Alexandria Engineering Journal (2015) **54**, 163–174

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Efficient mining fuzzy association rules from ubiquitous data streams

Amal Moustafa [*], Badr Abuelnasr, Mohamed Said Abougabal

Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, Egypt

Received 22 January 2015; revised 9 March 2015; accepted 15 March 2015
Available online 11 April 2015

KEYWORDS

Data mining;
Fuzzy association rules;
Fuzzy sets;
Data streams;
Ubiquitous data streams

Abstract Due to the development in technology, a number of applications such as smart mobile phone, sensor networks and GPS devices produce huge amount of ubiquitous data in the form of streams. Different from data in traditional static databases, ubiquitous data streams typically arrive continuously in high speed with huge amount, and changing data distribution. Dealing with and extracting useful information from that data is a real challenge. This raises new issues, that need to be considered when developing association rule mining techniques for these data. It should be noted, that data, in the real world, are not represented in binary and numeric forms only, but it may be represented in quantitative values. Thus, using fuzzy sets will be very suitable to handle these values.

In this paper the problem of mining fuzzy association rules from ubiquitous data streams is studied, and a novel technique FFP_USTREAM (Fuzzy Frequent Pattern Ubiquitous Streams) is developed. This technique integrates fuzzy concepts with ubiquitous data streams, employing sliding window approach, to mine fuzzy association rules. In addition, the complexity and the efficiency of this technique are discussed. Examples of real data sets are used **to test** the proposed technique. Further research issues are also suggested.

1. Introduction

Recent emerging applications, such as network traffic monitoring, sensor network data analysis, web click stream mining, power consumption measurement, and dynamic tracing of stock market fluctuations, call for studying a new kind of data. This is called stream data, which can be continuous, potentially infinite flow of information, as opposed to finite, statically stored data sets. Stream Data Mining is the process of extracting knowledge structures from continuous, and rapid data records.

The dissemination of data streams systems, wireless networks and mobile/handheld devices motivates the need for an efficient data analysis tool capable of gaining insights about these continuous data streams [1]. Ubiquitous data streams mining (UDM) is the process of pattern discovery on mobile, embedded and ubiquitous devices. It represents the next generation of data mining systems, that will support the intelligent and time-critical information needs of mobile users, and will facilitate “anytime, anywhere” data mining.

Fuzzy logic is a type of logic used in artificial intelligence. It is referred to as a multi-valued logic.Instead of having two

[*] Corresponding author.

Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2015.03.015
1110-0168 © 2015 Faculty of Engineering, Alexandria University. Production and hosting by Elsevier B.V.

values (true and false), there are a continuum of possible truth values [2]. In fuzzy logic, every proposition is a statement that is assigned a number between 0 (false) and 1 (true), such a statement is called a fuzzy proposition. Fuzzy logic provides a powerful tool to categorize a concept in an abstract way by introducing vagueness.

Many data streams applications exist, that require association rule mining, such as network traffic monitoring and web click streams analysis. These applications' goal is to discover important associations among items as the presence of some items will imply the presence of others.

Fuzzy association rule approach could combine data mining results with human expertise and background knowledge, in the form of rules, to attain labeled classes for classification of data streams. Another advantage of the fuzzy logic approach is that it gives classification results, which include a degree of probability.

This paper demonstrates the effectiveness of Fuzzy Association Rules Mining from Ubiquitous Data Streams. This will be revealed in the coming sections. For this purpose, the remaining part of the paper is organized as follows: the work related to Fuzzy Association rules mining and ubiquitous data streams mining are reviewed and summarized in Section 2. An efficient fuzzy association rules mining technique from ubiquitous data streams is proposed in Section 3, and its complexity is analyzed in Section 4. Moreover,experimental results are discussed in Section 5. The paper is concluded and future research issues are presented, in Section 6.

2. Related work

This paper belongs to different inter-related research fields. The two main related topics of this work are presented: Ubiquitous Data Streams Mining Techniques that can effectively analyze continuously streaming data and Fuzzy association rules mining algorithms. These two fields will be surveyed.

2.1. Ubiquitous data streams mining

The approach, based on finite statically stored data sets, is not satisfactory in several applications. These include wireless network analysis, intrusion detection, stock market analysis, sensor network data analysis, and, in general, any setting in which every information available should be used to make an immediate decision. Such situations demand new algorithms, that are able to cope with evolutions of data as shown in Table 1 [3].

Ubiquitous Data Mining (UDM) is the time-critical process of pattern discovery in data streams in a wireless environment [4]. The widespread use of mobile devices, with increasing computational capacity, is leading to the emergence of the ubiquitous computing paradigm. This paradigm facilitates continuous access to data and information by mobile users with handheld devices [5]. UDM is the process of analyzing data from distributed and heterogeneous sources with mobile devices or within sensor networks, where the data is continuously streamed to the device, and where there are temporal constraints, that necessitate analysis ''anytime, anywhere'' [6].

In the following subsections preprocessing required for data streams is presented.

Table 1 Taxonomy of data mining environment.

Data localization	
Centralized	A single entity can access every data
Distributed	Each node can access just a part of the data
Homogeneous	. . . data related to the same entity (e.g. people) are owned by just one node
Heterogeneous	. . . data related to the same entity (e.g. people) may be spread among several nodes
Data evolution	
Statical	Data are definitively stored and invariable (e.g. related to some past and concluded event)
Incremental	New data are inserted and access to past data is possible (e.g. related to an ongoing event)
Evolving	The dataset is modified with either updates, insertions or deletions, and access to past data is possible
Streaming	Data arrives continuously and for an indefinite time. Access to past data is restricted to a limited part of them or summaries

2.1.1. Data streams techniques

Research problems and challenges that appeared in mining data streams have their solutions using well established statistical and computational approaches. These solutions could be categorized to data-based and task-based ones. This classification is depicted in Fig. 1 [7]. In data-based solutions, the idea is to examine only a subset of the whole dataset or to transform the data vertically or horizontally to an approximate smaller size data representation. On the other hand, in task-based solutions, techniques from computational theory have been adopted to achieve time and space efficient solutions.

