Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6662254, 17 pages
https://doi.org/10.1155/2021/6662254

Research Article

Fuzzy Frequent Pattern Mining Algorithm Based on Weighted Sliding Window and Type-2 Fuzzy Sets over Medical Data Stream

Jing Chen,[1,2] **Peng Li**,[3,4] **Weiqing Fang**,[3] **Ning Zhou**,[3] **Yue Yin**,[3] **Hui Zheng**,[3] **He Xu**,[3,4] **and Ruchuan Wang**[3,4]

[1]*School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*Baotou Teachers' College of Inner Mongolia University of Science and Technology, Inner Mongolia, Baotou 014030, China*
[3]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[4]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, Jiangsu Province 210023, China*

Correspondence should be addressed to Peng Li; lipeng@njupt.edu.cn

Received 3 December 2020; Revised 21 March 2021; Accepted 24 March 2021; Published 26 December 2021

Academic Editor: Arun K. Sangaiah

Real-time data stream mining algorithms are largely based on binary datasets and do not handle continuous quantitative data streams, especially in medical data mining field. Therefore, this paper proposes a new weighted sliding window fuzzy frequent pattern mining algorithm based on interval type-2 fuzzy set theory over data stream (WSWFFP-T2) with a single scan based on the artificial datasets of medical data stream. The weighted fuzzy frequent pattern tree based on type-2 fuzzy set theory (WFFPT2-tree) and fuzzy-list sorted structure (FLSS) is designed to mine the fuzzy frequent patterns (FFPs) over the medical data stream. The experiments show that the proposed WSWFFP-T2 algorithm is optimal for mining the quantitative data stream and not limited to the fragile databases; the performance is reliable and stable under the condition of the weighted sliding window. Moreover, the proposed algorithm has high performance in mining the FFPs compared with the existing algorithms under the condition of recall and precision rates.

1. Introduction

Frequent itemset mining (FIM) is the primary processing for the association rule mining (ARM, Apriori 1993) algorithm, which mines associations in the form of rules, such as "IF eat more than regular THEN have obesity," for a dataset. The ARM algorithm has an irreplaceable advantage in real-world applications, especially in medical diagnosis and health prediction. ARM algorithm can mine potential disease-related rules. Also, traceability for disease relevant results, as ARM algorithm can easily find hidden associations between diseases and their related features in a dataset. For this reason, this paper will focus on mining frequent patterns in studying of computer-aided medical diagnosis and health prediction. In a database, the number of times an item appears refers as the item's support and the item is a frequent item (FI) if the items' support exceeds a defined threshold. Apriori [1], Eclat [2, 3], and FPGrowth [4, 5] algorithms are the three most classical algorithms in FIM.

A significant number of improved algorithms, such as AFOPT [6–8], FPGrowth∗ [9–11], CPFGrowth [11], and BFPGrowth [12], emerged after the classic FPGrowth algorithm [4, 5] was proposed. The key step of the association rule mining (ARM) algorithm is the FPM, which can process and analyze the association relationship corresponding to Boolean variables, sequences, or transactions, but cannot directly process the quantitative values represented by floating point numbers [13, 14].

In order to solve the problem, researchers began studying the discretization of the attribute features of continuous data into category attributes to discover the implicit association rules. Lee (1997) introduced fuzzification into the ARM algorithm [15] that the user would influence the process of discovery and obtain the details of rules discovered. The aim

of fuzzy transformation is to convert the quantitative data into fuzzy data.

The ARM has been improved to process the continuous data by implementing new rules or integrating several algorithms, such as incorporating a new "interest" metric [16] and inserting additional rule templates [17], implementing the bottom-up grouping method [18, 19], and introducing genetic algorithm [19, 20]. Moreover, in order to validate the ARM [21–23], the threshold condition could be continuously practiced based on minimum support and confidence.

The approach used above is to directly transform the continuous data into a crisp category. Sharp edges will emerge [24] if the transformed numerical attributes are located near the interval segmentation points, causing the issue of matching the segmentation points and the precision of the algorithm.

The research gap is how to minimize the issue of sharp boundaries; researchers started to implement the fuzzy association rule mining (FARM) algorithm in their research. These methods [25–28] apply the fuzzy theory to expand the clear interval to membership value interval for the mining the fuzzy association rules dealing with quantitative data. The fuzzy sets [29, 30] that correspond to the membership preserve the interval of the numerical attributes in the fuzzification process [31], which increases the amount of information on the transformed numerical data and decreases the sharp boundaries issue.

Hong et al. [32] introduced two different support thresholds instead of conventional thresholds to divide the candidate frequent itemsets into parts of different meanings, thus minimizing the workload of rescanning the initial dataset. The process of fuzzification is to extend the space to the fuzzy space. The main reason for the imbalance of fuzzy sets is that the amount of data in the central fuzzy set obtained by the fuzzy transformation increases, thereby increasing the additional fuzzy membership. Hong et al. [33] proposed a FARM based on the type-1 fuzzy set theory. The type-2 fuzzy set theory [34] was introduced to solve the problem of FARM in the regular database [35]. The methods of literature [32–35] inspired the motivation to solve the fuzzy association rule mining over medical quantitative data stream based on type-2 fuzzy set theory.

To the best of the authors' knowledge, there is no relevant research focused on the weighted sliding window to mine the FFIs over quantitative data stream based on type-2 fuzzy set theory to bridge the understanding between computers and human beings without the expert suggestions. This paper improves the model and proposes a new weighted sliding window quantitative data stream frequent pattern mining algorithm based on type-2 fuzzy set theory.

In this paper, the WSWFFP-T2 algorithm is proposed to efficiently mine the FFIs with one scan over the medical data streams. For the dynamic FFPs mining process, a fuzzy-list sorted structure and a WFFPT2-tree are suggested. The main contributions of this article are as follows:

(1) In order to distinguish the weights of historical and recent transactions, the weighted sliding window with decay factor is adopted that simultaneously overcomes the problem of unexpected concept drift in the number of items in data streams. Recall and precision rates are applied to control the influence of the decay factor on the frequent patterns and the critical frequent patterns of the sliding window

(2) A novel WSWFFP-T2 algorithm over medical data streams is proposed to efficiently mine the FFPs with only one scan using a fuzzy-list sorted structure (FLSS) and WFFPT2-tree. The construction and the deletion strategies are used in the form of a linked list to construct and maintain the WFFPT2-tree. Various examples and figures are provided to better understand the interval type-2 fuzzy sets theory mining process over the quantitative data stream

(3) The proposed WSWFFP-T2 algorithm is compared with the previous related algorithms to evaluate its performance. Several synthetic data streams applying decay factors with sliding window are used in the performance evaluation. The experimental results show that the proposed algorithm presents more outstanding performance than the previous ones

The rest of this paper is organized as follows. Related works and preliminaries are introduced in Sections 2 and 3, respectively. The proposed algorithm is presented in Section 4. An experimental study is illustrated in Section 5. Finally, Section 6 concludes this paper.

2. Related Works and Research Problems

This section briefly reviews the studies concerning sliding window [36] with fading factor, type-2 fuzzy sets theory, and the PF-tree algorithm from the quantitative data streams.

2.1. Sliding Window with Fading Factor. Under some conditions, especially in the data stream, new transactions are much more important than the historical transactions when mining the data streams. In order to distinguish the new and the old transactions and consider the importance of the latest data, the literature MSW [37] and SWP-tree [38] set different weights for the old and the new transactions. Obtaining a more reasonable frequent itemset will produce a large number of redundant patterns. The classical algorithms based on the window model include lossy count [39] and DSM_FT [40]. The most common window model is the sliding window model. For example, the algorithms MSW [38], MFI-CBSW [41], MFI-TimeSW [42], and TMFI [43] use the sliding window models to mine the frequent patterns in the data streams. The classical algorithms based on the attenuation model are estDec [44] and estDec+ [45]. According to the characteristics of the pattern, it can be divided into frequent patterns, frequent sequence patterns, frequent tree patterns, and frequent graph pattern mining methods.

2.2. Interval Type-2 Fuzzy Set Theory. Based on the pattern growth mechanism, Lin et al. [46] used the fuzzy frequent pattern tree method to discover the set of FFIs. When the

transaction volume is large, this method may require a significant amount of calculation. Therefore, the compressed fuzzy frequent pattern (CFFP) tree algorithm [47] has been used to reduce the size of the frequent pattern tree. Compared with the FFP-tree algorithm, the CFFP algorithm significantly reduces the number of tree nodes in the CFFP tree. However, it is necessary to reserve an additional array for each node to store the current processing node and the membership value. Therefore, the CFFP algorithm requires a large of memory to store this information, which is inefficient in a very sparse database. In order to solve this problem, the UBFFPT-tree [48] algorithm not only retains the simplified tree structure but also mines the fuzzy frequent itemsets in limited memory, which can effectively mine the FFIs with the same tree node size as the CFFP-tree algorithm. The above algorithm only uses one language word to represent the item being processed in the database, so the information found may be incomplete and insufficient. The above method uses type-1 fuzzy set theory without considering the uncertainty factor.

This concept has also been widely used in intelligent information systems. In data mining, when dealing with a transaction database with sales quantity information, the idea of fuzzy sets can be used to transform the number of items into language. It uses the semantic terms to express the concepts, which is more in line with human thinking. Fuzzy data mining algorithms have been proposed aiming at language association rules [49] and other issues.

To solve this limitation, the researchers designed and implemented a type-2 fuzzy set theory that considers the uncertainty problem. In 1965, Zadeh [50] proposed type-1 fuzzy set (T1-FS theory) and then the type-1 fuzzy logic system (T1-FLS) based on T1-FS to solve the problem of uncertainty, fuzzy, and inaccurate modeling method. Since the T1-FLS is represented by an accurate membership function, its ability to directly deal with the uncertainty of fuzzy rules is limited. Karnik and Mendel [51] proposed a type-2 fuzzy set (type-2 FS, T2-FS) that blurred the membership degree in the fuzzy set, that is, the membership degree itself was T1-FS, and T2-FS was enhanced. Chen et al. [52] proposed a priori-based standard method to discover FFI with type-2 fuzzy sets. The algorithm used a generation test mechanism based on fuzzy two-category membership functions to find the candidate objects hierarchically.

Interval type-2 fuzzy set [53] is an extension of traditional type-1 fuzzy set. Its essence is to defuzzify the membership value in the fuzzy set. The interval type-2 fuzzy set theory can directly deal with the uncertainty of fuzzy rules. Using the membership function of the second type of fuzzy set theory to mine the FFPs requires the use of a central mechanism to convert the fuzzy value (interval) of language terms into fuzzy values, which requires more time complexity to generate the candidate identification codes. The proposed algorithm adopts type-2 fuzzy set theory.

