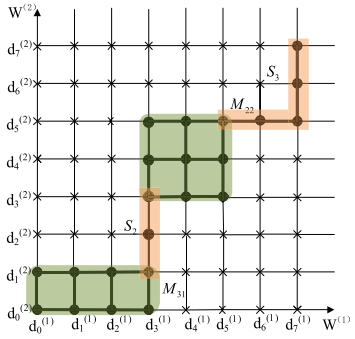

FIGURE 7. A oy data lattice.

In Fig. 7, the sequence of single-sequential data lattice $S_2 = \{d_2^{(2)} d_3^{(2)}\}$ is the unique sequence, similarly, $S_3 = \{d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}\}$. However, multi-sequential data lattice M_{31} corresponds to four sequences, such as $\{d_1^{(2)} d_1^{(1)} d_2^{(1)} d_3^{(1)}\}$, $\{d_1^{(1)} d_1^{(2)} d_2^{(1)} d_3^{(1)}\}$, $\{d_1^{(1)} d_2^{(1)} d_1^{(2)} d_3^{(1)}\}$, $\{d_1^{(1)} d_2^{(1)} d_3^{(1)} d_1^{(2)}\}$. We use the binary sequence to describe the occurrences of the multi-sequential data lattice. For example, for multi-sequential data lattice M_{31}, M_{22}, we will have the following:

$$M_{31} = \left\{\begin{matrix} d_1^{(2)} \\ d_1^{(1)}, d_2^{(1)}, d_3^{(1)} \end{matrix}\right\}, \quad M_{22} = \left\{\begin{matrix} d_4^{(2)}, d_5^{(2)} \\ d_4^{(1)}, d_5^{(1)} \end{matrix}\right\}.$$

In this paper, single-sequential data lattice and multi-sequential data lattice appear alternately, we transform the GDL structure into a sequence of the single-sequential data lattice and multi-sequential data lattice. Then we can get an effective parallel and serial candidate data stream. So we propose to do parallel frequent episodes mining first, then serial frequent episodes mining. Such steps ensure that all frequent episodes are mined in our job. The two-source data stream conversion process is described in Algorithm 2.

Algorithm 2 GDLtoSeq

Input: GDL: a binary file constituted by 0 and 1
Output: S: Sequence
1 for each Node p do
2 Subscript ← Subscript +1;
3 Superscript ← Superscript +1;
4 if (Subscript =1 || Superscript =1)
5 p ∈ $S_{(n-1)}$;
6 i ← min(Subscript), j ← min(Superscript);
7 r ← max(Subscript), t ← max(Superscript);
8 S = $G_{ij} \rightarrow G_{rt}$;
9 if (Subscript =0 || Superscript =0)
10 p ∈ $M_{(m-1)(n-1)}$;
11 S = $M_{(m-1)(n-1)}$;
12 return S;

Definition 7 (Global Data Point): Given a global data point G, we say that G corresponds to a happen-before data point or non-happen-before data point, if G' occurs after G, denoted $G \rightarrow G'$.

IF $G_{ij} = [d_i, d_j]$ and $G_{kj} = [d_k, d_j]$, $G_{ij} \rightarrow G_{kj} = [d_j, d_i, d_k]$.

IF $G_{ij} = [d_i, d_j]$ and $G_{ik} = [d_i, d_k]$, $G_{ij} \rightarrow G_{ik} = [d_i, d_j, d_k]$.

In the actual system, due to the existence of concurrent and message delay, there will be a variety of possibilities in the combination of the data stream, we can divide the sequence by single-sequential or multi-sequential, and reduce the space-time overhead for the frequent episodes mining. For example, the sequence $G_{00} \rightarrow G_{77}$ can be expressed as:

$$d_0^{(1)} d_0^{(2)} \left\{\begin{matrix} d_1^{(2)} \\ d_1^{(1)} d_2^{(1)} d_3^{(1)} \end{matrix}\right\} d_3^{(1)} d_2^{(2)} d_3^{(2)} \times \left\{\begin{matrix} d_4^{(2)} d_5^{(2)} \\ d_4^{(1)} d_5^{(1)} \end{matrix}\right\} d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}.$$

To ensure that all frequent episodes are mined, we mine the frequent episodes just in the multi-sequential data lattice as

the new candidate episodes, after that, we decompose multi-sequential data lattices by Cartesian product. At last, four data streams are mined to get frequent episodes without binary sequences, such as:

(1) $d_0^{(1)} d_0^{(2)} d_1^{(2)} d_3^{(1)} d_2^{(2)} d_3^{(2)} d_4^{(2)} d_5^{(2)} d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}$;

(2) $d_0^{(1)} d_0^{(2)} d_1^{(2)} d_3^{(1)} d_2^{(2)} d_3^{(2)} d_4^{(1)} d_5^{(1)} d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}$;

(3) $d_0^{(1)} d_0^{(2)} d_1^{(1)} d_2^{(1)} d_3^{(1)} d_3^{(1)} d_2^{(2)} d_3^{(2)} d_4^{(2)} d_5^{(2)} d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}$;

(4) $d_0^{(1)} d_0^{(2)} d_1^{(1)} d_2^{(1)} d_3^{(1)} d_3^{(1)} d_2^{(2)} d_3^{(2)} d_4^{(1)} d_5^{(1)} d_6^{(1)} d_7^{(1)} d_6^{(2)} d_7^{(2)}$.