2.1.1.1. Data-based techniques. Data-based techniques refer to summarizing the whole dataset or choosing a subset of the

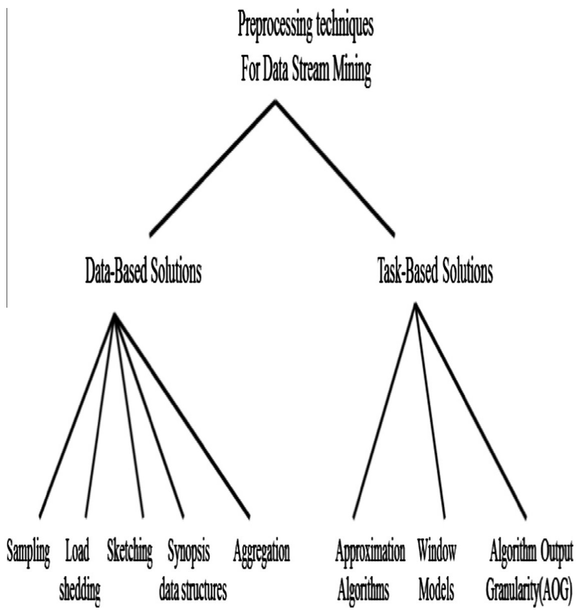


Figure 1 Classification of data streams preprocessing methods [7].

incoming stream to be analyzed. Sampling, load shedding and sketching techniques are most commonly used [7].

2.1.1.2. Task-based techniques. Task-based techniques are those methods that modify existing techniques,or invent new ones in order to address the computational challenges of data streams processing. Approximation algorithms, window models (landmark window model, damped window model, or sliding window model), and algorithm output granularity represent this category [7].

2.1.2. Data streams association rules

A number of algorithms have been proposed for extracting knowledge from streaming information. These include clustering, classification, frequency counting and time series analysis techniques. As the number of applications on mining data streams grows rapidly, there is an increasing need to perform association rule mining on stream data. An example application of data streams association rule mining is to estimate missing data in sensor networks [8]. Another example is to predict frequency estimation of Internet packet streams [9].

In the MAIDS project [10], this technique is used to find alarming incidents from data streams. Association rule mining can also be applied to monitor manufacturing flow to predict failure, or generate reports based on web log streams [11].

Several frequent pattern algorithms [12–18] were suggested in the literature. Most of these approaches either produce approximate results, discover special subsets (closed, maximal, and constraint-based) or provide exact results. These are compared in Table 2 [19].

2.2. Fuzzy association rules mining

Most studies have shown how binary valued transaction data may be handled. However, transaction data in real-world applications, usually consist of fuzzy and quantitative values. Thus, designing sophisticated data-mining algorithms, able to deal with various types of data presents a challenge to workers in this research field. Three Fuzzy Association rules algorithms [20–22] were recently proposed in the literature. These are compared in Table 3.

2.3. Need to extend existing work

Close study of Tables 2 and 3 reveals the need to develop a novel technique for mining fuzzy association rules from ubiquitous data streams.This technique should enjoy the following features:

– the ability of handling the continuous flow of data streams,
– the ability of handling data concept drift over time,
– the ability of handling memory bounded size, suitable for ubiquitous applications,
– facilitates data analysis and quick decision support for users, by scanning the flow of data streams only once,
– could determine fuzzy sets & membership functions,
– could determine user-defined min support, and
– gives accurate results.

In the following sections, an efficient Fuzzy Frequent Pattern Ubiquitous Streams technique (FFP_USTREAM), that satisfies the above features is proposed.

3. Proposed technique: FFP- USTREAM

The proposed technique consists of four steps, as indicated in Algorithm A.1. These are detailed below.

3.1. Step 1: Specifying sliding window

To handle continuously generated ubiquitous data streams, a sliding window model is used as shown in Algorithm A.2, to find exact recent fuzzy frequent patterns. The first step is to determine a sliding window size. Old transactions are expired,

Table 2 Comparative analysis of frequent pattern algorithms [19].

Algorithm name	Window model	Algorithm type	Updation rate	Merits	Limitations
Lossy counting [12]	Landmark	Approximate	Batch wise	No false negatives in results	Setting up of relaxed minimum support threshold leads to dilemma
Data Stream Mining Frequent Itemsets (DSM-FI) [13]	Landmark	Approximate	Batch wise	Compact tree structure has been designed to store the frequent patterns	It needs more tree traversals for the frequency count
Compact Prefix tree Structure (CPS) [14]	Sliding window	Exact	Batch wise	It maintains the frequency count lists at the last node, which can reduce the size of the prefix tree	Additional computational cost needed for restructing the tree after every pane insertion.
Frequent Pattern (FP-Stream) [15]	Tilted time window	Approximate	Batch wise	It extracts complete set of frequent patterns using time sensitive data streams	FP-Stream tree becomes very large with time
Weighted sliding Window (WSW) [16]	Sliding window	Exact	Batch wise	A single pass algorithm was developed to discover the frequent itemsets	Weights of each window affected the mining results. So, user should specify the reasonable weight for each window
Weighted Support Frequent Pattern (WSFP) [17]	Sliding window	Exact	Transaction wise	It collects the important recent frequent patterns with limited memory space	Initial setting of normalized minimum and maximum weight is given as random
Variable Sliding Window VSW [18]	Sliding window	Exact	Batch wise	Obselete transactions are deleted with respect to calculated formula	The prefix tree becomes very large where there is no changes occurred in the frequent patterns in processing

Table 3 Fuzzy association rules algorithms.

Algorithm	Mining task	Advantages	Disadvantages
F-APACS [20]	– Transformed quantitative attribute values into linguistic terms – Used the adjusted difference analysis to find interesting associations among attributes – Based on the a priori approach	– Discover both positive and negative associations – Avoid the use of some user-supplied thresholds	– Membership function must to be given – Cause iterative database scans – High computational costs – Does not handle data streams
Fuzzy Frequent Pattern Tree (FFPT) [21]	– Update the fuzzy sets at all the phases of the algorithm – Preserve the original information of the data sets – Based on Frequent Pattern Tree approach	– Avoid the candidate generation phase – Avoid the repetitive scanning of the original database	– Could not generate the membership value of an itemset – Local frequent fuzzy regions were used to construct the FFPT – Tree structure was loose and huge – Does not handle data streams
Compressed Fuzzy Frequent Pattern Tree (CFFPT) [22]	– Integrates the fuzzy-set concepts and the FP-tree-like approach to efficiently find the fuzzy frequent itemsets from the quantitative transactions – Create CFFP-tree – CFFP-growth mining algorithm is proposed to derive fuzzy frequent itemsets from the constructed CFFP tree	– The node number in the tree can be reduced – The CFFP tree Structure is tighter and more compressed – Can process quantitative transactions	– Additional array stored in each node – The proposed approach will spend more execution time because the former will need to transfer the quantitative values into fuzzy sets – The database is assumed static – Minimum support threshold is given – Does not handle data streams

once the new transactions arrive into the current window. The size of the window depends on the application and the system resources [19]. In this work, the size of window is considered constant.Variable window size is left for future work.