Type-2 FSs are described by the MFs that are characterized by more parameters than the MFs for type-1 FSs. Hence, the type-2 FSs provide more design degrees of freedom [54]. Therefore, the use of type-2 fuzzy sets has the potential to outperform the use of type-1 fuzzy sets, especially, in uncertain environments.

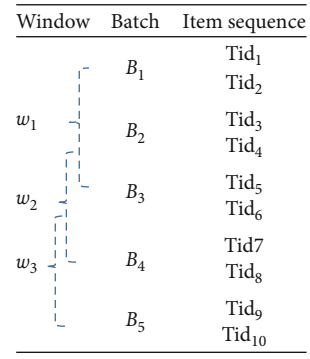

Figure 1: Data list of the data stream.

3. Preliminaries

At present, most FFPs algorithms are based on the framework and algorithm evolution of Apriori [1] and FPGrowth [4]. However, these methods cannot handle the data stream immediately. In the process of the data stream mining, some projects rarely appear as time goes by, but as new transactions continue to increase, they may become more frequent in the future. According to the dynamic changes of the data, the importance of the data itself also changes. Therefore, different data processing models should be adopted according to the changing law of the data. When the importance of the data gradually decreases, the attenuation model should be used for data processing. The important factor in the attenuation model is the attenuation rate factor.

Definition 1 (quantitative data stream (QDS)). Given $I = \{I_1, I_2, \cdots, I_m\}$ is the set of items, the QDS is a sequence $\{T_1, T_2, \cdots, T_n\}$ that arrives continuously at a certain speed represented by quantitative values, where T_i represents the ith transaction $(1 \le i \le k)$, for any given T_i has $T_i \subset I$.

Definition 2 (weighted sliding window [36]). The start and end of the sliding window $w = \{w_1, w_2, \cdots, w_n\}$ are determined by the batch and the current time, and each has its decay factor during the data stream mining process. The window contains 3 batches, and each batch contains two time series. As shown in Figure 1, the first sliding window w_1 contains three batches like B_1, B_2, B_3. The batch B_1 contains two T_1 and T_2. The number of batches in the algorithm implemented in this paper can be adjusted according to the actual situation of the actual data stream, meaning it is a variable dynamic weighted sliding window.

Definition 3 (decay factor [55, 56]). The decay factor δ refers to the attenuation degree of data over time. In the sliding window model where data elapses over time, the knowledge contained in the patterns in the latest window is more important or valuable than the patterns in the historical window. For example, the reference value of the current monitored medical data is larger than in the past, and the monitoring data has a more important reference value for medical diagnosis and prediction.

Window	Batch	Data sequence	Quantitative items
w_1	B_1	T_1	(d:4), (b:1), (e:5)
		T_2	(b:1), (e:3)
	B_2	T_3	(b:2), (f:2), (e:1)
		T_4	(d:1), (b:3), (c:2)
w_2	B_3	T_5	(c:5), (d:5), (a:3), (b:3), (e:4)
		T_6	(c:4), (a:1), (b:4)
w_3	B_4	T_7	(d:4), (b:2)
		T_8	(c:4). (b:4), (f:3)
	B_5	T_9	(c:3), (d:4), (b:2), (f:1)
		T_{10}	(b:5), (f:5)

Figure 2: Quantitative data stream with sliding window.

Definition 4 (frequent pattern). N is the number of current transaction items in the data stream. Let θ be the minimum support threshold. If the pattern P satisfies Equation (1), P is a frequent pattern.

$$\text{freq}(P, N) \geq \theta \times N. \tag{1}$$

Definition 5 (error factor). The error factor ε is the maximum allowable error threshold ($\varepsilon \in (0, \theta)$). If the pattern P satisfies Equation (2), then P is a critical frequent pattern.

$$\theta \times N \geq \text{freq}(P, N) \geq \varepsilon \times N. \tag{2}$$

If $\text{freq}(P, N) < \varepsilon \times N$, then P is an infrequent pattern.

Definition 6 (lingual variable [35]). There is the following relationship $i \subset I$; the value of the lingual variable is determined according to the membership value given by the user.

Example: the membership relationship in Figure 2 can be transformed into the interval represented by $(R_{i1}, R_{i2}, \cdots, R_{ih})$ through the type-2 fuzzy process. For example, Figure 2 shows 10 sequences through which the data stream passes, and the corresponding data items T_1 are $(d:4)$, $(b:1)$, $(e:5)$. The value of i is represented by v_{iq}, so the items in T_1: (b), (d), and (e) are transformed as $v_{d1}(=4)$, $v_{b1}(=1)$, and $v_{e1}(=1)$, respectively.

4. Proposed WSWFFP-T2 Algorithm

The proposed WSWFFP-T2 algorithm consists of four phases for mining the FFPs. The first phase is the data stream input whose value is the object's quantitative data, not the crisp database. The second phase consists of two steps, the weighted sliding window and the fuzzy-preprocessing, based on the interval type-2 fuzzy theory including the original fuzzification and fuzzy compression process. From quantitative data to linguistic value, the membership attribute was transformed and further details would be presented in the second phrase. In order to avoid the problem of concept drift [57], the decay and the error factors are used during the weighted sliding window over the data stream process to generate the frequent patterns and the critical frequent patterns. The third phase is the construct-delete process of WFFPT2-tree with fuzzy-list sorted structure. The WFFPT2-tree-based algorithm for generating FFPs is the last step. Figure 3 presents the structure of the WSWFFP-T2 algorithm.

4.1. Preprocessing Fuzzification

4.1.1. Prefuzzification Based on Interval Type-2 Fuzzy Theory. For the designed WSWFFP-T2 algorithm to mine the FFPs on a data stream, the quantitative attribute should be first fuzzified by the defined membership functions based on the interval type-2 fuzzy theory. The step is called the preprocessing of the fuzzification because the fuzzified data should be compacted after the first step. In the preprocessing of the fuzzification, the quantitative attribute is transformed into a linguistic value (low, middle, and high) as shown in Figure 4. Equations (3)–(5) are the transformed functions that have three groups like $(fv_{\text{low}}^{\text{lower}}, fv_{\text{low}}^{\text{upper}})$, $(fv_{\text{mid}}^{\text{lower}}, fv_{\text{mid}}^{\text{upper}})$, and $(fv_{\text{high}}^{\text{lower}}, fv_{\text{high}}^{\text{upper}})$, and the value of distance "a" can be defined by the user according to the actual demand. For example, the original quantitative data $(T_1 : (d : 4), (b : 1), (e : 5))$ shown in Figure 2 is then transformed into membership attribute in Figure 4, and the fuzzified results are listed in Figure 5.

The membership function construction methods of two interval fuzzy sets are the individual member function method and the interval endpoint method. Figure 4 shows the membership μ of type-2 fuzzy set theory. The abscissa is the size of different transactions, and the ordinate is the degree of membership. The language terms low (L), medium (M), and high (H) are used to denote the division of different intervals. At the same time, the users can formulate expressions of language terms according to the actual application requirements.

$$fv_{\text{low}}^{\text{lower}} = \begin{cases} 1, & X \leq 1, \\ -\dfrac{1}{2}x + \dfrac{3}{2}, & 1 < X \leq 3, \\ 0, & X > 3, \end{cases}$$
$$fv_{\text{low}}^{\text{upper}} = \begin{cases} 1, & X \leq 1, \\ -\dfrac{1}{2+a}x + \dfrac{3+a}{2+a}, & 1 < X \leq 3+a, \\ 0, & X > 3+a, \end{cases} \tag{3}$$

$$fv_{\text{mid}}^{\text{lower}} = \begin{cases} 0, & X \leq 1, \\ \dfrac{1}{2}x - \dfrac{1}{2}, & 1 < X \leq 3, \\ -\dfrac{1}{2}x + \dfrac{5}{2}, & 3 < X \leq 5, \\ 0, & X > 5, \end{cases}$$
$$fv_{\text{mid}}^{\text{upper}} = \begin{cases} 0, & X \leq 1-a, \\ \dfrac{1}{2+a}x + \dfrac{a-1}{2+a}, & 1-a < X \leq 3, \\ -\dfrac{1}{2+a}x + \dfrac{5+a}{2+a}, & 3 < X \leq 5+a, \\ 0, & X > 5+a, \end{cases} \tag{4}$$

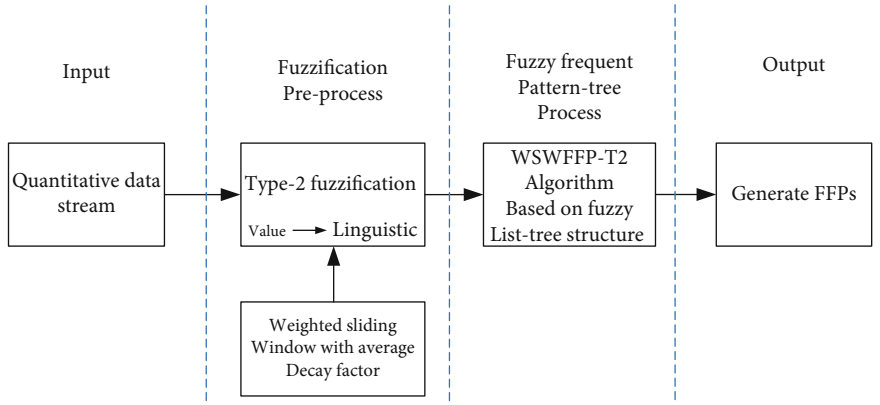

Figure 3: Proposed WSWFFP-T2 Algorithm Structure.

$$fv_{\text{high}}^{\text{lower}} = \begin{cases} 0, & X \le 3, \\ \frac{1}{2}x - \frac{3}{2}, & 3 < X \le 5, \\ 1, & X > 5, \end{cases}$$
$$fv_{\text{high}}^{\text{upper}} = \begin{cases} 0, & X \le 3 - a, \\ \frac{1}{2+a}x + \frac{a-3}{2+a}, & 3 - a < X \le 5, \\ 1, & X > 5. \end{cases} \quad (5)$$

After the prefuzzification of interval type-2 fuzzy set theory, the fuzzified data structure is presented as $(fv.\text{upper}, fv.\text{lower})/\text{Iname.ftp}$, where the Iname represents the name of the item and $(fv.\text{upper}, fv.\text{lower})$ is the membership value calculated from Equations (3)–(5). In order to unify the abbreviation, after the initial fuzzy conversion, the linguistic attributes of ftp (fuzzy type) like low, mid, and high are represented as Iname.L, Iname.M, and Iname.H, respectively.