Especially, if we instantiate the sequence $G_{00} \rightarrow G_{77}$ as AB $\left\{ \begin{matrix} E \\ DCA \end{matrix} \right\}$ CA $\left\{ \begin{matrix} BE \\ DA \end{matrix} \right\}$ BCAA, we regard the parallel episode $\left\{ \begin{matrix} E \\ D \end{matrix} \right\}$ as the new candidate episode X. The original data stream can be transformed into the parallel candidate data stream, (1) ABXCAXBCAA. Then, we decompose the original data stream by Cartesian product as serial candi-date data stream, (2) ABECABEBCAA, (3) ABECADABC AA, (4) ABDCACABEBCAA, (5) ABDCACADABCAA. In the end, we got five parallel and serial candidate data stream, we can mine the frequent episodes in turn from these data streams.

V. DATA STREAM FREQUENT EPISODE MINING

According to the data streams in Section 4, frequent episode mining [31] is to find the episodes from the parallel and serial candidate data stream. Frequent episode mining is mainly divided into serial episodes mining and parallel episodes mining. Mixed episodes mining is a combination of existing serial frequent episodes and parallel frequent episodes. Besides, for a fast-growing sequence data stream, old episodes may become obsolete while new useful episodes keep emerging. More importantly, in time-critical applications [32], we need a fast solution to discover the latest frequent episodes from a growing data stream. To this end, we have proposed a streaming frequent episode mining solution.

A. MINING FRAMEWORK

Related research on frequent episode mining based on multi-source data streams scenarios, we build the Frequent Episode Tree (FET) by using a variable sliding window and minimal occurrence. The main idea is to realize the efficient management by treating the data stream continuously synthesized from the source as discrete segments in the time series. With creating and updating the FET, the candidate data stream can be mined to obtain parallel episodes, which have better performance in terms of data storage, update, and memory consumption.

First of all, we set some parameters and definitions:

(1) *Sequence:* Let S be the finite set of events with the strict chronological order, denoted as S=$\{E_1\text{-}t_1, E_2\text{-}t_2, \ldots, E_k\text{-}t_k\}$.

(2) *Episode:* An episode is defined as a non-empty totally ordered set, denoted $\alpha = \{E_1 ldots E_k\}$.

(3) *L, the Length of the Window:* Only consider the last L timestamps in the frequent episode mining.

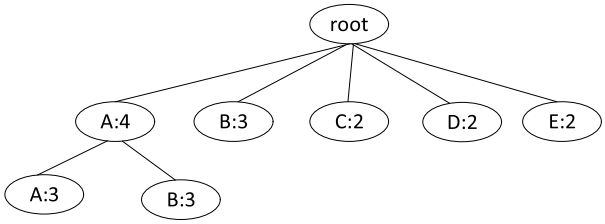

FIGURE 8. Frequent Episode Tree. build the corresponding FET based on the data stream ABECADABCAA, the nodes of the tree are made up of event: support.

(4) δ, *the Maximum Occurrence Window Threshold:* The episode must have occurred in the window δ.

(5) *Minimal Occurrence:* Given two time windows, $[t_1, t_2]$ and $[t_1', t_2']$. $[t_1, t_2]$ is subsumed by $[t_1', t_2']$ if $t_1' \leq t_1, t_2 \leq t_2'$, we define the minimal occurrence window of α as $(\alpha, [t_1, t_2])$.

(6) *Last Occurrence:* Given time window $[t_i, t_j]$, episode occurrence $(\alpha, [t_1, t_2])(t_i \leq t_1 \leq t_2 \leq t_j)$ is the last in the time window α if and only if there is no other occurrence of $(\alpha, [t_1', t_2'])(t_i \leq t_1' \leq t_2' \leq t_j, t_1' > t_1)$.

(7) *Support:* The number of occurrences of episode α on the event sequence S is called the support of α, denoted as sup.

(8) *Frequent Episode:* An episode is called frequent, if and only if its support is no less than min_sup, a user-specified minimum support threshold. Otherwise, the episode is infrequent.

Secondly, we know the FET is a root tree, each node represents an event. Each node can have child nodes. The data stream from the root node to the end represents an episode consisting of several events. The following attributes are maintained in the node, event type E, the support threshold of the node, and the timestamp of the event occurrence location. The growth of the FET is expanded layer by layer. The position of each node is used to divide the data stream, which narrows the search space. Its typical structure of the serial candidate data stream ABECADABCAA is shown as Fig. 8.

For fast-growing sequence data, old data stream may be outdated, and new useful data stream is constantly emerging, we can use the variable sliding window to discover the latest frequent episodes in the sequence. The variable sliding window is similar to the traditional window model, focusing on the last occurrence. The initial size is specified by the user, and then the window size can be dynamically adjusted according to the mining results. When the data item arrives, the data is inserted into the FET through the algorithm of the FET tree, and then the algorithm is used to mine the tree and the concept drift detection is performed on the mining result. If a concept drift occurs, delete the data before the checkpoint and shrink the window size, otherwise continue to insert data, and the window continues to increase with the increase of data. In the variable window, each FET has a root node, which is an empty node, and the child node composed of a data item that arrives in the data stream. With this data structure, each node in the FET (except the root node) represents an episode.