An example of sliding window in fuzzy data streams is depicted in Fig. 2. In this example, the window size (number of panes) and pane size (number of transactions) are set to 2. The notation (A:5) indicates the number of purchased units of item A.

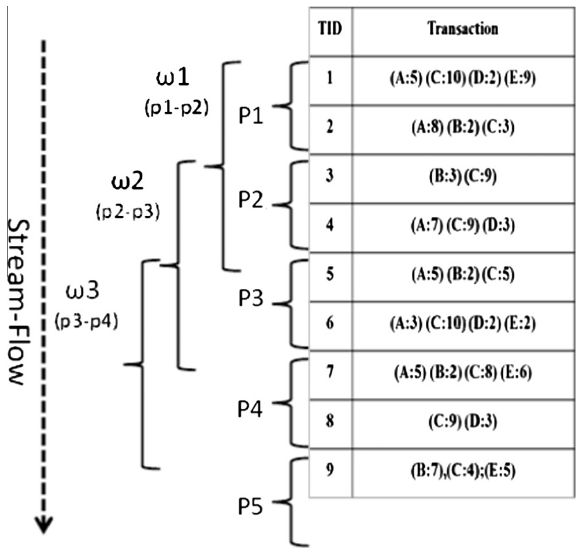
3.2. Step 2: Fuzzification

Fuzzification of quantative values of data streams attributes involves two processes. Firstly, derive the fuzzy sets for variables and represent them with linguistic terms [23]. Secondly, estimate the membership functions [24]. In practice, membership functions can have multiple different types, such as triangular waveform,trapezoidal waveform, Gaussian waveform, bell-shaped waveform, sigmoidal waveform, and S-curve waveform. In this work triangular and trapezoidal waveforms are used.

3.3. Step3: Tree structure construction

In order to infer Fuzzy Association Rules from ubiquitous data streams, Dynamic Fuzzy Frequent Pattern tree (DFFP-tree) is constructed. The basic structure of DFFP-tree is similar to Frequent Pattern tree (FP-tree) [25]. However, a membership value is added to each node as shown in Algorithm A.3.

In order to render this technique suitable for limited resources ubiquitous devices, an Algorithm A.4 is added to delete old panes, and insert new panes (i.e. restructuring trees), as windows are changed.



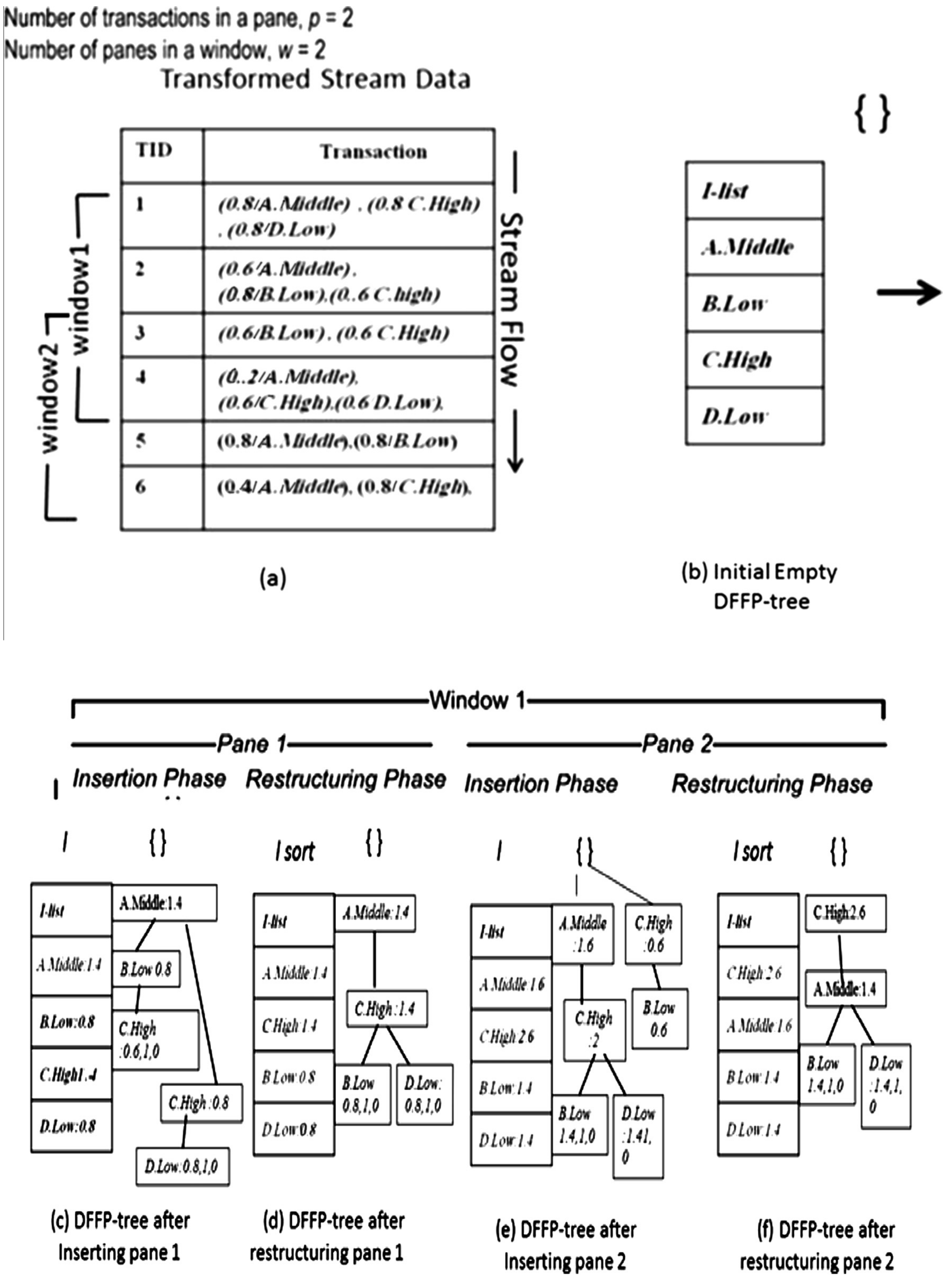
TID	Transaction
1	(A:5) (C:10) (D:2) (E:9)
2	(A:8) (B:2) (C:3)
3	(B:3) (C:9)
4	(A:7) (C:9) (D:3)
5	(A:5) (B:2) (C:5)
6	(A:3) (C:10) (D:2) (E:2)
7	(A:5) (B:2) (C:8) (E:6)
8	(C:9) (D:3)
9	(B:7),(C:4);(E:5)