Definition 7. The linguistic term set is represented as f_{iq}, and the corresponding membership degree or called fuzzy value could have the membership degree relationship of type-2 fuzzy theory which is defined as $f_{iq} = \mu_i(v_{iq})$ in the following:

$$f_{iq}\left(= \frac{\left(fv_{iq1}^{\text{lower}} + fv_{iq1}^{\text{upper}}\right)}{R_{i1}} + \frac{\left(fv_{iq2}^{\text{lower}} + fv_{iq2}^{\text{upper}}\right)}{R_{i2}} + \cdots + \frac{\left(fv_{iqh}^{\text{lower}} + fv_{iqh}^{\text{upper}}\right)}{R_{ih}} \right). \quad (6)$$

For instance, the item (d) in Tid_1 has a corresponding value of 4, which can be converted as follows:

$$\left(\frac{(0.5, 0.63)}{d.M} + \frac{(0.5, 0.63)}{d.H}\right). \quad (7)$$

Therefore, the items $(d:4)$, $(b:1)$, and $(e:5)$ in T_1 are transformed from quantitative data to linguistic terms

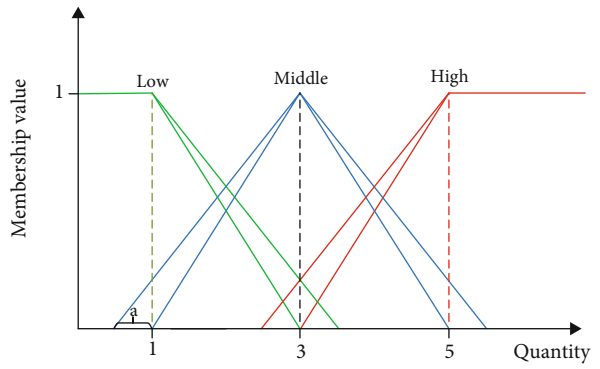

Figure 4: Membership function in type-2 fuzzy set theory.

($d.M$, $d.H$, $b.L$, $b.M$, $e.M$, and $e.H$), and the conversion results of all itemsets in Figure 2 can be indicated in Figure 5 based on the defined split point number.

$$\left(\frac{(0.5, 0.63)}{d.M} + \frac{(0.5, 0.63)}{d.H}, \frac{(1,1)}{b.L} + \frac{(0, 0.25)}{b.M}, \frac{(0, 0.25)}{e.M} + \frac{(1,1)}{e.H}\right). \quad (8)$$

4.1.2. The Fuzzy Compression Process. To reduce the complexity of the preprocessing step, the interval value is represented by the fuzzy compression process as defined in Definition 8. For example, in the transaction T_1 shown in Figure 5, the item (d) with its quantity 4 is then transformed as $(((0.5, 0.63)/d.M) + ((0.5, 0.63)/d.H))$. Then, the interval is transformed as $(0.5 + 0.63)/2 = 0.56$ using the type-reduction method with the defined split point number according to Equation (9). After that, both $d.M$ and $d.H$ are reduced as $((0.56/d.M) + (0.56/d.H))$ in Figure 6.

Definition 8. The membership degree of a linguistic term R_{ih} in the fuzzified dataset D' is denoted as fv_{iqh}^{c} [35] and defined as follows:

$$fv_{iqh}^{c} = \frac{fv_{iqh}^{\text{lower}} + fv_{iqh}^{\text{upper}}}{2}. \quad (9)$$

Window	Batch	Tid	Quantitative data after type-2 fuzzy conversion
	B_1	T_1	$\frac{(0.5,0.63)}{d.M}+\frac{(0.5,0.63)}{d.H},\frac{(1,1)}{b.L}+\frac{(0,0.25)}{b.M},\frac{(0,0.25)}{e.M}+\frac{(1,1)}{e.H}$
		T_2	$\frac{(1,1)}{b.L}+\frac{(0,0.25)}{b.M},\frac{(0,0.25)}{e.L}+\frac{(1,1)}{e.M}+\frac{(0,0.25)}{e.H}$
w_1	B_2	T_3	$\frac{(0.5,0.63)}{b.L}+\frac{(0.5,0.63)}{b.M},\frac{(0.5,0.63)}{f.L}+\frac{(0.5,0.63)}{f.M},\frac{(1,1)}{e.L}+\frac{(0,0.25)}{e.M}$
		T_4	$\frac{(0,0.25)}{b.M}+\frac{(1,1)}{b.H},\frac{(0,0.25)}{f.M}+\frac{(1,1)}{f.H}$
		T_5	$\frac{(0,0.25)}{c.M}+\frac{(1,1)}{c.H},\frac{(0,0.25)}{d.M}+\frac{(1,1)}{d.H},\frac{(0,0.25)}{a.L}+\frac{(1,1)}{a.M}+\frac{(0,0.25)}{a.H},\frac{(0,0.25)}{b.L}+\frac{(1,1)}{b.M}+\frac{(0,0.25)}{b.H},\frac{(0.5,0.63)}{e.M}+\frac{(0.5,0.63)}{e.H}$
w_2	B_3	T_6	$\frac{(0.5,0.63)}{c.M}+\frac{(0.5,0.63)}{c.H},\frac{(1,1)}{a.L}+\frac{(0,0.25)}{a.M},\frac{(0.5,0.63)}{b.M}+\frac{(0.5,0.63)}{b.H}$
	B_4	T_7	$\frac{(0.5,0.63)}{d.M}+\frac{(0.5,0.63)}{d.H},\frac{(0.5,0.63)}{b.L}+\frac{(0.5,0.63)}{b.M}$
w_3		T_8	$\frac{(0.5,0.63)}{c.M}+\frac{(0.5,0.63)}{c.H},\frac{(0.5,0.63)}{b.M}+\frac{(0.5,0.63)}{b.H},\frac{(0,0.25)}{f.L}+\frac{(1,1)}{f.M}+\frac{(0,0.25)}{f.H}$
		T_9	$\frac{(0,0.25)}{c.L}+\frac{(1,1)}{c.M}+\frac{(0,0.25)}{c.H},\frac{(0.5,0.63)}{d.M}+\frac{(0.5,0.63)}{d.H},\frac{(0.5,0.63)}{b.L}+\frac{(0.5,0.63)}{b.M},\frac{(1,1)}{f.L}+\frac{(0,0.25)}{f.M}$
	B_5	T_{10}	$\frac{(1,1)}{d.L}+\frac{(0,0.25)}{d.M},\frac{(0,0.25)}{b.L}+\frac{(1,1)}{b.M}+\frac{(0,0.25)}{b.H},\frac{(0.5,0.63)}{c.L}+\frac{(0.5,0.63)}{c.M}$

Figure 5: The initial value of quantitative data after fuzzy conversion from Figure 2.

Window	Batch	Tid	The converted fuzzy value
	B_1	T_1	$\frac{0.56}{d.M}+\frac{0.56}{d.H},\frac{1}{b.L}+\frac{0.13}{b.M},\frac{0.13}{e.M}+\frac{1}{e.H}$
		T_2	$\frac{1}{b.L}+\frac{0.13}{b.M},\frac{0.13}{e.L}+\frac{1}{e.M}+\frac{0.13}{e.H}$
w_1	B_2	T_3	$\frac{0.56}{b.L}+\frac{0.56}{b.M},\frac{0.56}{f.L}+\frac{0.56}{f.M},\frac{1}{e.L}+\frac{0.13}{e.M}$
		T_4	$\frac{0.13}{b.M}+\frac{1}{b.H},\frac{0.13}{f.M}+\frac{1}{f.H}$
	B_3	T_5	$\frac{0.13}{c.M}+\frac{1}{c.H},\frac{0.13}{d.M}+\frac{1}{d.H},\frac{0.13}{a.L}+\frac{1}{a.M},\frac{0.13}{a.H}\frac{0.56}{b.M}+\frac{0.56}{b.H},\frac{0.56}{e.L}+\frac{0.56}{e.M}$
w_2		T_6	$\frac{0.56}{c.M}+\frac{0.56}{c.H},\frac{0.56}{a.L}+\frac{0.56}{a.M},\frac{0.56}{b.M}+\frac{0.56}{b.H}$
	B_4	T_7	$\frac{0.56}{d.M}+\frac{0.56}{d.H},\frac{0.56}{b.L}+\frac{0.56}{b.M}$
w_3		T_8	$\frac{0.56}{c.M}\ \frac{0.56}{c.H},\frac{0.56}{b.M}+\frac{0.56}{b.H},\frac{0.13}{f.L}+\frac{1}{f.M}+\frac{0.13}{f.H}$
	B_5	T_9	$\frac{0.13}{c.L}+\frac{1}{c.M}+\frac{0.13}{c.H},\frac{0.56}{d.M}+\frac{0.56}{d.H},\frac{0.56}{b.L}+\frac{0.56}{b.M},\frac{1}{f.L}+\frac{0.13}{f.M}$
		T_{10}	$\frac{1}{d.L}+\frac{0.13}{d.M},\frac{0.13}{b.L}+\frac{1}{b.M}+\frac{0.13}{d.M},\frac{0.56}{b.L}+\frac{0.56}{b.M}$

Figure 6: Fuzzy compression results from Figure 5.

4.2. Weighted Sliding Window Model. In this paper, the sliding window model based on the average decay factor (ADF) is applied to construct the WFFPT2-tree of the type-2 fuzzy set [51] to mine the fuzzy frequent patterns (FFPs). The ADF was proposed by Han and Ding [55] for mining the FPs on the transaction data stream, but not for the quantitative data stream. The combination of the sliding window and the ADF not only considers the latest data in a limited time window but also fully considers the influence of historical data. The mining process of the FFPs on the data streams can effectively improve the integrity of the regular patterns. Moreover, the sliding window uses an improved decay factor δ to assign different weights to the new and the old data in different periods. When the data stream continues to arrive, the itemset changes steadily, avoiding the problem of data concepts drifting in the sliding window with the error factor.