Algorithm 3-1 BuildTree

Input: s: Sequence, min_sup: minimum support threshold, $\delta\delta$: maximum occurrence window threshold
Output: root: Frequent Episode Tree pruning
1 root ← createNode();
2 for each Event e in Find1_episode(s) do
3 node ← createNode(e);
4 add childNode node to root;
5 for each Node n in childNode do
6 expandTree(n);
7 return root;

Algorithm 3-2 Find 1_Episode

Input: s: Sequence, min_sup: minimum support threshold
Output: res: frequent 1- episode
1 res ← null;
2 alter ← null;
3 for each Event e in s do
4 add e to alter;
5 sup ← sup +1;
6 for each Event e in alter do
7 if (sup >= min_sup)
8 add e to res;
9 return res;

Algorithm 3-3 expandTree

Input:p: start position
Output:T_t: expand tree;
1 alter ← null;
2 for each StartPosition p do
3 for each Event e in δ
4 if(e = false)
5 e ← true;
6 add e,p to alter;
7 if(alter = false)
8 sup ← sup +1;
9 if (alter = null)
10 return T_t;
11 for each EventItem e in alter do
12 if(sup > min_sup)
13 n ← createNode(e);
14 add childNode n to p;
15 expandTree(n);

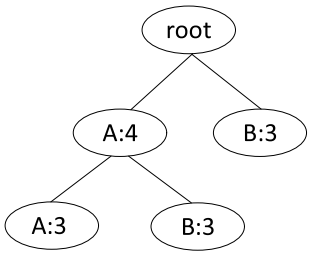

FIGURE 9. Prune FET.

Further, by defining the same node table, the process of episode iteration is transformed into the pruning and scanning process of the FET, which improves the generation efficiency. Therefore, FET describes the data stream information in the current time window, which is convenient for mining frequent episodes in the future.

When new data arrives, in the first step, the sliding window L moves forward, introducing new components; in the second step, we transfer the new component to δ, generate FET and determine the node type.

In the first step, FET construction and pruning steps are as follows, scan the sequence and find all the frequent 1-episode, and add the frequent 1-episode to the first-level nodes of the episode tree. For all frequent n-episode, follow-up events are probed in the sequence according to δ. If the min_sup can be satisfied and the event is not used by other frequent episodes, create a child node for the node and set the event's use flag to true. Repeat until you cannot construct a new node. In this process, there is no need to generate candidate episodes. The construction of the FET is shown in Algorithms 3-1 to 3-3.

For the sequence of events given in Fig. 8, the support for the episodes C, D, and E is 2, when we prune the FET, we can set the support threshold and scan interval, such as min_sup = 2, $\delta = 2$, the pruning result is as shown in Fig. 9.

By analyzing and summarizing the nature of FET, this method has the following advantages:

(1) Save memory. Compared with the traditional method to manage the data stream in the frequent episode mining, FET can make full use of the characteristics of the prefix tree to compress memory;

(2) States update efficiently. FET uses its structural advantages to extend the last minimum occurrence node, reduce redundancy, and achieve efficient state update.

(3) Adapt well to different processing models. The binary sequence-structure generated by multi-source data stream synthesis can still be processed by the prefix tree model.

(4) Accuracy. Since the data items are updated with time, we establish the FET to save the last frequent episodes, and will not pruning them frequently. We can ensure that the frequent episodes excavated are accurate, including serial frequent episodes and parallel frequent episodes.

(5) Complement. We establish a global data lattice by describing the successive relationships between multi-source data streams, and completely retain all logical relationships and causal relationships between data items. On this basis, mining is performed to ensure that complete frequent episodes are obtained.

B. FET COMPLEXITY ANALYSIS

Let L be the sequence length, e be the event typeset, and FE is the frequent episode set. The number of nodes in the FET is m. It is easy to see from the tree construction process that each time a new frequent episode is mined, a node is added to the FET. Therefore, the value of m is | FE |. The number of

all frequent 1-episodes is n, and the maximum n is | e |. The total number of identical nodes is K, and the maximum K is L / min_sup.

The construction of pruning episode tree is the main time costs of this algorithm. The construction of the FET of the MDSOFE algorithm first generates all frequent 1-episodes with a generation complexity of O(L). The number of nodes in the FET is | FE |, and the detection cost of each node is the number of times the event recorded in the node occurs in the sequence p times the detection maximum interval δ^*| FE | $<$ L, can be derived, avp $<$ L / | FE |, the growing cost of the tree is $O(\delta^*|\text{FE}|^*\text{avp}) < O(\delta^*|\text{FE}|^*\text{L}/|\text{FE}|) = O(\delta\text{L})$, so the time complexity is $O(\delta$L). All nodes in the FET have at most | FE|, so the space overhead for storing nodes is O(| FE |).

VI. ALGORITHM MDSOFE

We will introduce serial frequent episodes and parallel frequent episodes obtained by traversing the GDL and the FET by the Multi-source Data Stream Online Frequent Episode Mining (MDSOFE) algorithm in this section.

A. DISCOVERING FREQUENT EPISODES

Considering the parallel relationship in the data stream, it still exists in the results of frequent episode mining, and the mining of parallel episodes is based on the binary sequence. According to the Algorithm 4, there are altogether four steps for getting frequent episodes, 1) build the GDL; 2) get the parallel and serial candidate data stream; 3) mine the parallel and serial episodes; 4) get the mixed episodes. The concrete steps of mining frequent episodes are as follows:

1) STEP 1: BUILD THE GDL

Through Algorithm 1, the sequence ''happen-before'' relation between different data sources is retained, and the GDL is established.

2) STEP 2: GET THE PARALLEL AND SERIAL CANDIDATE DATA STREAM

Through the Algorithm 2, the traversal of GDL generates an effective parallel and serial candidate data stream.