Figure 2 Sliding window-based data streams.

Step by step construction procedure is shown in Fig. 3. In addition a comparative analysis between DFFP-tree and previous tree structures is illustrated in Table 4.

3.4. Step 4: Extracting fuzzy association rules

Once the DFFP-tree is constructed, Algorithm A.5 is used to generate the complete set of exact (not approximate) fuzzy frequent patterns from the current window. Hence, fuzzy


Number of transactions in a pane, $p = 2$
Number of panes in a window, $w = 2$

Transformed Stream Data

TID	Transaction
1	(0.8/A.Middle) . (0.8 C.High) . (0.8/D.Low)
2	(0.6/A.Middle). (0.8/B.Low),(0..6 C.high)
3	(0.6/B.Low) . (0.6 C.High)
4	(0..2/A.Middle). (0.6/C.High).(0.6 D.Low).
5	(0.8/A.Middle).(0.8/B.Low)
6	(0.4/A.Middle). (0.8/C.High).

Figure 3 Dynamic Fuzzy Frequent Pattern tree (DFFP-tree) construction.

Table 4 Comparative analysis between DFFP-tree and previous tree structures.

	No. of data base scans	Data sets types	Item values	Modules required
Frequent Pattern Tree (FP-tree)	More than one	Static Database	Crisp	• FP-Tree Construction • FP-Mining Association Rules
Dynamic Frequent Pattern Tree (DFP-tree)	Only one	Data Streams	Crisp	• Sliding Window • DFP-Tree Construction • DFP-Tree Restructuring • DFP-Mining Association Rules
Fuzzy Frequent Pattern Tree (FFP-tree)	More than one	Static Database	Fuzzy	• Fuzzy Sets Transformation • FFP-Tree Construction • FFP-Mining Association Rules
Dynamic Fuzzy Frequent Pattern Tree (DFFP-tree)	Only one	Data Streams	Fuzzy	• Sliding Window • Fuzzy Sets Transformation • DFFP-Tree Construction • DFFP-Tree Restructuring • DFFP-Mining Association Rules

Table 5 Dataset characteristics.

Data set name	No. of trans.	No. of items	Total length	Average length %
Accidents [26]	30571	342	3164207	30.2%
Retail [27]	29387	13737	1362940	0.34%

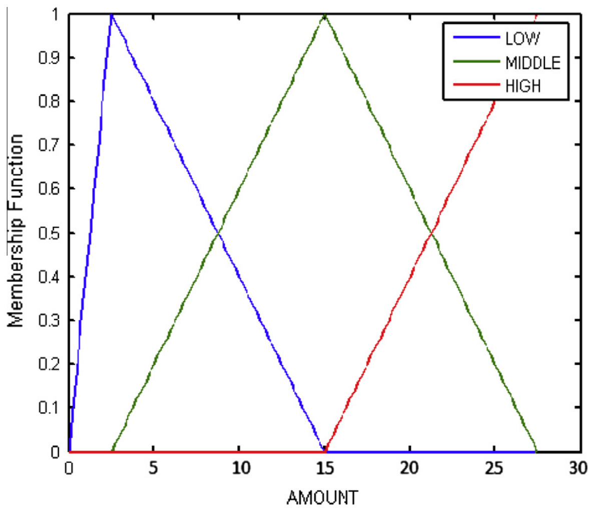


Figure 4 Membership functions with three Regions Fuzzy Sets.

association rules are deduced according to a certain confidence using Algorithm A.6.

4. Discussion of the complexity of the Algorithms

In this section, the complexity of FFP-USTREAM is analyzed. The following notations are defined for this discussion.

P, is the number of transactions in each window.

S, is the number of fuzzy sets.

I, is the number of distinct attributes relative to the study.

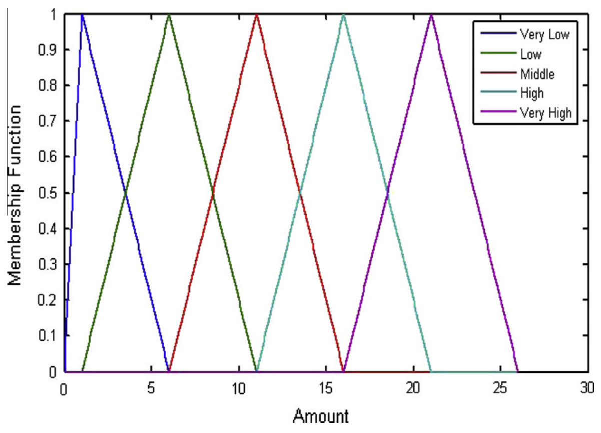


Figure 5 Membership functions with five Regions Fuzzy Sets.

The discussion is separated into two parts: runtime complexity, and memory complexity.

4.1. Runtime complexity

It may be shown that the run time complexity is divided into three major steps.

– Fuzzification process, which is the mapping of attributes quantitative values to fuzzy sets. The number of mapping will be of order $O(|S|*|P|*|I|)$ maps.
– DFFP-tree construction, it is the core of the proposed technique. The complexity will be of order $O(|P|*|I|)$, representing updating node value, or creating new node.
– Extracting the fuzzy association rules, its complexity will be of order $O(|P|*|I|)$, representing number of comparison.