Based on the continuity of the data stream, the knowledge contained therein will change over time. Usually, the importance or the value of the recent transactions is greater. Therefore, a decay factor is used in this paper to weight the transactions. According to the definition of the decay factor δ, it can be seen that the smaller the value, the less important the corresponding data, and the faster the attenuation. Conversely, the larger the value, the greater the importance and the slower the decay.

$$\delta \in (0, 1]. \tag{10}$$

In the time decay model [58, 59], the data of the sliding window w_n has the same decay factor δ. In the sliding window w_n, the support value $\text{freq}(P, w_n)$ of the item P can be represented as follows:

$$\text{freq}_w(P, w_n) = \begin{cases} \delta, & n = 1, \\ \text{freq}_w(P, w_{n-1}) \times \delta + r, & n \geq 2, \end{cases} \tag{11}$$

$$r = \begin{cases} 1, & P \in w_n, \\ 0, & P \notin w_n. \end{cases} \tag{12}$$

Literature [55] used a time decay model to mine the frequent patterns of the data stream closure through experiments. It is found through experiments that the mode support number after using the decay factor in the time attenuation model is much smaller than the support number without attenuation. If it is performed according to the conventional minimum support, the mining will lose possible frequent patterns. In order to solve this problem, the maximum allowable error factor is ε. In other words, both the

Window	Batch	Tid	$\theta = 0.5$	$\theta = 0.5,\ \varepsilon = 0.1,\ \delta_{\text{average}}$
	B_1	T_1	$\frac{0.13}{b.M}+\frac{1}{b.L}+\frac{0.56}{d.M}$	$\frac{0.13}{b.M}+\frac{1}{b.L}+\frac{0.56}{d.M}+\frac{0.56}{d.H}+\frac{0.13}{e.M}$
		T_2	$\frac{0.13}{b.M}+\frac{1}{b.L}$	$\frac{0.13}{b.M}+\frac{1}{b.L}+\frac{1}{e.M}$
w_1	B_2	T_3	$\frac{0.56}{b.M}+\frac{1}{b.L}$	$\frac{0.56}{b.M}+\frac{1}{b.L}+\frac{0.13}{e.M}+\frac{0.56}{f.M}$
		T_4	$\frac{0.13}{b.M}+\frac{1}{b.H}$	$\frac{0.13}{b.M}+\frac{1}{b.H}+\frac{0.13}{f.M}$
w_2	B_3	T_5	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.13}{c.M}+\frac{0.13}{d.M}$	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.13}{c.M}+\frac{0.13}{d.M}+\frac{1}{c.H}+\frac{1}{d.H}+\frac{0.56}{e.M}$
		T_6	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.56}{c.M}$	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.56}{c.M}+\frac{0.56}{c.H}$
	B_4	T_7	$\frac{0.56}{b.M}+\frac{0.56}{b.L}+\frac{0.56}{d.M}$	$\frac{0.56}{b.M}+\frac{0.56}{b.L}+\frac{0.56}{d.M}+\frac{0.56}{d.H}$
w_3		T_8	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.56}{c.M}$	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{0.56}{c.M}+\frac{0.56}{c.H}+\frac{1}{f.M}$
	B_5	T_9	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{1}{c.M}+\frac{0.56}{d.M}$	$\frac{0.56}{b.M}+\frac{0.56}{b.H}+\frac{1}{c.M}+\frac{0.56}{d.M}+\frac{0.13}{c.H}+\frac{0.56}{d.H}+\frac{0.13}{f.M}$
		T_{10}	$\frac{1}{b.M}+\frac{0.13}{b.L}+\frac{0.13}{b.H}+\frac{0.56}{c.M}+\frac{0.13}{d.M}$	$\frac{1}{b.M}+\frac{0.13}{b.L}+\frac{0.13}{b.H}+\frac{0.56}{c.M}+\frac{0.13}{d.M}$

Figure 7: Descending order of frequent pattern and critical frequent patterns.

FPs and the critical FPs (Definitions 4 and 5) will be saved during the mining process.

$$\begin{aligned}\text{freq}_w(P, w_n) &= \text{freq}_w(P, w_{n-1}) \times \delta + r = \sum_i r_i \times \delta^{n-i} \\ &= r_1 \times \delta^{n-1} + r_2 \times \delta^{n-2} + \cdots + r_m \\ &\le \delta^{n-1} + \delta^{n-2} + \cdots + r_m \le \frac{1}{1-\delta}.\end{aligned} \tag{13}$$

The decay factor of the sliding window $w_i \in \{w_1, w_2, \cdots, w_n\}$ is used to divide the value based on the fuzzification process including the three batches B_i, B_{i+1}, and B_{i+2}. Each batch contains two itemsets in chronological order in the data stream, and the size of the sliding window can be changed.

After determining the minimum support and the maximum allowable error thresholds according to the literature [38], the assumed 100% recall and 100% accuracy rates are used to estimate the decay factor δ. That is, when the recall rate is 100%, Equation (14) is satisfied, which is called the lower limit δ_{recall}. While, when the accuracy rate is 100%, Equation (15) is satisfied as the upper limit $\delta_{\text{precision}}$.

$$\delta \ge \sqrt[(2N-\theta N-1)]{\left[\frac{(\theta-\varepsilon)}{\theta}\right]^2} \quad \text{Recall} = 100\%, \tag{14}$$

$$\delta < \frac{(\theta-\varepsilon)N-1}{(\theta-\varepsilon)N} \quad \text{Precision} = 100\%. \tag{15}$$

Since both recall and accuracy cannot be considered at the same time, the choice of decay factor needs to balance the two. Therefore, this article uses the average value δ_{average} of the upper and the lower limits as used in the literature [59].

$$\delta_{\text{average}} = \frac{\left(\delta_{\text{recall}} + \delta_{\text{precision}}\right)}{2}. \tag{16}$$

Definition 9. The support value of a linguistic term is denoted as sup (R_{jh}), which is the scalar cardinality of each fuzzy frequent item [35] and is defined as follows:

$$\text{sup}\ (R_{ih}) = \sum_{R_{jh} \subseteq T_q \wedge T_q \in D'} f v_{iqh}^c. \tag{17}$$

According to Definitions 4 and 5, when the minimum support θ, the average decay factor δ_{average}, and the error factor ε are known, the frequent pattern and the critical frequent patterns weighted by the sliding window can be obtained according to Equations (11) and (13), respectively. Then, the patterns are sorted in descending order according to the accumulated count value of the frequent patterns and the critical frequent patterns as shown in Figure 7. Despite the prefuzzification step performed first, the weighted sliding window method will significantly change the count value of the item after multiple iterations due to the variability and the continuous characteristics of the data stream. The FFPs and the critical FFPs are listed in Figure 7. After the weighted sliding window process, there are critical FFPs that reflect the increment part based on the minimum support value designed by the user, which can reduce the concept drift problem.

4.3. The Construction-Renew Process of WFFPT2-Tree. This paper uses fuzzy-list sorted structure and WFFPT2-tree to generate fuzzy frequent patterns. The fuzzy-list sorted structure is used to store the attributes of the fuzzy frequent items after the prefuzzification. The attributes of FFIs include the name of the fuzzy frequent item, the count, and the link node. The construction process of WFFPT2-tree consists of adding transactions and deleting the transactions based on the fuzzy-list sorted structure. The proposed WFFPT2-tree is improved from the FP-tree [5]. The WFFPT2-tree with fuzzy-list sorted structure will reduce the memory usage and speed up the mining process.

In the WFFPT2-tree structure, each node represents a fuzzified semantic tree transaction, which includes the

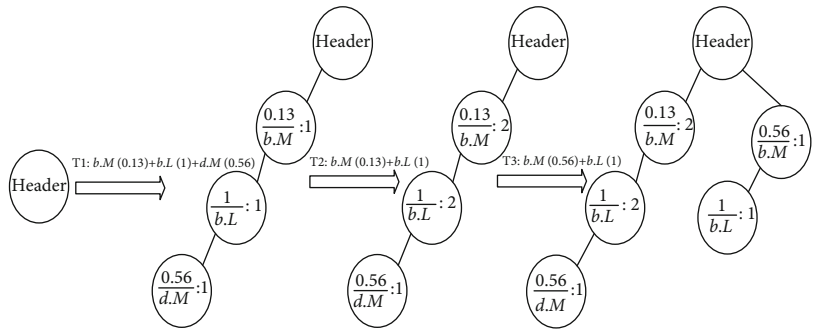

Figure 8: WFFPT2-Tree construction process of Transaction T1 to T3.

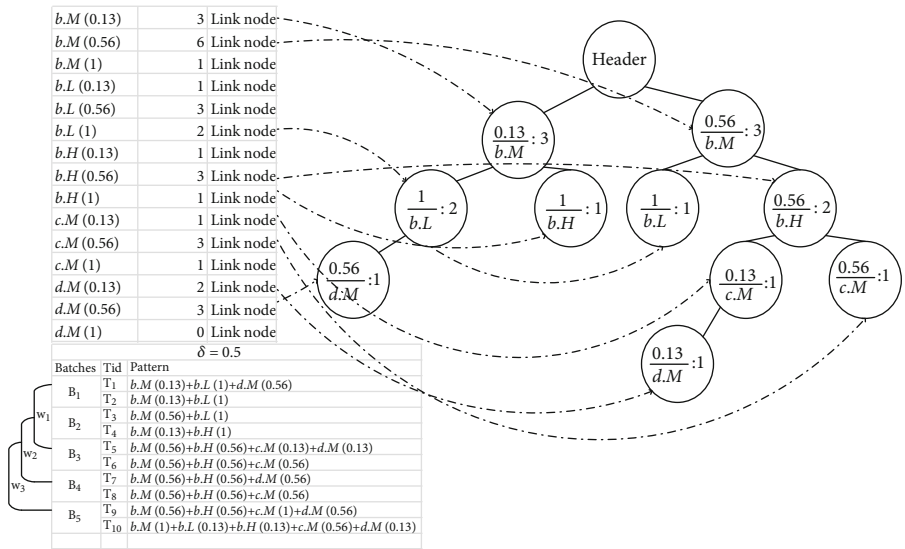

Figure 9: WFFPT2-tree construction result of the sliding window w_1.

weighted fuzzified semantic representation of itemsets and the corresponding membership and count values. Since the list is sorted in descending order first, the cache is stored in the form of a queue. Figure 8 shows the construction process of the WFFPT2-tree according to Figure 7.

The idea of generating fuzzy pattern trees in descending order adopted in this paper is derived from the tail expansion strategy. This paper adopts the horizontal link to quickly find the corresponding node on the node of the prefix pattern tree. The horizontal link will link the nodes with the same fuzzy linguistic in the descending prefix pattern tree, but the count value will be different, which is convenient for searching and inserting new subtrees or single nodes during the sliding window process.

4.3.1. Inserting Phase of the New Tree Transaction. The construction of the WFFPT2-tree is based on the fuzzy-list sorted structure. When the new tree transaction T_k generates, it will be added in the tree with the link node and sorted in the fuzzy-list structure. The specific implementation process is as follows. The fuzzy-list sorted structure table contains the frequent and the critical frequent patterns after the weighted sliding window and sorted by the count value as shown in Figure 9.

When the WFFPT2-tree is empty, the first fuzzified transaction T_1 through the weighted sliding window contains three fuzzy frequent items $\mathrm{T1} : 0.13/b.M : 1 + 1/b.L : 1 + 0.56/d.M : 1$, which are inserted into the root header in descending order. When the T_2 comes, the descending subtree has the same patterns and then the count increases as shown in Figure 8. When T_3 enters, there is no identical node in the current prefix pattern tree, so a new subtree is established. Figure 8 shows the detailed process of adding new branches to the fuzzy frequent pattern tree.