3) STEP 3: MINE THE PARALLEL AND SERIAL EPISODES

Mine the parallel and serial episodes. Through the Algorithm 3, when the new data arrives, judge whether the minimal occurrence episode support threshold in the detection window is greater than the minimum support threshold, and if there is a threshold greater than the minimum support threshold, the minimum occurrence is the frequent parallel and serial episode; if not, this episode will be pruned to determine the next minimal occurrence.

4) STEP 4: GET THE MIXED EPISODES

We merge the serial episodes and parallel episodes, then we regard all the mixed episodes as the frequent episodes from multi-source data stream mining.

Algorithm 4 MDSOFE Algorithm

Input: L Δ: the size of the window, δ: maximum occurrence window threshold, min_sup: minimum support

Output: FE: frequent episodes contain serial episodes(SE) and parallel episodes(PE)

```
G ← BuildGDL(k, table[i], n);
S ← GDLtoSeq(G);
T_t ← BuildTree(S, min_sup, δ);
for each Node g ∈ T_t do
    α ← ep(g);
     PE ← PE ∪α;
for each Node p ∈ T_t do
    α ← ep(p);
     SE ← SE ∪α;
FE ← SE ∪ PE;
return FE;
```

B. MDSOFE COMPLEXITY ANALYSIS

In short, combined with section III and section V, we analyze the time and space complexity of the MDSOFE algorithm. In section III, the lattice storage space cost is $O(\text{n}^2\text{d}^\text{n})$, and the worst-case time of GDL is $O(\text{n}^3\text{d}^\text{n})$. As far as we know, in the process of the mining, GDL is bounded by the window size δ, we can use $O(\text{n}^2\delta^\text{n})$ and $O(\text{n}^3\delta^\text{n})$ to describe the worst-case space and time cost of the GDL, where δ is much less than d. The worst-case time of the n-source data stream frequent episode mining is $O(\text{n}^3\delta^\text{n}+\delta\text{L})$, and the worst-case space is $O(\text{n}^2\delta^\text{n}+|\text{FE}|)$. From the performance analysis, the window size δ and the number of data stream n can bound the cost of the multi-source data streams frequent episode mining. Thus, the proposed algorithm is useful in a multi-source environment.

VII. EXPERIMENTS

In this section, we first introduce an example of multi-source data stream online frequent episode mining, then we conduct extensive simulations to evaluate the performance of our proposed the algorithm MDSOFE by comparing it with the following two schemes.

- *SDSOFE:* The algorithm SDSOFE uses the traditional observer perspective, and mine the unique data stream according to the principle of ''First Come First Process'', so you cannot do parallel episodes mining.
- *MDSOFE-BF:* The algorithm MDSOFE-BF is a violent method that takes all combinations between data items into account by enumeration, and mines all the combined data streams. Algorithm MDSOFE-BF can be seen as multiple repeated experiments of algorithm SDSOFE, it still does not include parallel episodes mining.

To our best knowledge, we are the first to propose the scheme that exploits both multi-source data stream combinations and mixed frequent episodes mining.

A. DATA PREPARATION

We begin our experiments with a synthetic data stream set and five real-world data stream sets, denoted by Distributed