4.2. Memory complexity

The memory complexity varies with the window size and the order of items in data streams transactions. Assuming worst-case scenario, number of nodes in DFFP-tree may be derived to be of order $O(|P|*|I|)$.

Table 6 Runtime distribution with variation of fuzzy regions.

Data sets	Window size	Pane size					
		20		50		100	
		Runtime in sec.		Runtime in sec.		Runtime in sec.	
		Three fuzzy regions	Five fuzzy regions	Three fuzzy regions	Five fuzzy regions	Three fuzzy regions	Five fuzzy regions
Accidents (Dense)	2	3.4788	5.1387	8.3928	10.23	16.4872	19.152
	4	6.552	10.24	16.38	20.34	32.532	35.858
	8	13.1678	15.542	32.5574	35.94	65.0862	71.988
Retail (Sparse)	2	0.421	1.1644	1.53252	2.3394	4.09	5.14
	4	1.404	2.3213	2.5741	4.8373	8.2	9.677
	8	3.0199	3.276	8.2294	8.651	15.852	16.618

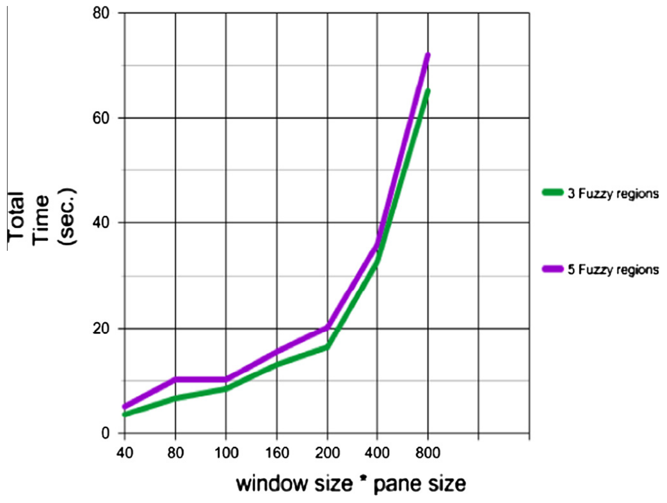


Figure 6a "Accidents" runtime distribution on variation of fuzzy regions.

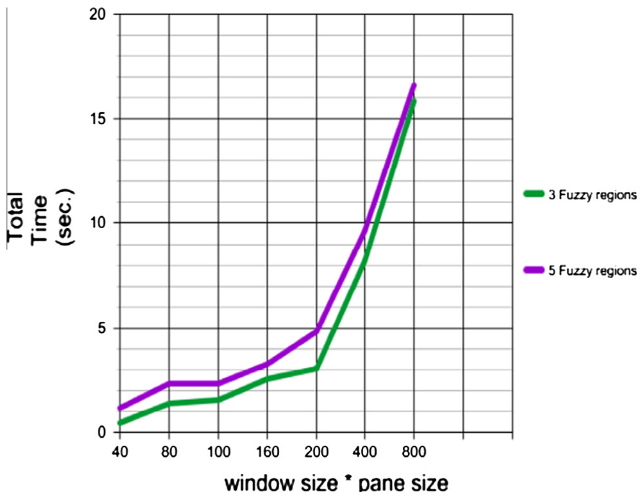


Figure 6b "Retail" runtime distribution on variation of fuzzy regions.

5. Experimental results

To assess the validity of the proposed technique, experiments are conducted using real data sets, in Microsoft Visual Studio 2010 C#. Two different data sets were chosen, "Accidents" and "Retail" data sets. These were downloaded from fimi.cs.helsinki.fi/data/ and summarized in Table 5.

5.1. Required parameters

Two main parameters have great effect on the experimental results performance, the selection of window size & pane size, and the selection of an appropriate membership function with fuzzy sets regions.

5.1.1. Window & pane size

DFFP-tree dynamically restructures itself after each window slide. The runtime and memory complexity vary depending on the window parameters *w* (window size) and *p* (pane size), as mentioned in Section 4 above and illustrated in Section 5.2.

5.1.2. Membership functions & fuzzy sets

An important step in FFP_USTREAM, was the selection of the membership functions and the number of fuzzy sets. Thus, an appropriate membership function must be selected carefully depending on the nature of the data sets and the user application.In this work, Fuzzy Logic Tool Box in Matlab 7 was used.

Table 7 Node distribution.

Data sets	Window size	Pane size								
		10			20			50		
		Mid.	Leaf	Total	Mid.	Leaf	Total	Mid.	Leaf	Total
Accident	2	124	134	258	135	251	386	154	484	638
	4	135	251	386	149	411	560	164	750	914
	8	149	411	560	161	658	819	183	1155	1338
Retail	2	105	27	132	211	54	265	453	178	631
	4	211	54	265	405	143	548	879	451	1330
	8	405	143	548	691	322	1013	1570	1168	2738

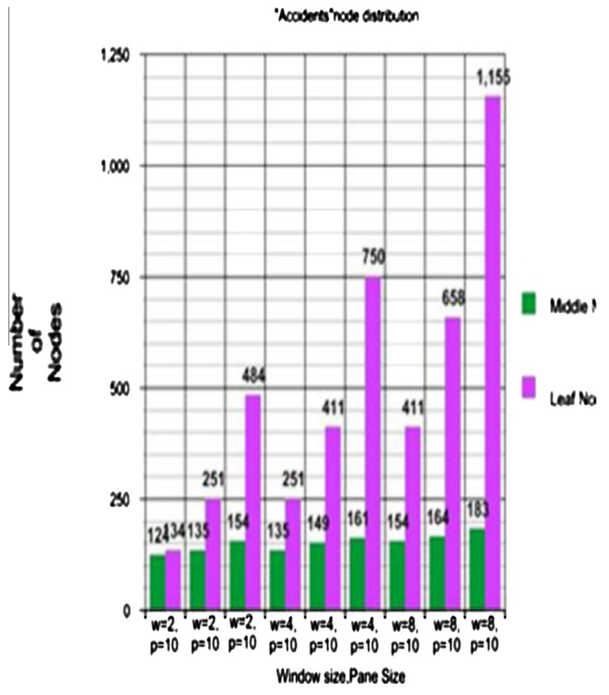


Figure 7a "Accidents" node distribution.