A similar process can be obtained when T_4, T_5, and T_6 are added to WFFPT2-tree in Figure 9. Figure 9 shows the sorted list structure of all items in the sliding window w_1, including the fuzzy semantic names, the decay factor, and the link node.

4.3.2. Deleting Phase of the Old Tree Transaction. This operation occurs during the sliding phase of the weighted sliding window over the data stream. For example, after the establishment of tree transactions of w_1 shown in Figure 9, the fuzzy frequent transactions enter from w_1 to w_2 stage, which should be adding to the WFFPT2-tree shown in blue in Figure 10 and the old tree transactions should be deleted shown in red in Figure 10. First, pop a piece of data from

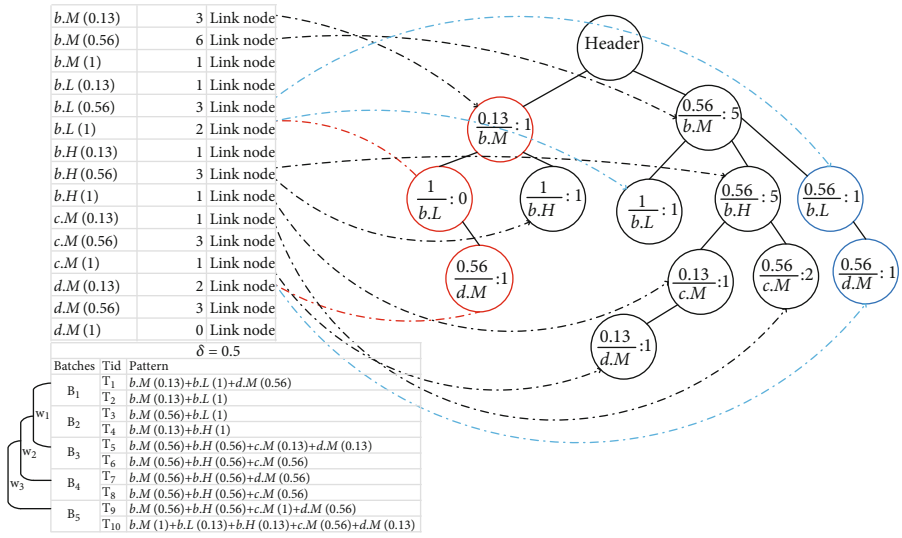

FIGURE 10: WFFPT2-tree construction process of the sliding window from w_1 to w_2.

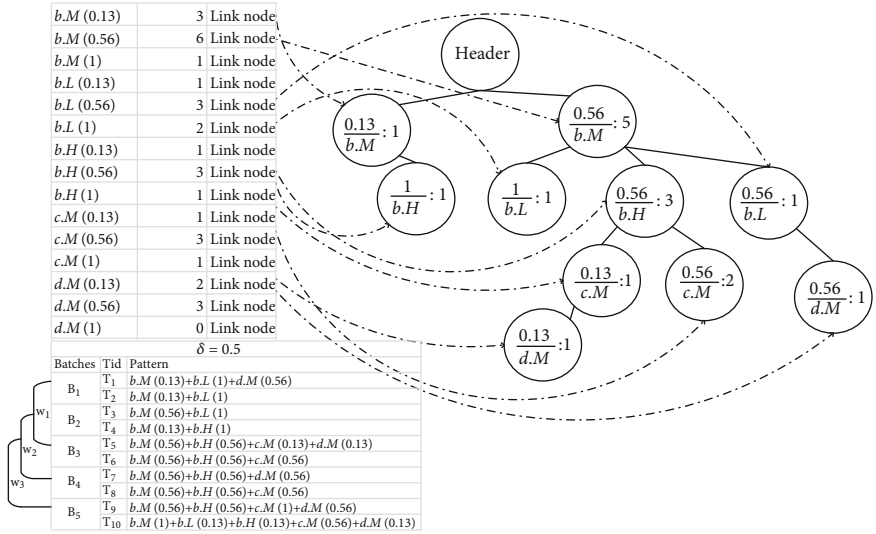

FIGURE 11: WFFPT2-tree construction result of the sliding window w_2.

the front of the fuzzy-list sorted structure table, and use the pointer to point to the end of the tree transaction in the WFFPT2-tree to find the transaction which needs to be deleted in the tree. Then, from the tail of the WFFPT2-tree to the root, update the count value of the node along the path. It can be seen from the path that the red part whose weighted count value does not meet the minimum support needs to be deleted from the tree. Usually, their horizontal links are also deleted at the same time.

Figure 10 shows the new node in the sliding window in blue and the results of the deletion. Figure 11 shows the current WFFPT2-tree of the sliding window w_2 after deleting the node. Figures 12 and 13 show the processes of adding and deleting after the sliding window w_3 has passed, respectively. Figures 8–13 show the example of WSWFFP-T2 algorithm and illustrate the entire process of the sliding window based on the fuzzy-list sorted structure in the proposed algorithm, including the processes of construction and adding and deleting the fuzzy node.

4.3.3. FFPs Mining Phase. The mining process of frequent patterns is performed by querying the mining results of the current window. In the WFFPT2-tree, all frequent patterns can be mined through recursion to facilitate the finding of the entire WFFPT2-tree, and the patterns with support greater than the threshold are frequent patterns. Although the WSWFFP-T2 algorithm proposed in this paper mines frequent patterns with a fixed window size, the processes of inserting the new nodes and deleting the nodes in the algorithm are independent of each other. Thus, increasing and reducing the number of sliding windows will not affect the correctness of the algorithm. This part is also tested and verified in subsequent experiments. The algorithm can easily be applied to the variable-length sliding window model.

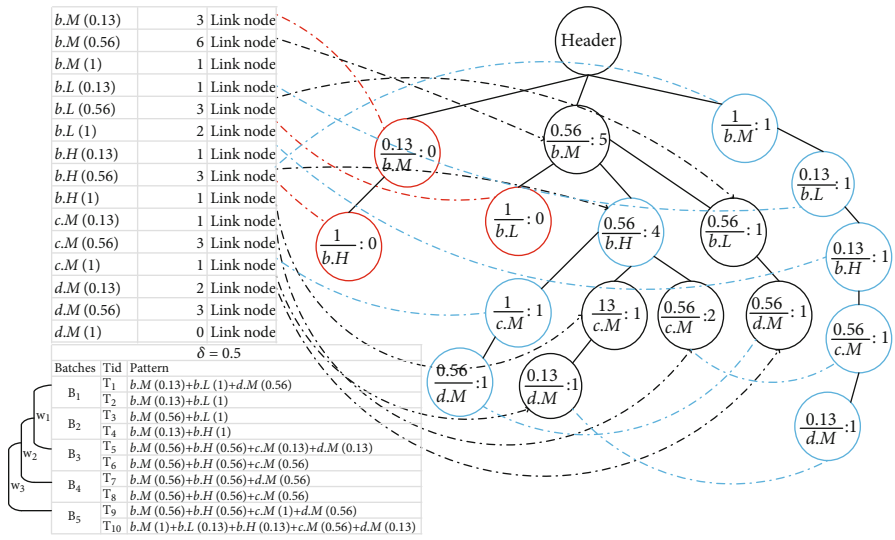

Figure 12: WFFPT2-tree construction process of the sliding window from w_2 to w_3.

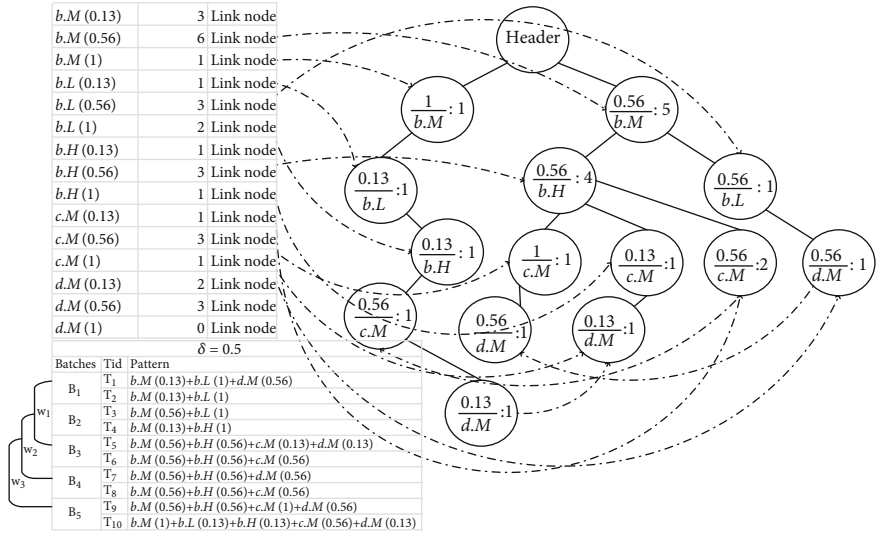

Figure 13: WFFPT2-tree construction result of the sliding window w_3.

4.3.4. The Pseudocode of the Proposed WFFPT2-Tree Algorithm.

5. Experimental Study

5.1. Experimental Environment and Data Stream. This section presents the experimental results and the performance analysis of the WSWFFP-T2 algorithm. All the experiments are performed on Windows 10/64-bit system, which is configured as Intel Premium, and all algorithms are implemented in Java (IntelliJ IDEA 2019.3.3 x64) [60]. The data stream studied in this article is the real artificial data stream of the breast.w.arff (Wisconsin Prognostic Breast Cancer Database) to support the findings of this study.

The dataset contains 10 attributes: Clump_Thickness numeric, Cell_Size_Uniformity numeric, Cell_Shape_Uniformity numeric, Marginal_Adhesion numeric, Single_Epi_Cell_Size numeric, Single_Epi_Cell_Size numeric, Bare_Nuclei numeric, Bland_Chromatin numeric, Normal_Nucleoli numeric, Mitoses numeric, and Class {benign,malignant}. The benign and malignant give the results of the breast cancer. The data of the attributes are integer [1, 10].