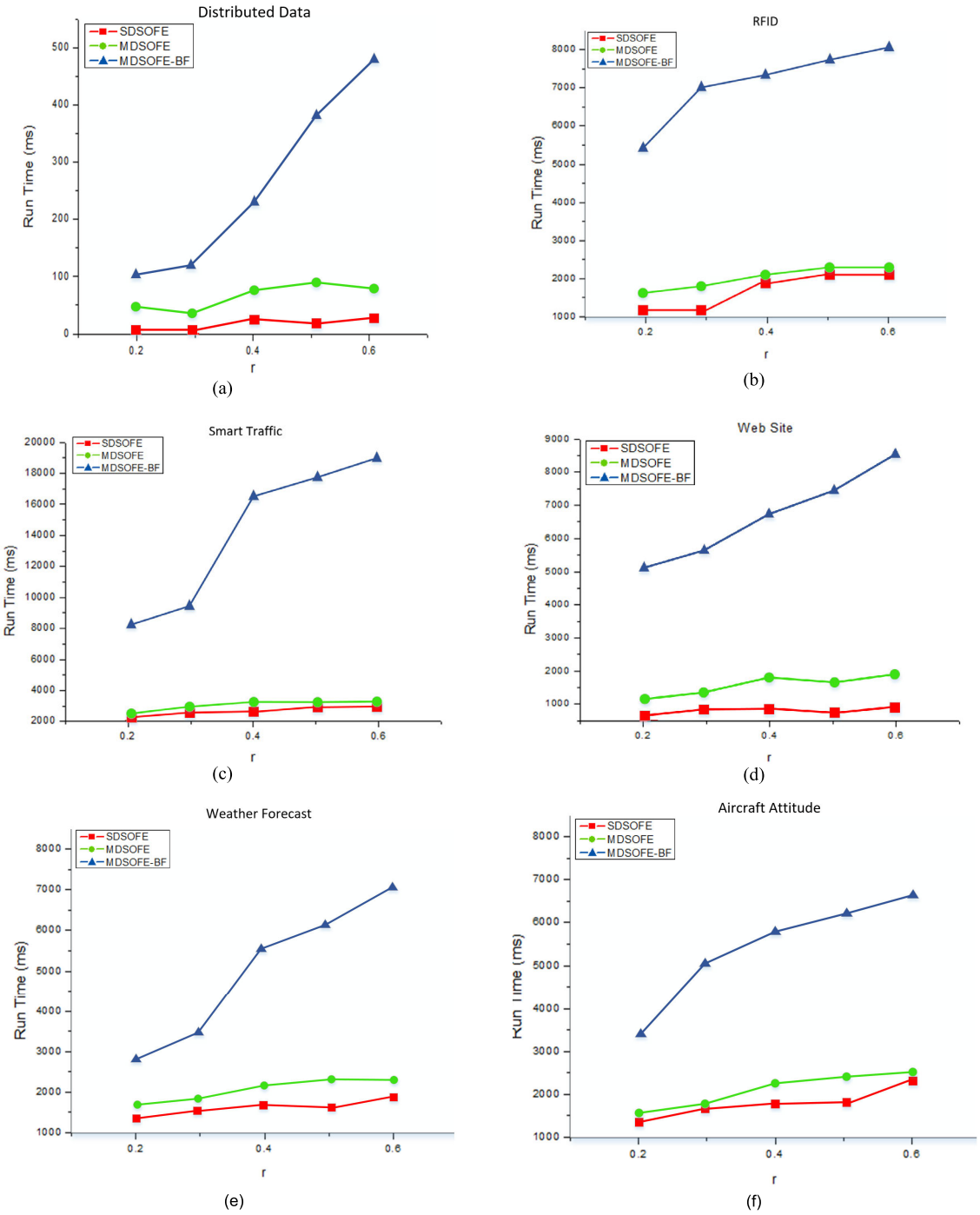

FIGURE 10. Running time with radio r. (a) Running time required in the Distributed Data. (b) Running time required in the RFID. (c) Running time required in the Smart Traffic. (d) Running time required in the Web site. (e) Running time required in the Weather Forecast. (f) Running time required in the Aircraft Attitude.

Data, RFID, Smart Traffic, Web Site, Weather Forecast, and Aircraft Attitude. these data samples can simulate two-source and three-source data streams. The first data stream set Distributed Data (synthetic dataset) consists of two data streams generated by two sources, the two sources can generate new data by combination with each other. The second data stream set records the data of sensors, different sensors record the distinct behavioral information under the office environment, but there exists a relation for a certain action. The third Smart Traffic data stream set comes from the Regional Transportation Management Center (RTMC), the real data is collected by sensors on the Twin Cities Metro freeways from the website [33]. And the next dataset acquires the user's preferences and potential needs through the acquisition of multiple