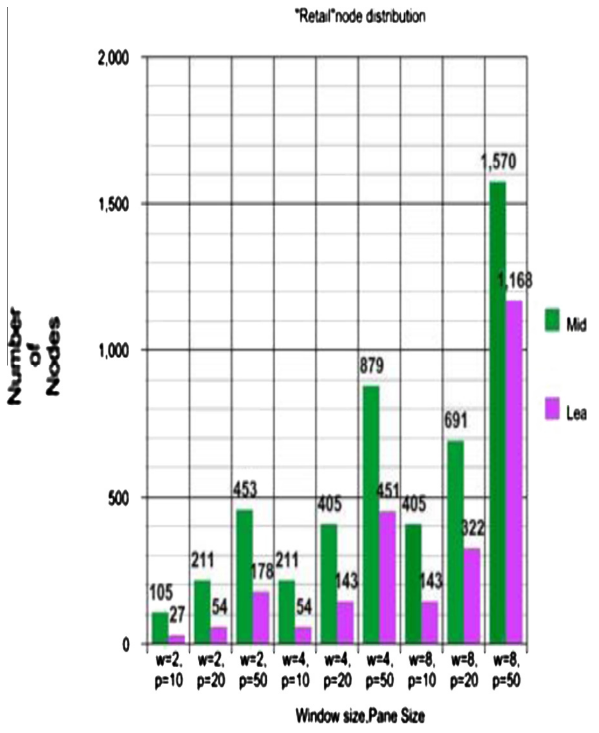


Figure 7b "Retail" node distribution.

In Fig. 4, the membership function with three fuzzy regions ('LOW', 'MIDDLE', 'HIGH') is presented. The membership function with five fuzzy regions ('VERYLOW', 'LOW', 'MIDDLE', 'HIGH', 'VERY HIGH') is also presented in Fig. 5.

5.2. Analysis of results

In the following sections, the results of DFFP-tree construction are presented. Extensive experimental analyses show that DFFP-tree is highly efficient in terms of memory storage when finding exact fuzzy frequent patterns from a high-speed ubiquitous data streams.

5.2.1. Runtime trend

For each data sets, the overall runtime,of DFFP-tree, based on changes in the fuzzy set regions was evaluated. The results are depicted in Table 6 and Fig. 6. In the graphs, y-axes represent the total time including DFFP-tree construction time and tree restructuring time. The x-axes show the variation of the product of number of window size and pane size.

A close study of Fig. 6 reveals the following trends.

- Larger pane and window sizes imply longer total tree construction & restructuring times for the two types of data sets (Dense & Sparse).
- The number of fuzzy sets regions affects the tree construction runtime. A long runtime will be required with large fuzzy sets regions.
- Higher runtime will be required for dense data sets as compared to sparse data sets.

5.2.2. Memory efficiency

Experiments have been conducted to verify the memory requirements for DFFP-tree construction on different data sets by varying the window size. FFP_USTREAM always captures the window contents in full (i.e., not depending on support

Table 8 Fuzzy frequent patterns.

Data sets	Fuzzy frequent patterns				
	1-Itemsets		2-Itemsets		
	Items	Count	First items	Second items	Count
Accidents	1.Very High	3	1.Very High	2.Low	3.8
	10.High	4.6	7.Low	5.High	3.2
	24.Very High	8	8.Very Low	8.High	3.2
	25.Very High	9.8	9.Very High	12.Medium	9
	26.Very High	9.8	13.Very High	13.Medium	9
	27.Very High	11.2	15.Very High	15.Medium	9.4
	28.Very High	2.2	17.Very High	16.Medium	9.4
	29.Very High	2.6	43.Very High	41.Medium	7.4
Retail	0.Very High	1	1.Low	0.Very High	0.8
	1.Very High	1	2.Low	1.Very High	0.6
	10.Very Low	0.6	4.Low	2.Very High	0.6
	2.Very High	4.8	5.High	3.Very High	0.6
	22.Low	7.2	12.Medium	4.Very High	0.8
	23.High	4	18.Very High	5.Very High	0.8
	24.High	3.4	19.Very High	39.Very High	2.8
			20.Very High	48.Very Low	1.6
			21.Very High	49.Very Low	2

threshold) in order to minimize the database scan to only once. Thus, the support threshold values do not influence the required memory in this approach. The number of nodes in DFFP-tree with the window size and pane size variations, is depicted in Table 7. Table 7 is represented by Fig. 7.

It should be notated that the number of nodes is constant, with the variation of window size for different data sets.

5.2.3. *Fuzzy frequent patterns list*

After the construction of DFFP-tree, fuzzy association rules algorithm has been applied to extract exact fuzzy frequent patterns. Data sets mentioned in Table 5 and membership functions with five regions fuzzy sets ("Very Low", "Low", "Medium", "High", "Very High") mentioned in Fig. 5 are used. Table 8 represents a sample of the final fuzzy frequent 1-itemsets and 2-itemsets with their total scale cardinality in all transactions calculated as "count" value for pane size = 20. The complete sets are not presented for lack of space. Predefined minimum support for "Accidents" data sets is selected = 5% where it is = 2.5% for "Retail" data sets. Hence, fuzzy association rules could be deduced as shown in Table 9.

Table 9 Fuzzy association rules.

Data sets	Fuzzy association rules
Accidents	– The Very High value of item: 17 ⇒ Medium value of item: 16
	– The Very High value of item: 9 ⇒ Medium value of item: 12
	– The Very High value of item: 43 ⇒ Medium value of item: 41
	– The Very Low value of item: 1 ⇒ Low value of item: 2
Retail	– The Very High value of item: 19 ⇒ Very High value of item: 39
	– The Very High value of item: 21 ⇒ Very Low value of item: 49
	– The Very High value of item: 20 ⇒ Very Low value of item: 48
	– The Low value of item: 1 ⇒ Very High value of item: 0
	– The High value of item: 5 ⇒ Very High value of item: 3

Note that the numbers written before Linguistic terms represent weather conditions (sunny, cloudy, etc.) for "Accidents" data sets, where,they represent purchased items (milk, bread, eggs,...) for "Retail" data sets.