5.2. Performance Analysis. Figures 14 and 15 show the time and the memory consumption of the proposed algorithm on the breast cancer database, which is generated as a data stream. The number of the transactions is 64000. During the memory consumption and the running time, the window size gradually expands from 4 to 250. It can be seen from Figure 14 that as the window increases during the operation of the algorithm, the time consumption changes within a certain interval, but the overall trend is stable. The reason is to maintain a record of all tree transactions in the sliding window. The algorithm needs to add the newly added sliding window list and delete the list of nodes that are not in the window of the tree transaction. Since the length of the transaction itemset in the data stream is uncertain, it will bring

```
Input:
    Int   SizeP//the size of Batch
    Int   Sizew//the size of the window
    Double minsup //the minimum support value
    List   ItemTalbe//the fuzzy-list sorted-structure table
    List   InstanceList//the storage table of Instance
    ArffFIleStream Arff//Arff data stream
Output:FFPs
    Main(){//main function
    1 Arff.LoadArffFIle();//Read in data stream and fuzzify linguistic data based on type 2 fuzzification method
    2     CreateWFFPWindow();//Create the weighted sliding window
    3     While(Arff.hasInstance!=nul)//Read data stream
    4       {
    5         Instance=LoadNextInstance();//read the next instance
    6         FuzzySetCompute(instace);//Calculate the membership degree of type 2 fuzzy set by formula
    7         Compress(Tmp);//Compression of the fuzzy calculation result; ;
    8         UpdateWFFPWindows();//update the weighted fuzzy frequent window
    9       }
    10    OutputFFPT2(minsup); //Output fuzzy frequent patterns meeting the minimum support degree;
    11    }
Sub Algorithm 1: CreateWFFPWindow
1   CreateWFFPWindow(){
2     Times=0;
3     While(Times<SizeW){
4     Instance=Arff.nextInstace();//Read according to the window size
5     UpdateItem(Instance); //Update Item table
6     InstanceList.add(Instance);
7     }
8     //ItemTable sorts itself first, and obtains the path Order according to the Count in the Item table
9     ItemTable.sort();
10      Getorder();
11      InstanceList.SortByOrder();//Sort the instances in Oder order
12      CreateTree(InstanceLIst);//Insert the instance into the tree
13     }
Sub Algorithm 2: FuzzySetCompute
1     FuzzySetCompute(instance){
2     //Calculate F.lower() and F.upper() according to the model
3     Tmp.add(F.lower(),F.upper());
4     }
Sub Algorithm 3: UpdateWFFPWindows
1     UpdateWFFPWindows{                    //Update window
2     DelTree();      //Same as createwindow(), delete the old pane instance from the tree
3   }
Sub Algorithm 4: OutputFFPs
1     OutputPPFs(minsup){
2     //caculate the Sup(Item) from ItemTable
3     If sup(item)>minsup{
4     //output the results through the corresponding Item.Link
5     OutputFromLink(item);
6     }}
```

Algorithm 1: WFFPT2-tree algorithm.

time in the obfuscation process. Uncertainty is caused by the characteristics of the data in the data stream. At the same time, the first step in the calculation of the algorithm is to perform type-2 fuzzification on the digital data stream. This will also increase the number of nodes that need to be processed and increase the time overhead of the algorithm. However, the overall running time is stable and concentrated within an acceptable range.

Figure 15 shows that the memory increases with the continuous increase of the window size, which indicates that the tree transaction and the data in the list structure to be processed are increased with the increase in the sliding window. After the transactions in the sliding window are fuzzified, the fuzzification results increase the number of original transactions in a linear relationship of k times. Therefore, it can be seen that the window and memory consumption

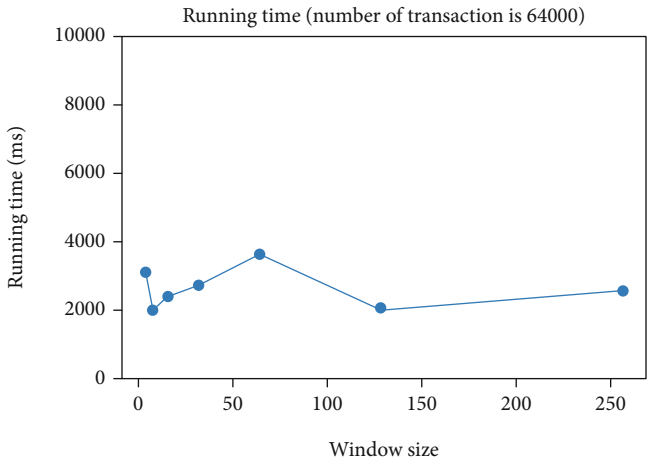

Figure 14: Running time based on variable sliding window.

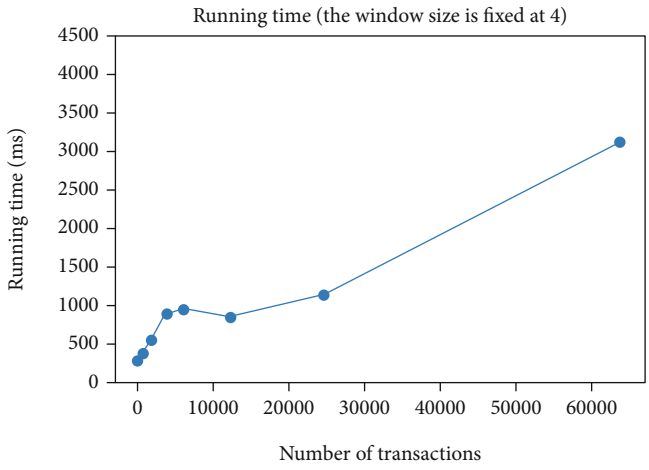

Figure 16: Running time based on transactions.

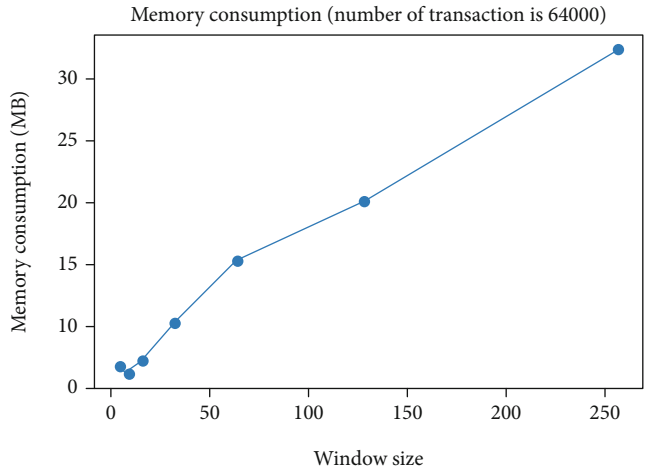

Figure 15: Memory based on variable sliding window.

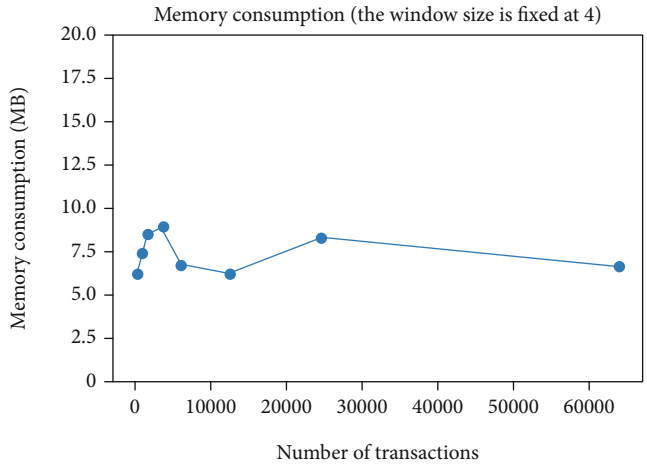

Figure 17: Memory consumption based on transactions.

increase in a linear manner. The problem of excessive memory consumption can be avoided by reasonably selecting the appropriate sliding window for the data stream. The authors will continue to study how to dynamically adjust the sliding window for the data stream.

The above experimental analysis shows that the sliding window continues to increase, and the memory consumption of the algorithm increases linearly. Further experiments are conducted to verify the relationship between the number of transactions and memory and the operating consumption when the sliding window is fixed. Figures 16 and 17 show the relationship between running time and transaction volume when the sliding window is fixed at 4 and the number of transactions continues to increase.

As can be seen from Figure 16 that running time continues to increase, as the volume of the transactions increases. This is because the volume of the transactions that need to be processed increases, and the sliding window needs to be fixed. After the algorithm needs to update the fading factor of each sliding window, the time of transaction obfuscation also increases. The other reason is that the more the transaction volume, the longer the running time. Figure 17 shows that under the same sliding window, the memory consumption tends to be stable as the transaction volume increases. During the creation of the sliding window, as the window continues to move, new data is added to the list and the tree is created. When the sliding window is across the data stream, the nodes that are not in the current window need to be clear from the list structure, and the input and the output operation brought about makes the memory consumption fluctuate. However, because the amount of data in the sliding window tends to be stable during the Java

Table 1: Percentage of running time of each module of WSWFFPs-T2 algorithm.

Read data	Fuzzy process	Renew node	Delete node	Others	Window sizes	Item numbers
17.8%	44.4%	1.8%	2.2%	33.8%	4	64000
17.0%	40.9%	2.9%	3.0%	36.2%	8	64000
22.8%	36.0%	5.7%	7.7%	27.8%	16	64000
20.6%	22.6%	3.8%	6.6%	46.4%	32	64000
25.0%	18.3%	7.0%	6.1%	43.6%	64	64000
23.0%	28.4%	3.3%	5.1%	40.2%	128	64000
15.9%	17.9%	2.8%	9.2%	54.2%	256	64000