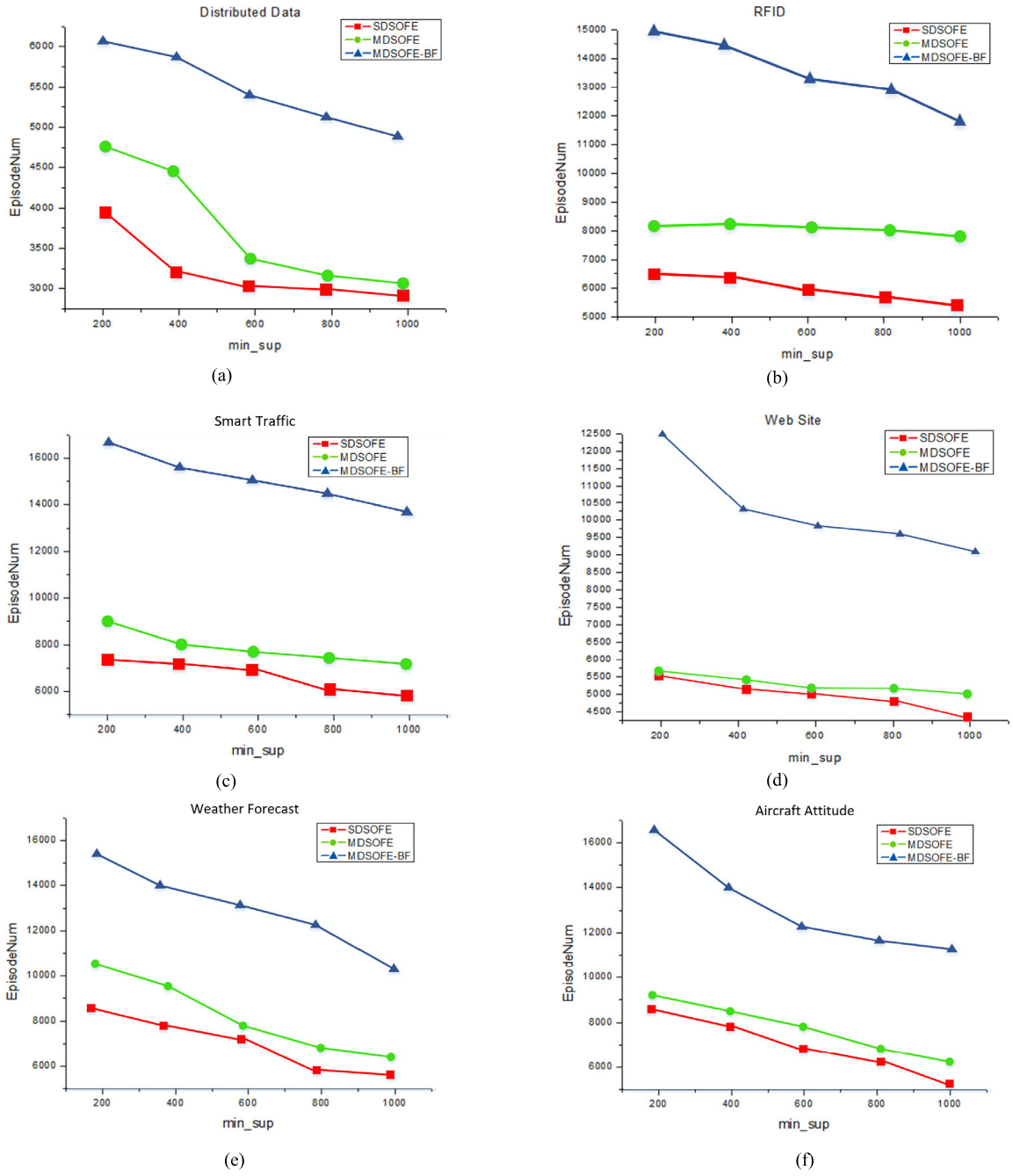

FIGURE 11. Frequent episode number with min_sup. (a) Frequent episode number required in the Distributed Data. (b) Frequent episode number required in the RFID. (c) Frequent episode number required in the Smart Traffic. (d) Frequent episode number required in the Web site. (e) Frequent episode number required in the Weather Forecast. (f) Frequent episode number required in the Aircraft Attitude.

information such as user registration information, online behavior and social relation, this data stream set is downloaded from the data mining portal [34]. The Weather Forecast dataset provided by Climate Observation Association of NWPU includes different dimensions of meteorological features such as temperature, humidity, and sunshine duration to analyse and predict weather. The last dataset is offered from the Laboratory of Aeronautics College of NWPU, including several properties of aircraft which determine the attitude when flying. The data information of all the data stream sets are listed in Table 2, we use the random sampling method to set a certain number of the binary sequence in the GDL according to the ratio, and generate the data with different degrees of complexity relationship.

B. ALGORITHM COMPARISON

We compared the algorithm SDSOFE, MDSOFE-BF and our algorithm MDSOFE. The introduction of the recall radio and

TABLE 2. Data information.

Data Stream	Data Scale	Sequence Number	Average Sequence
Distributed Data	308M	6000	5.2
RFID	1009M	18000	15.1
Smart Traffic	1152M	20000	17.6
Web Site	825M	15000	12.6
Weather Forecast	600M	9000	7.3
Aircraft Attitude	913M	17000	16.6

TABLE 3. Recall radio.

Algorithms	Sequence Number			
	200	300	400	500
SDSOFE	0.5008	0.5983	0.5742	0.7284
MDSOFE	0.9988	0.9901	0.9892	0.9996
MDSOFE-BF	0.9995	0.9992	0.9990	0.9998

TABLE 4. Precision radio.

Algorithms	Sequence Number			
	200	300	400	500
SDSOFE	0.6842	0.5733	0.5977	0.7604
MDSOFE	0.2065	0.3658	0.3006	0.3724
MDSOFE-BF	0.9886	0.9973	0.9943	0.9803

the precision radio, so that we can compare the three algorithms with each other. The multi-source observer considers the order of the data items, which reduced data redundancy in binary sequence group, but contain complete information, so frequent episode is accurate and complete by algorithm MDSOFE in the multi-source method.

Recall radio = accurate episode in other method/ all the frequent episodes in the multi-source method.

Precision radio = accurate episode in other method/ all the frequent episodes in another method.

To evaluate the practicability of the algorithm, we randomly generate data items with the "happen-before" relation, through the window sliding, data arrived, the formation of "streaming" form. We select 200, 300, 400,500 data items in the Distributed Data set. It is concluded that the recall radio and the precision radio of the frequent episodes by the MDSOFE algorithm are both 1 and the recall radio of the frequent episodes by method SDSOFE is between 50% and 70%, they do not contain accurate and complete information, as shown in TABLE 3 and TABLE 4. The violent algorithm MDSOFE-BF mines all meaningful and meaningless episodes, so the frequent episode recall radio is higher, the precision radio is low, only 20%-30%.

The rate of multi-sequential data lattice length to the total is denoted as r. Fig. 10 compares the running time of the algorithms SDSOFE, MDSOFE, and MDSOFE-BF for the four data stream sets along with the increase of radio r. It is shown that the running time of the algorithms SDSOFE and MDSOFE change a little along with the increase of radio r, while the algorithm MDSOFE-BF has a small amount of time when r is small, that is, the complexity of the "happen-before" relation is low, when the complexity increases, the

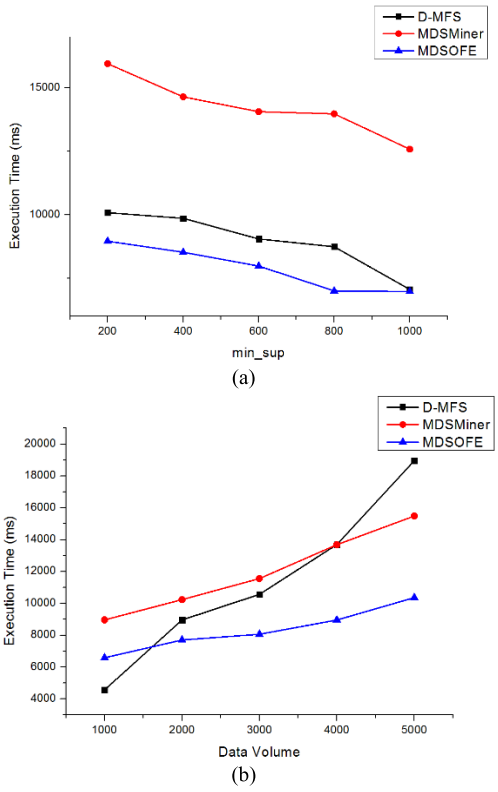

FIGURE 12. Execution Time with min_sup and Data Volume. (a) Execution Time required along with the increase of min_sup. (b) Execution Time required along with the increase of Data Volume.

running time of the algorithm MDSOFE-BF grows faster, at the same time, the algorithms SDSOFE, MDSOFE have high implementation efficiency. Fig. 11 shows that the number of frequent episodes about the algorithms SDSOFE and MDSOFE decrease rapidly with the increase of the minimum support degree. The algorithm MDSOFE-BF mines frequent episodes but contains a large number of redundant episodes.