6. Conclusion & future work

To the authors' knowledge,there are no existing techniques in the literature, that mine fuzzy association rules from a high-speed ubiquitous data streams. In this paper, an efficient technique suitable for ubiquitous applications, and satisfying the features mentioned in Section 2.3, was proposed. Also, a novel tree structure Dynamic Fuzzy Frequent Pattern tree (DFFP-tree) that combines fuzzy frequent pattern tree with the concept of dynamic tree restructuring (i.e. deleting old panes, and inserting new ones) at runtime was introduced. The proposed FFP_USTREAM technique could be very helpful in many practical situations for managers to make more significant and flexible decisions such as.

– Determining stock required in retail applications.
– Determining methods of treatment in medical applications.
– Determining methods of precaution in road safety applications.

The following are some suggestions for possible future work.

– The current FFP_USTREAM is based on the assumption that fuzzy sets and membership function are generated in triangular and trapezoidal waveforms only. Other waveforms such as Gaussian, bell-shaped, sigmoidal, and S-curve should also be considered depending on the applications.

– In this paper, window size & pane size were assumed constant. It is worthwhile to consider variable window and pane size.
– The proposed FFP_USTREAM handles distributed homogenous data streams. Thus, there is a need to investigate the cases where the distributed flow of data streams are heterogeneous, and the data sets are incompatible.
– The problem of mining fuzzy association rule from ubiquitous data streams can be introduced using other window techniques like tilted-window or land mark window.

Appendix A. Algorithms used in the proposed technique

Algorithm A.1. FFP_USTREAM.

```
Input: Data Streams (DS), window size ($\omega$), pane size ($p$),
minimum support ($s$), confidence ($c$), fuzzy set region ($R$) and
membership function ($f$).
Output: A set of fuzzy frequent patterns (FFP).
        A set of fuzzy association rules (FAR).
Method: FFP_USTREAM Algorithm is implemented as follows.
Begin
    (1) Create Sliding Window ().
    (2) Fuzzy Set Transformation ().
    (3) Dynamic Fuzzy Tree construction ().
    (4) Find Fuzzy Association Rules ().
END
```

Algorithm A.2. Create sliding window module.

```
Create_Window($T$,$Size_w$,$Size_p$)
Input:   Data Streams $DS$, window size $Size_w$, pane size $Size_p$;
Output: Data Streams Window Transaction T.
Method:  Create Sliding Window Algorithm is implemented as
follows.
Begin
1-  w = ø, p = ø;
2-  If T = ø Then    // First Window
3-      Call Insert Pane ( );
4-  Else
5-      Call Delete Pane ( ); // Next Window
6-      Call Insert Pane ( );
7-  End If
End
 Inset Pane ($T, Size_w, Size_p$)
 Begin
1-  $\omega$ = ø, p = ø;
2-  While $\omega \neq Size_w$, do
3-      While $p \neq Size_p$, do
4-  insert a transaction of $DS$ into $T$ according to its incoming
order;
5-          $p \leftarrow p + 1$;
6-      End while
7-    $p$ = ø;
8-    $\omega \leftarrow \omega + 1$;
9-  End While
 End
Delete Pane ($T, Size_w, Size_p$)
Begin
1-    $\omega$ = ø, p = ø;
2-    While $\omega \neq Size_w$, do
3-      While $p \neq Size_p$, do
4-      Delete transactions from $T$;
5-          $p \leftarrow p + 1$;
6-      End while
7-    $p$ = ø;
8-    $\omega \leftarrow \omega + 1$;
9-  End While
End.
```

Algorithm A.3. Construction of a dynamic fuzzy frequent pattern tree DFFP-tree.

```
Create_DFFP – tree($UDS, Size_w, Size_p$)
Input: The updated data streams (After applying fuzzy set
transformation), Window_size w, pane size p.
Output: T: a Dynamic fuzzy frequent pattern tree (DFFP-tree).
Method: The Dynamic Fuzzy Frequent Pattern tree construction
algorithm is implemented as follows.
Begin
1-    w ← Ø;
2-    T ← a prefix tree with null initialization;
3-    Current_sort_order ←Initial_Sort_order;
4-      While (w ≠ window_size) do // For the first Window
5-      Call Insert_Pane(T); // Insertion Phase
6-      Current_Sort_Order←Frequency_descending sort
order;
7-         Call Restructure ( T,I_list); // Restructuring Phase
8-         w = w + 1;
9-      End While
10-       Repeat //At each slide of window.
11-          Delete the oldest pane information from T;// Delete
old pane.
12-          Call Insert_Pane(T);// Insertion Phase
13-          Current_Sort_Order← Frequency_descending sort
order;
14-          Call Restructure (T,I_list);// Restructuring Phase
End
Insert_Pane (T)
Begin
1-  p← ø;
2-  While (p ≠ pane_size) do
3-      Insert transaction into T according to
Current_Sort_Order;
4-      p = p + 1;
```

```
5-  End While
End
```

Algorithm A.4. Restructure DFFP-tree.

```
Restruct_DFFP-tree (T, I)
Input: Prefix tree T, I list.
Output: Restructure tree T_sort, I_sort.
Method: The Dynamic Fuzzy Frequent Pattern tree restructure
algorithm is implemented as follows.
Begin
1-      For each branch B_i in T
2-         For each unprocessed path P_i
3-            If P_i is sorted then
4-               S_Branch ( );
5-            Else
6-               Sort_Path ();
7-            End if
8-         End For
9-      End For
End
Process_Branch (T)
Begin
1-      For each path, pi in T, do
2-         If pi is not sorted, do
3-            extract and sort pi depending on Order;
4-      reinsert pi into T and set pi's tail node information again;
5-      End If
6-      End For
End
Sort_path(T, I_sort)
Begin
1.      For each path, pi in the old pane, do
2.         find a tail node, ti for pi;
3.         For each node, nk in pi, do//Bottom up manner
4.            nk.val= nk.val ti.val.first;
5.            If nk.val  = 0, do
6.               delete nk;
7.            Else
           shift pane counters of all remaining tail nodes in T
           to left by one;
8.               End If
9.            End For
10.             For each transaction, tri of the new pane in
                   $DS, do/1 \leqslant i \leqslant Sizep$
11.             insert tri into according to Order;
12.             set tail node information for tri;
13.                End For
14.      End For
End.
```