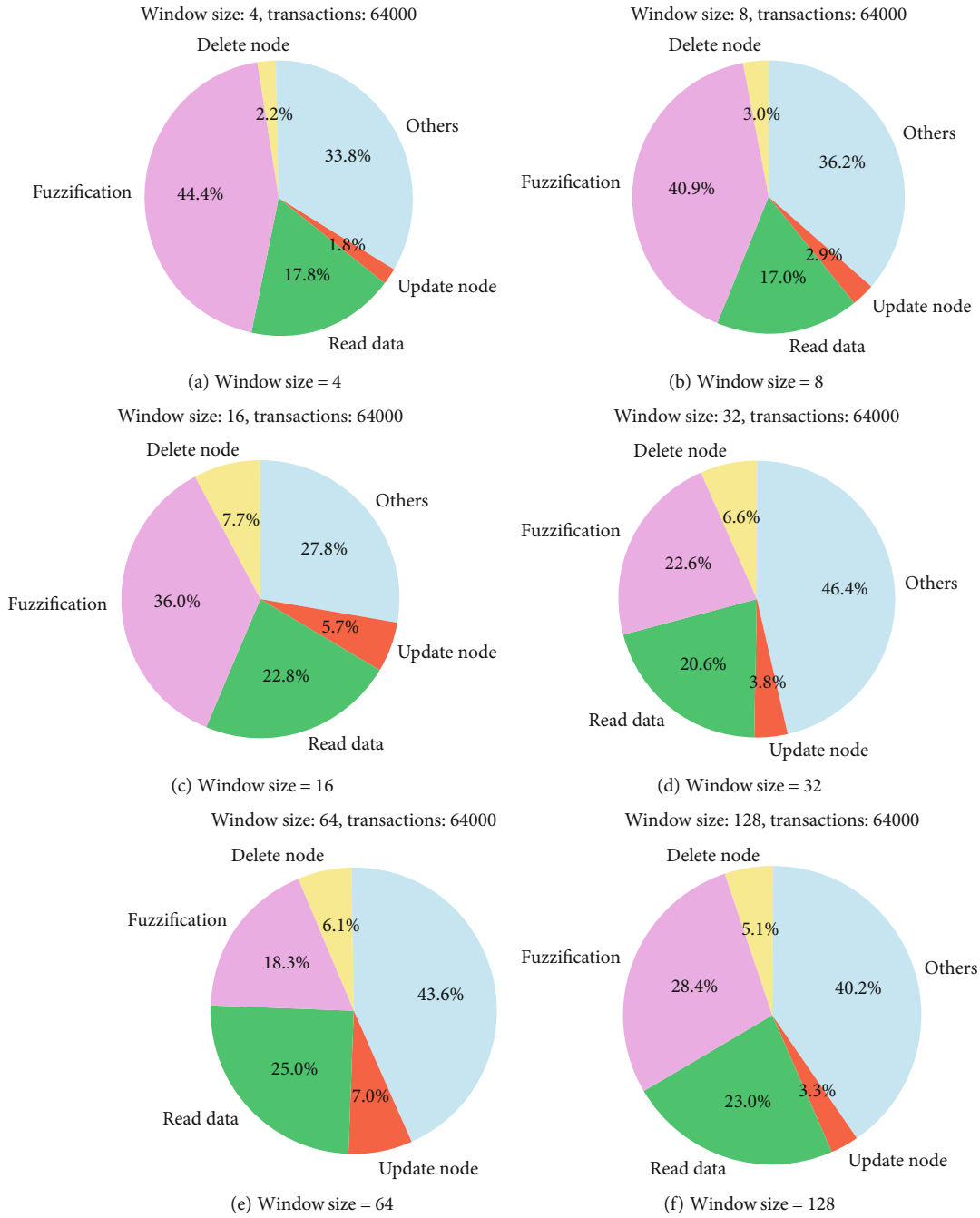

Figure 18: The running time ratio of each part of the algorithm based on different window sizes.

programming process, the memory consumption of the algorithm operation tends to change steadily within a specific time interval. It is also the benefit of the sliding windows, and the memory consumption of the algorithms without sliding windows will continue to increase because the release speed is relatively slow.

In order to further analyze the performance of the algorithm proposed in this paper, Table 1 shows the running time test of each function of the proposed algorithm in this paper when the number of transactions is fixed at 64000. The tested parameters include the ratio of the time taken by the node to be deleted, the ratio of the time required to obscure the data stream, the ratio of time to update the node, the ratio of time to delete the node, and the ratio of other times in the operation.

Figure 18 shows that when the number of transactions is fixed at 64000, the size of the sliding window increases from 4 to 128 and the time proportions of each part of the algorithm are tested as follows. Figures 18(a)–18(f) show that the operation time of reading the data increases from 17.8% to 23%, the fuzzification process is reduced from 44.4% to 28%, updating the nodes is increased from 1.8% to 3.3%, deleting the nodes is increased from 2.2% to 5.7%, and the other operations varied from 27.8% to 46.45%. Figure 19 shows an

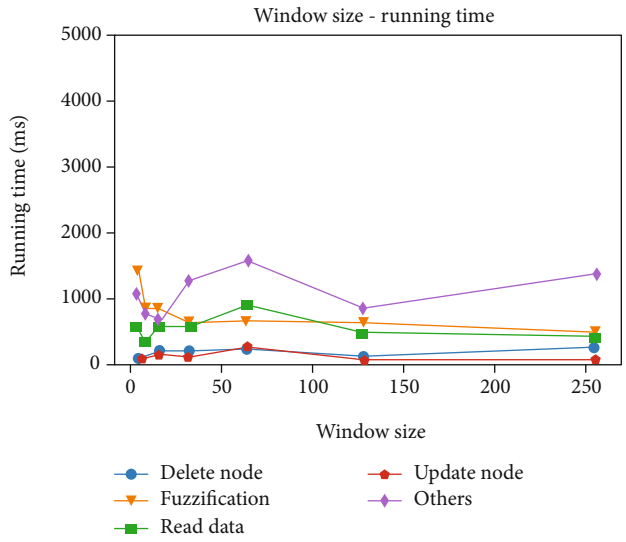

Figure 19: The average running time ratio of each part of the algorithm based on different window sizes.

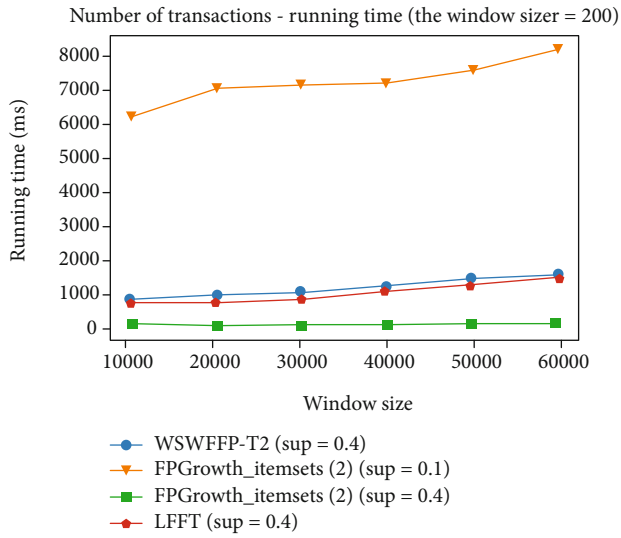

Figure 20: Execution time of the compared algorithms.

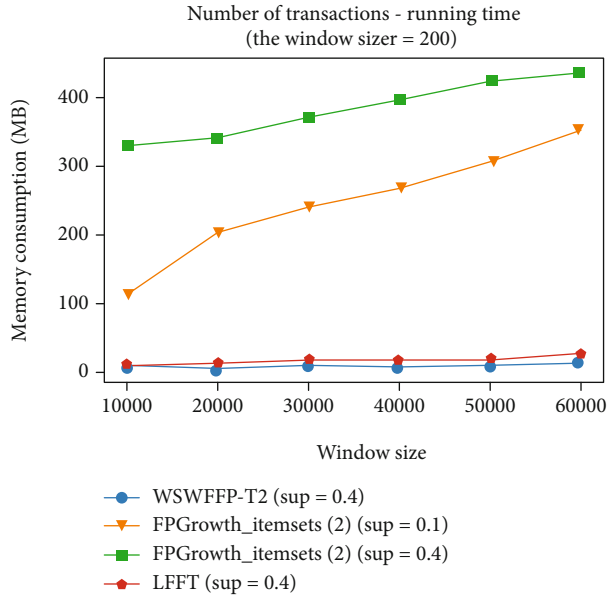

Figure 21: Memory consumption of the compared algorithms.

average time distribution of each part. In the process of selecting the size of the sliding window, an appropriate size of the sliding window can be selected as a reference to Figures 18(a)–18(f) given in the experiments.

In the overall operation of the algorithm proposed in this paper, the time consumption is mainly concentrated in the data reading and fuzzification process, because when the window is fixed, updating and deleting the nodes in the construction of lists and trees are relatively small and not more than 10%.

5.3. Results and Comparisons. In the experiments, the database is tested according to the predefined membership and support with different numbers of transactions. In literature [35], the LFFP algorithm outperforms the Apriori algorithm; thus, it is unnecessary to conduct the performance of the Apriori algorithm in our experiments. The execution time of sup = 0.1 of FPGrowth_itemsets algorithm consumes more time than sup = 0.4. For the FPGrowth_itemsets algorithm is not adapted to the data stream, the authors do the experiment at the same number of transaction with sliding window.

From Figure 20, it can be observed that the execution time of the proposed algorithm is longer than the FPGrowth_itemsets algorithm but in an acceptable scale. The first reason is that the proposed algorithm of fuzzification process is based on the type-2 fuzzy set theory generating numbers of fuzzy nodes including the frequent patterns and the critical frequent patterns shown in Figures 18(a)–18(f). The second reason is that the input and output data operation and fuzzification processes consume nearly 50% of the whole algorithm from the experimental results in Figure 19. But the proposed algorithm can provide linguistic frequent pattern over the data streams that the people can understand the results meaning without the help of experts. The other one is the proposed algorithm which uses the precision and recall rates to avoid the concept drift during the data stream mining process.

It can be observed from Figure 21 that the proposed WSWFFP-T2 algorithm generally has little memory consumption than the FPGrowth_itemsets algorithm at the same support with different numbers of transactions. As the number of transactions increases, the memory consumption of FPGrowth_itemsets algorithm is linearly increasing with the number of transactions. The proposed algorithm has stable memory consumption at a lower level.

Based on the above observations, it can be concluded that the proposed WSWFFP-T2 algorithm with one scan can mine with varied weighted sliding window to mine MFFIs over the data stream with efficiency and high performance compared with the other related algorithms.

6. Conclusion

In this paper, an efficient fuzzy-list sorted structure and WSWFFP-T2 tree are presented to keep the necessary fuzzy

information for mining the MFFIs based on the artificial datasets of medical data stream. The weighted sliding window and the construction strategy of fuzzy frequent pattern tree are designed to adapt to the real-time data stream and also to reduce memory consumption in mining the MFFIs. The precision and recall factors are designed to ensure the accuracy of the proposed algorithm. The experimental results and comparative analysis demonstrate that the proposed algorithm outperforms the classical and the latest algorithms over the quantitative data stream in the real-time medical data stream.

Data Availability

The Wisconsin Prognostic Breast Cancer Database (breast.-cancer.arff data) is used to support the findings of this study released upon application which can be achieved on the GitHub as https://github.com/renatopp/arff-datasets/tree/master/classification.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research is sponsored by the National Natural Science Foundation of China (No. 61762071, No. 61872196, No. 61872194, and No. 61902196), the Scientific and Technological Support Project of Jiangsu Province (No. BE2019740, No. BK20200753, and No. 20KJB520001), the Major Natural Science Research Projects in Colleges and Universities of Jiangsu Province (No. 18KJA520008), the Six Talent Peaks Project of Jiangsu Province (RJFW-111), and the Postgraduate Research and Practice Innovation Program of Jiangsu Province (No. KYCX19_0909, No. KYCX19_0911, No. KYCX20_0759, No. KYCX21_0787, No. KYCX21_0788 and No. KYCX21_0799) and BSYKJ2021-ZZ01.

References

[1] T.-I.-A.-S. R. Agrawal, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216, Washington, D.C., United States, 1993.