Based on the above analysis, considering the accuracy and completeness of the information, the algorithm MDSOFE is superior to the algorithms SDSOFE and MDSOFE-BF. Therefore, the MDSOFE satisfies the condition of the multi-source data stream frequent episodes mining algorithm. That is, for a multi-source distributed environment of massive data, the multi-source data stream online frequent episodes mining algorithm has the same or better performance than the existing algorithms that all data are concentrated into a single source computer system.

Several methods for data stream frequent episode mining, namely D-FMS, MDSMiner [35], are compared in global frequent episode mining. The D-FMS uses a three-level engine, which is divided into a first-level front engine, a second-level middle engine, and a third-level back engine, according to the flow direction of the data stream. The data stream is reduced by level, the three-layer architecture of D-FMS forms a multilayer filtering function for data. For a single source stream, the direction of data flow is consistent, and the data

traffic is attenuated after being processed by the primary site. The MDSMiner proposes the concept of alternative support for discovering frequent and rare episodes, and defines the semantic similarity of event sequences for analyzing the relationships between data streams.

Here, the proposed the MDSOFE performs to process a multi-source data stream. Based on ensuring that global frequent sequences can be mined, our algorithm can also obtain parallel frequent episodes. Next, we will compare the support threshold and the transaction data volume in the Distributed Data set by these three algorithms in Fig. 12.

VIII. CONCLUSION

The work of the multi-source data stream online frequent episode mining work combines multiple data sources, retains the interactive information between data streams, and then mines the frequent episodes, streamlining complex and redundant data streams to obtain effective information. This research can be widely used in the field of smart transportation and sensors. By collecting simple and effective information, the hidden associations between roads can be obtained, then we can improve the road construction.

ACKNOWLEDGMENT

The authors would like to thank all researchers who provided guidance and suggestions for this article. They would also like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] H. Tao, J. Zhou, and S. Liu, "A survey of network security situation awareness in power monitoring system," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI2)*, Beijing, China, Nov. 2017, pp. 1–3.

[2] A. Ng and A. W.-C. Fu, "Mining frequent episodes for relating financial events and stock trends," in *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science). 2003, pp. 27–39.

[3] S. Krishnaswamy, J. Gama, and M. M. Gaber, "Mobile data stream mining: From algorithms to applications," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Bengaluru, India, Jul. 2012, pp. 360–363.

[4] B. Li, W. Cui, and B. Wang, "A robust wireless sensor network localization algorithm in mixed LOS/NLOS scenario," *Sensors*, vol. 15, no. 9, pp. 23536–23553, Sep. 2015.

[5] S. Seshadri, V. Kumar, B. Cooper, and L. Liu, "A distributed stream query optimization framework through integrated planning and deployment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 10, pp. 1439–1453, Oct. 2009.

[6] A. V. Dinh, R. J. Palmer, R. J. Bolton, and R. Mason, "Multichannel multipoint distribution service system synchronization using global positioning system clock," in *Proc. Can. Conf. Electr. Comput. Eng. Conf. Proc. Navigating New Era*, Halifax, NS, Canada, vol. 2, May 2000, pp. 875–879.

[7] R. Gross and R. Melkers, "Clock management data analysis for satellite communications," in *Proc. IEEE Int. Freq. Control Symp. Expo.*, Vancouver, BC, Canada, Aug. 2005, p. 7.

[8] A. D. Kshemkalyani and J. Cao, "Predicate detection in asynchronous pervasive environments," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1823–1836, Sep. 2013.

[9] Y. Yang, Y. Huang, J. Cao, X. Ma, and J. Lu, "Design of a sliding window over asynchronous event streams," *Comput. Sci.*, vol. 25, no. 10, pp. 2551–2560, 2011.

[10] Y. Huang, Y. Yang, J. Cao, X. Ma, X. Tao, and J. Lu, "Runtime detection of the concurrency property in asynchronous pervasive computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 4, pp. 744–750, Apr. 2012.

[11] D. M. Amitu, "Maximizing data gathering in mobile wireless sensor networks," in *Proc. IEEE Conf. Wireless Sensors (ICWiSE)*, Langkawi, Malaysia, Oct. 2016, pp. 1–6.

[12] B.-D. Lee and K.-H. Lim, "An adaptive data reporting scheme considering node contexts in wireless sensor networks," in *Proc. 1st ACIS/JNU Int. Conf. Comput., Netw., Syst. Ind. Eng.*, Jeju Island, South Korea, May 2011, pp. 382–386.

[13] Y. Yue, H. Fan, J. Li, and Q. Qin, "Large-scale mobile wireless sensor network data fusion algorithm," in *Proc. IEEE Int. Conf. Big Data Anal. (ICBDA)*, Hangzhou, China, Mar. 2016, pp. 1–5.

[14] N. Tatti and B. Cule, "Mining closed episodes with simultaneous events," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, San Diego, CA, USA, 2011, pp. 1172–1180.

[15] W. Cho, M.-S. Gil, S. Lee, and Y.-S. Moon, "A storm-based sampling system for multi-source stream environment," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Shanghai, China, Jan. 2018, pp. 503–506.