Algorithm A.5. Extract exact fuzzy frequent patterns.

```
Extract_FFP(T)
Input: The constructed DFFP tree, Header_Table, and the
predefined minimum support threshold s.
Output: The Fuzzy Frequent Patterns list I_fp.
Method: The algorithm is implemented as follows.
Begin
3.      For each fuzzy region in Header_table
4.      Process the fuzzy regions Rj;
5.      Find all the nodes with Rj in the DFFP tree through the
links;
6.      For each node K extracted in DFFP
7.      Extract the fuzzy itemsets and their membership values
stored in K;
8.      Sum the membership values;
9.      Sum_val = Sum_val + member_val;
10.        End For
11.        If Sum_val > = minimum_support (s) then
12.        Add Rj in I_fp list;
13.        End If 14.      Output I_fp as fuzzy frequent pattern;
15.        End For
End
```

Algorithm A.6. Deduce fuzzy association rules.

```
Extract_FAR(I_fp,conf)
Input: I_fp list.
Output: The Fuzzy Association rules list I_ar.
Method: The algorithm is implemented as follows.
Begin
10-      For each Item Ii in I_fp.
11-      For each Item I_j in I_fp.
12-      Rule_val = ∪(I_i Sum_val, I_j Sum_val)
13-      If Rule_val > = Confidence (c) then
14-         Add Ii,Ij and Rule_val in I_ar list;
15-      End If
16-      Output I_ar as fuzzy association rules;
17-      End For
End
```

References

[1] M.M. Gaber, S. Krishnaswamy, A. Zaslavsky, "Ubiquitous Data Stream Mining," Current Research and Future Directions Workshop Proceedings held in conjunction with PAKDD, Sydney, Australia, May 26, 2004.

[2] M. Hellmann, Fuzzy logic introduction, Université de Rennes 1, 2001.

[3] M. May, B. Berendt, A. Cornuejols, J. Gama, F. Giannotti, A. Hotho, D. Malerba, E. Menesalvas, K. Morik, R. Persen, et al, Research challenges in ubiquitous knowledge discovery, Next Generation Data Min. (2008) 131–150.

[4] Z. urRehman, M. Shahbaz, M. Shaheen, S. Mehmood, S.A. Masood, Anomalous pattern detection using context aware ubiquitous data mining, Life Sci. J. 9 (3) (2012) 6–12.

[5] J. Gama, Data Stream Mining: The Bounded Rationality, Informatica, Slovenia 37 (1) (2013) 21–25.

[6] H.Kargupta, B.Park, S.Pittie, L.Liu, D.Kushraj, and K.Sarkar, MobiMine: Monitoring the Stock Market from a PDA,. ACM SIGKDD 2002.

[7] O.Z. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, vol. 1, Springer, New York, 2005.

[8] M. Halatchev, Le Gruenwald, Estimating missing values in related sensor data streams, in: Int'l Conf. on Management Data, Jan. 2005.

[9] E.D. Demaine, A. Lpez-Ortiz, J. Ian Munro, Frequency estimation of internet packet streams with limited space, in: European Symposium on Algorithms, Sept. 2002.

[10] Y. Dora Cai, G. Pape, J. Han, M. Welge, L. Auvil, MAIDS: mining alarming incidents from data streams, in: Int'l Conf. on Management Data, June 2004.

[11] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, D. Handy, VEDAS: a mobile and distributed data stream mining system for real-time vehicle monitoring, in: SIAM Int'l Conf. on Data Mining, 2004.

[12] G.S. Manku, R. Motwani, Approximate frequency counts over data streams, in: Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, August 2002, pp. 346–357.

[13] Hua-Fu Li, Suh-Yin Lee, Man-Kwan Shan, An efficient algorithm for mining frequent itemsets over the entire history of data streams, in: Proceedings of the First International Workshop on Knowledge Discovery in Data Streams, in Conjunction with the 15th European Conference on Machine Learning ECML and the 8th European Conference on the Principals and Practice of Knowledge Discovery in Databases PKDD, Pisa, Italy, 2004.

[14] S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, Y-K. Lee, Sliding window-based frequent pattern mining over data streams, Inform. Sci. 179 (2009) 3843–3865.

[15] C. Gianella, J. Han, J. Pei, X. Yan, P.S. Yu, Mining frequent patterns in data streams at multiple time granularities, in: H. Kargupta, A. Joshi, D. Sivakumar, Y. Yesha (Eds.), Data Mining: Next Generation Challenges and Future Directions, MIT/AAAI Press, pp. 191–212.

[16] P.M. Tsai, Mining frequent itemsets in data streams using the weighted sliding window model, Expert Syst. Appl. 36 (2009) 11617–11625.

[17] Y. Kim, W. Kim, U. Kim, Mining frequent itemsets with normalized weight in continuous data streams, J. Inform. Process. Syst. 6 (1) (2010) 79–89.

[18] M. Deypir, M.H. Sadreddini, S. Hashemi, Towards a variable size sliding window model for frequent itemset mining over data streams, Comput. Ind. Eng. 63 (2012) 161–172.

[19] P.G. Student, I. Madurai, Survey on frequent pattern mining over data streams, Int. J. Eng. 2 (12) (2013).

[20] Chan, C.C. Keith, A.U. Wai-Ho, Mining fuzzy association rules, in: Proceedings of the Sixth International Conference on Information and Knowledge Management, ACM, 1997.

[21] S. Papadimitriou, S. Mavroudi, The fuzzy frequent pattern tree, in: The WSEAS International Conference on Computers, July 2005, pp. 1–7.

[22] C.W. Lin, T.P. Hong, W.H. Lu, An efficient tree-based fuzzy data mining approach, Int. J. Fuzzy Syst. 12 (2) (2010) 150–157.

[23] C.W. Lin, T.P. Hong, W.H. Lu, Linguistic data mining with fuzzy FP-trees, Expert Syst. Appl. 37 (6) (2010) 4560–4567.

[24] S. Medasani, J. Kim, R. Krishnapuram, An overview of membership function generation techniques for pattern recognition, Int. J. Approximate Reasoning 19 (1998) 391–417.

[25] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases, September 1994, pp. 487–499.

[26] K. Geurts, Traffic accidents data set [online], 2003. < http://fimi.cs.helsinki.fi/data/accidents.pdf > .

[27] T. Brijs, Retail market basket data set, Workshop on Frequent Itemset Mining Implementations (FIMI'03), 2003.