[2] M.-J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372–390, 2000.

[3] M.-J. Zaki and G. Karam, "Fast vertical mining using diffsets," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '03*, pp. 326–335, Washington, D.C., United States, 2003.

[4] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1–12, 2000.

[5] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[6] G. Liu, H. Lu, W. Lou, Y. Xu, and J. X. Yu, "Efficient mining of frequent patterns using ascending frequency ordered prefix-tree," *Data Mining and Knowledge Discovery*, vol. 9, no. 3, pp. 249–274, 2004.

[7] G. Liu, H. Lu, Y. Xu, and J. X. Yu, "Ascending frequency ordered prefix-tree: efficient mining of frequent patterns," in *Eighth International Conference on Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings*, pp. 65–72, Kyoto, Japan, 2003.

[8] G. Liu, H. Lu, J. X. Yu, W. Wang, and X. Xiao, "AFOPT: an efficient implementation of pattern growth approach," *FIMI*, pp. 1–10, 2003.

[9] G. Grahne and Z. Jianfei, "Efficiently using prefix-trees in mining frequent itemsets," *FIMI*, vol. 90, p. 65, 2003.

[10] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 10, pp. 1347–1362, 2005.

[11] B. Schlegel, G. Rainer, and L. Wolfgang, "Memory-efficient frequent-itemset mining," in *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 461–472, Uppsala Sweden, 2011.

[12] Q. Jun-Feng and L. Mengchi, "A high-performance algorithm for frequent itemset mining," in *International Conference on Web-Age Information Management*, pp. 71–82, Springer, 2012.

[13] Y. Aumann and Y. Lindell, "A statistical theory for quantitative association rules," *Journal of Intelligent Information Systems*, vol. 20, no. 3, pp. 255–283, 2003.

[14] H. Kalia, S. Dehuri, and A. Ghosh, "A survey on fuzzy association rule mining," *International Journal of Data Warehousing and Mining*, vol. 9, no. 1, pp. 1–27, 2013.

[15] K. H. Lee, *An extension of association rules using fuzzy sets*, UCI repository of machine learning databases, Seoul Korea, 1997.

[16] Y. Ke, J. Cheng, and W. Ng, "An information-theoretic approach to quantitative association rule mining," *Knowledge and Information Systems*, vol. 16, no. 2, pp. 213–244, 2008.

[17] K. Wang, T. Soon-Hock-William, and L. Bing, "Interestingness-based interval merger for numeric association rules," *InKDD*, vol. 98, pp. 121–128, 1998.

[18] G. J. Simon, P. J. Caraballo, T. M. Therneau, S. S. Cha, M. R. Castro, and P. W. Li, "Extending association rule summarization techniques to assess risk of diabetes mellitus," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 130–141, 2015.

[19] J. Mata, J. L. Alvarez, and J. C. Riquelme, "An evolutionary algorithm to discover numeric association rules," in *Proceedings of the 2002 ACM Symposium on Applied Computing - SAC '02*, pp. 590–594, Madrid, Spain, 2002.

[20] A. Salleb-Aouissi, C. Vrain, and C. Nortet, "QuantMiner: a genetic algorithm for mining quantitative association rules," *IJCAI*, vol. 7, pp. 1035–1040, 2007.

[21] G. Yang, "A novel method for mining association rules from continuous attributes based on cultural immune algorithm," *Journal of Information and Computational Science*, vol. 10, no. 9, pp. 2845–2853, 2013.

[22] R. Rastogi and K. Shim, "Mining optimized association rules with categorical and numeric attributes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 29–50, 2002.

[23] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes," *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 1–12, 1999.

[24] B. Chen, X. P. Liu, S. S. Ge, and C. Lin, "Adaptive fuzzy control of a class of nonlinear systems by fuzzy approximation approach," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 6, pp. 1012–1021, 2012.

[25] S.-M. Bridges and R.-B. Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection," in *Proceedings of 12th Annual Canadian Information Technology Security Symposium*, pp. 109–122, Ottawa, Canada, 2000.

[26] J. S. Yue, E. Tsang, D. Yeung, and D. Shi, "Mining fuzzy association rules with weighted items," in *2000 IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions' Cat. No.0*, pp. 1906–1911, Nashville, TN, USA, 2000.

[27] A. Fu, M. H. Wong, S. C. Sze, W. C. Wong, W. L. Wong, and W. K. Yu, "Finding fuzzy sets for the mining of fuzzy association rules for numerical attributes," *Proceedings of the First International Symposium on Intelligent Data Engineering and Learning (IDEAL'98)*, vol. 25, no. 5, pp. 263–268, 1998.

[28] C. M. Kuok, A. Fu, and M. H. Wong, "Mining fuzzy association rules in databases," *ACM SIGMOD Record*, vol. 27, no. 1, pp. 41–46, 1998.

[29] L.-A. Zadeh, "A computational approach to fuzzy quantifiers in natural languages," *Computers & Mathematics with Applications*, vol. 9, no. 1, pp. 149–184, 1983.

[30] L.-A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning–I," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[31] V. Ranjbar and G. Hesamian, "Copula function for fuzzy random variables: applications in measuring association between two fuzzy random variables," *Statistical Papers*, vol. 61, no. 1, pp. 503–522, 2020.

[32] T.-P. Hong, C.-Y. Wang, and Y.-H. Tao, "A new incremental data mining algorithm using pre-large itemsets1," *Intelligent Data Analysis*, vol. 5, no. 2, pp. 111–129, 2001.

[33] T.-P. Hong, C.-S. Kuo, and S.-C. Chi, "Mining association rules from quantitative data," *Intelligent Data Analysis*, vol. 3, no. 5, pp. 363–376, 1999.

[34] N. N. Karnik and J. M. Mendel, "Type-2 fuzzy logic systems: type-reduction," in *1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, pp. 2046–2051, San Diego, CA, USA, 1998.

[35] T.-Y. Wu, J. C.-W. Lin, U. Yun, C.-H. Chen, G. Srivastava, and X. Lv, "An efficient algorithm for fuzzy frequent itemset mining," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 5787–5797, 2020.

[36] G. Lee, U. Yun, and K. H. Ryu, "Sliding window based weighted maximal frequent pattern mining over data streams," *Expert Systems with Applications*, vol. 41, no. 2, pp. 694–708, 2014.

[37] G.-H. Li and C. Hui, "Mining the frequent patterns in an arbitrary sliding window over online data streams," *Journal of Software*, vol. 19, no. 10, pp. 2585–2596, 2009.

[38] H. Chen, L. C. Shu, J. Xia, and Q. Deng, "Mining frequent patterns in a varying-size sliding window of online transactional data streams," *Information Sciences*, vol. 215, pp. 15–36, 2012.

[39] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, , pp. 346–357, Elsevier, 2002.

[40] H.-F. Li, M.-K. Shan, and S.-Y. Lee, "DSM-FI: an efficient algorithm for mining frequent itemsets in data streams," *Knowledge and Information Systems*, vol. 17, no. 1, pp. 79–97, 2008.

[41] M. Memar, M. H. Sadreddini, M. Deypir, and S. M. Fakhrahmad, "A block-based approach for frequent itemset mining over data streams," in *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp. 1647–1651, Shanghai, China, 2011.

[42] H.-F. Li and S.-Y. Lee, "Mining frequent itemsets over data streams using efficient window sliding techniques," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1466–1477, 2009.

[43] F. Guidan and Y. Shaohong, "A frequent itemsets mining algorithm based on matrix in sliding window over data streams," in *2013 Third International Conference on Intelligent System Design and Engineering Applications*, pp. 66–69, Hong Kong, China, 2013.

[44] J.-H. Chang and W.-S. Lee, "Finding recent frequent itemsets adaptively over online data streams," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 487–492, Washington, D.C, 2003.

[45] S.-J. Shin, D.-S. Lee, and W.-S. Lee, "CP-tree: an adaptive synopsis structure for compressing frequent itemsets over online data streams," *Information Sciences*, vol. 278, pp. 559–576, 2014.

[46] C.-W. Lin, H. Tzung-Pei, and W.-H. Lu, "Linguistic data mining with fuzzy FP-trees," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4560–4567, 2010.

[47] J. C.-W. Lin, T.-P. Hong, and T.-C. Lin, "A CMFFP-tree algorithm to mine complete multiple fuzzy frequent itemsets," *Applied Soft Computing*, vol. 28, pp. 431–439, 2015.

[48] C.-W. Lin and T.-P. Hong, "Mining fuzzy frequent itemsets based on UBFFP trees," *Journal of Intelligent & Fuzzy Systems*, vol. 27, no. 1, pp. 535–548, 2014.

[49] J. Delin, S. Sharoff, S. Lillford, and C. Barnes, "Linguistic support for concept selection decisions," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 21, no. 2, pp. 123–135, 2007.

[50] L.-A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[51] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems: handling the uncertainty associated with surveys," in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, pp. 1546–1551, Seoul, Korea (South), 1999.

[52] C.-H. Chen, T.-P. Hong, and Y. Li, "Fuzzy association rule mining with type-2 membership functions," in *Asian Conference on Intelligent Information and Database Systems*, pp. 128–134, Springer, Cham, 2015.

[53] J.-M. Mendel, "Computing with words and its relationships with fuzzistics," *Information Sciences*, vol. 177, no. 4, pp. 988–1006, 2007.

[54] C.-H. Chen, H. Chou, T.-P. Hong, and Y. Nojima, "Cluster-based membership function acquisition approaches for mining fuzzy temporal association rules," *IEEE Access*, vol. 8, pp. 123996–124006, 2020.

[55] M. Han and J. Ding, "Efficient methods to set decay factor of time decay model over data streams," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 6, pp. 5807–5820, 2019.

[56] H. Zheng, J. He, Y. Zhang, and Y. Shi, "A fuzzy decision tree approach based on data distribution construction," *Proceedings of the Australasian Computer Science Week Multiconference*, 2017, pp. 1–10, Geelong, Australia, 2017.

[57] S. Du, M. Han, M. Shen, C. Zhang, R. Sun, and T. Gao, "A survey of ensemble classification over concept drift data streams," *Journal of Nonlinear and Convex Analysis*, vol. 21, no. 7, pp. 1567–1579, 2020.

[58] L. Chen and Q. Mei, "Mining frequent items in data stream using time fading model," *Information Sciences*, vol. 257, pp. 54–69, 2014.

[59] M. Han, D. Jian, and L. Juan, "TDMCS: an efficient method for mining closed frequent patterns over data streams based on time decay model," *International Arab Journal of Information Technology*, vol. 14, no. 6, pp. 851–860, 2017.

[60] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz et al., "The SPMF open-source data mining library version 2," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 36–40, Springer, Cham, 2016.