[16] T. Kawakami, Y. Ishi, T. Yoshihisa, and Y. Teranishi, "An evaluation of load distribution method on multi-source P2P sensor data stream delivery system," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Feb. 2014, pp. 486–491.

[17] Z. Tong, H. Liao, and X. Jin, "A real-time frequent pattern mining algorithm for semi structured data streams," in *Proc. IEEE 2nd Int. Conf. Big Data Anal. (ICBDA)*, Beijing, China, Mar. 2017, pp. 269–275.

[18] C.-X. Meng, "An efficient algorithm for mining frequent patterns over high speed data streams," in *Proc. WRI World Congr. Softw. Eng.*, Xiamen, China, May 2009, pp. 319–323.

[19] J. H. Chang and W. S. Lee, "Finding recent frequent itemsets adaptively over online data streams," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 487–492.

[20] Y. Yamamoto and K. Iwanuma, "Online pattern mining for high-dimensional data streams," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Santa Clara, CA, USA, Oct. 2015, pp. 2880–2882.

[21] C. K.-S. Leung, F. Jiang, and Y. Hayduk, "A landmark-model based system for mining frequent patterns from uncertain data streams," in *Proc. 15th Symp. Int. Database Eng. Appl. (IDEAS)*, 2011, pp. 249–250.

[22] C. Xu, Y. Chen, and R. Bie, "Sequential pattern mining in data streams using the weighted sliding window model," in *Proc. 15th Int. Conf. Parallel Distrib. Syst.*, Dec. 2009, pp. 886–890.

[23] Y. He and M. Yue, "Parallel frequent itemset mining on streaming data," in *Proc. 10th Int. Conf. Natural Comput. (ICNC)*, Xiamen, China, Aug. 2014, pp. 725–730.

[24] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," Tech. Rep., 2002.

[25] R. Motwani, G. S. Manku, "Approximate frequency counts over data streams," in *Proc. 28th Int. Conf. Very Large Data Bases*, 2002, pp. 346–357.

[26] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," Tech. Rep., 2004.

[27] F. Mattern, "Virtual time and global states of distributed systems," in *Proc. Int. Workshop Parallel Distrib. Algorithms*, Holland, MI, USA, 1989, pp. 215–226.

[28] E. Atoofian and A. G. Bavarsad, "AGC: Adaptive global clock in software transactional memory," in *Proc. Int. Workshop Program. Models Appl. Multicores Manycores ACM*, 2012, pp. 11–16.

[29] L. Lamport and C. Time, "The ordering of events in a distributed system," *Commum. ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[30] G. A. Grätzer, *General Lattice Theory* 2nd ed. Basel, Switzerland: Birkhäuser, 2003.

[31] L. Wan, L. Chen, and C. Zhang, "Mining frequent serial episodes over uncertain sequence data," in *Proc. Int. Conf. Extending Database Technol.*, 2013, pp. 215–226.

[32] Z. Zhou, Z. Peng, J.-H. Cui, and Z. Shi, "Efficient multipath communication for time-critical applications in underwater acoustic sensor networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 28–41, Feb. 2011.

[33] [Online]. Available: http://www.d.umn.edu/~tkwon/TMCdata/TMCarchive.html

[34] [Online]. Available: https://www.kdnuggets.com/datasets/index.html

[35] Z. Hu, W. Liu, and H. Wang, "Mining both frequent and rare episodes in multiple data streams," in *Proc. 10th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Shenyang, China, Jul. 2013, pp. 753–761.

TAO YOU was born in Henan, China, in 1983. He received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2011.

In 2010, he was a Visiting Researcher with the Department of Computer, Purdue University. He is currently an Associate Professor and a Master's Supervisor. His main research interests include real-time distributed computing and data mining, and complex networks.

Dr. You's awards and honors include the Prize for National Defense Science and Technology, the Science and Technology Prize of Shaanxi Provincial Education Department, and the Prize for Xi'an Science and Technology Progress.

YAMIN LI was born in Henan, China, in 1994. She received the bachelor's degree in computer science and technology, in 2017. She is currently pursuing the master's degree with the Department of Computer Science, Northwestern Polytechnical University, Xi'an, China.

She holds one patent. Her research interest includes the development of data stream mining.

Ms. Li's awards and honors include the Northwestern Polytechnical University Award.

BINGKUN SUN was born in Xi'an, China, in 1995. He received the bachelor's degree in engineering automation from Xi'an Jiaotong University, in 2018. He is currently pursuing the master's degree in computer science and technology with Northwestern Polytechnical University, Xi'an.

As a postgraduate of Northwestern Polytechnical University, his research interests include data mining and intelligent transportation systems. His awards and honors include the Northwestern Polytechnical University Award.

CHENGLIE DU was born in Xi'an, China, in 1970. He received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2000.

In 2009, he was a Visiting Researcher with the Department of Computer, Portland State University. He is currently a Professor and a Ph.D. Supervisor. His main academic part-time positions: the Director of the China Computer Automatic Measurement and Control Technology Association, a Senior Member of the China Computer Society, and a member of the System Software Professional Committee and the Software Engineering Professional Committee. His main research interests include real-time distributed computing and embedded computing, including research support for virtual experimental verification system operation support environment, simulation verification of complex embedded systems, and cyber-physical fusion systems.

Dr. Du's awards and honors include commendation for Manned Space Engineering, prizes for aerospace science and technology progress, and prize for national defense science and technology.