# Mining Frequent Fuzzy Itemsets Using Node-List

Trinh T. T. Tran[1,2(✉)], Giang L. Nguyen[3], Chau N. Truong[4], and Thuan T. Nguyen[1,2]

[1] Graduate University of Science and Technology—VAST, Hanoi, Vietnam
thuytrinh85@gmail.com, nguyentanthuan2008@yahoo.com
[2] Duy Tan University, Da Nang, Vietnam
[3] Institute of Information Technology—VAST, Hanoi, Vietnam
nlgiang@ioit.ac.vn
[4] Danang University of Technology, Da Nang, Vietnam
truongngocchau@yahoo.com

**Abstract.** Data mining plays an important in knowledge discovery in databases; many types of knowledge and technology have been proposed for data mining. Among them, association rule mining is the problem important not only in data mining task but also in many practical applications in different areas of life. These previous studies mostly focused on showing the transaction data with binary values. However, in real-world applications, transactions also contain uncertain and imprecise data. To solve the above-mentioned problem, fuzzy association rule mining algorithms are developed to handle quantitative data using fuzzy set. In this paper, we present proposed algorithm NFFP, an improved fuzzy version of PPV algorithm for discovering frequent fuzzy itemsets using Node-List structure.

**Keywords:** Fuzzy frequent itemsets · Fuzzy sets · Node-List · Quantitative databases

## 1 Introduction

Owing to the speedy development of data, in many enterprises, enormous amount of data have been collected and stored in the database. In the fact, many important business information is hidden in data, so efficient tools are necessary for acquiring useful information. For this reason, data mining which is used to find information and knowledge from the incomplete, unclear data has developed with a variety of techniques. Since being introduced in [1], the technique of association rule mining (ARM) has received interest by the data mining community and a lot of researchers in the world. ARM detects interesting associations, correlations, frequent patterns, and relationships of data values in the transaction databases. One of them, frequent itemset


V. Bhateja et al. (eds.), *Information Systems Design and Intelligent Applications*, Advances in Intelligent Systems and Computing 672,
https://doi.org/10.1007/978-981-10-7512-4_5

mining (FIM), is an elemental task of this research field. The previous studies in [2–4] mostly focused on representing the transaction data of binary value that is only concerned about the presence or absence of these items. When dealing with quantitative data, it is difficult to discover the frequent itemsets with crisp values. To resolve this issue, the approach of fuzzy set has been applied to handle the quantitative databaseQuery.

The earlier algorithms used in mining fuzzy association rules are mostly fuzzy versions of the Apriori algorithm. With this approach, it has to scan database repeatedly to generate a large set of candidates and then check the support of them, so it is usually slow and ineffective in the case of huge database. The algorithm uses the principles of memory dependences as FFP_Growth [5–7] and MFFP_Tree [8] gain benefits over the Apriori approach finding out the frequent fuzzy itemsets without generating candidates. However, these strategies extract recursively frequent fuzzy itemsets from the tree structure throughout the entire process of algorithm; therefore, it tends to require big memories to store the temporal trees. In addition, with this approach, it must access database frequently; this will cause damage to the performance.

In this paper, the proposed algorithm NFFP uses the data structure called Node-List to extract the frequent fuzzy itemsets. In this approach, we only scan database two times: The first time is to convert the crisp value in quantitative database into fuzzy sets. The second time is to calculate the support of each fuzzy region in updated database and build the FPPC_tree. The tree is used to generate PP_code for each node by traversing the tree in pre- and post-order. After finishing its task, FPPC_tree will be deleted and the process of mining the fuzzy itemsets will be executed based on PP_-code, so it can reduce the memory usage requirements.

The remaining part of the paper includes the following: Part 2 reviews the basic concepts and related works, Part 3 presents the proposed algorithm NFFP, and Part 4 states the conclusion.

## 2 Basic Concepts and Related Work

### 2.1 Frequent Fuzzy Itemset Mining Problem

Let be a set of items, is the quantitative database of transactions. Each $T_q$ in QD is where TID is a transaction identifier and X is an itemset that contains several items with their quantities and is a minimum support threshold (minsup) and is membership function defined by user.

**Definition 1:** [9] Fuzzy set

Assume that where each (fuzzy term represented in natural language) is the element in the fuzzy set of *i* and are, respectively, their fuzzy values (defined by the membership function). The fuzzy set is presented as follows:

**Definition 2:** [9] The support of fuzzy item

The support of the fuzzy item denoted is defined as follows:

Where is the updated fuzzy database after converting quantitative values into fuzzy values.

**Definition 3:** [9] In this paper we uses τ—norm in fuzzy set such as $a$ τ $b$ are *min* ($a$, $b$)for finding the fuzzy support value. The support of a fuzzy k-itemset, or, is the minimum of the fuzzy values of the fuzzy items, is:

**Lemma 1** [8] If there are n transactions, then we have.

**Problem statement**

Frequent fuzzy itemset mining (FFIM) is the problem that extracts all frequent fuzzy itemsets as:

**Example 1:** Table 1 presents the quantitative database (QD) which will be used for illustration in this paper. QD has six transactions with five items indicated; the minsup is set to 30%. Assume that we use the triangular membership function for all items shown in Fig. 1, and quantities are set into one of fuzzy terms: Low, Middle, and High.

**Table 1.** Quantitative database

| TID | Items |
|---|---|
| 1 | ($I_1$:5) ($I_3$:10) ($I_4$:2) ($I_5$:9) |
| 2 | ($I_1$:8) ($I_2$:2) ($I_3$:3) |
| 3 | ($I_2$:3) ($I_3$:9) |
| 4 | ($I_1$:5) ($I_2$:3) ($I_3$:10) ($I_5$:3) |
| 5 | ($I_1$:7) ($I_3$:9) ($I_4$:3) |
| 6 | ($I_2$:2) ($I_3$:8) ($I_4$:3) |

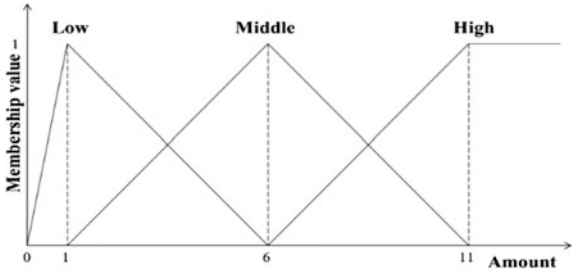


**Fig. 1.** Triangular membership function for all items

## 2.2 FPPC_Tree

The FPPC_tree is built on integrating of fuzzy concepts and PPC_tree [2] like approach.

**Definition 4:** FPPC_tree is a tree with a structure contains one root and a set of item prefix subtree; each node in a subtree comprises five elements: *f_fuzzy_term, f_support, children_list, fpre_code,* and *fpost_code,* which are the fuzzy items, the support of *f_fuzzy_term* in this node, list of child nodes, the pre- and post-order codes of the node, respectively.

The FPPC_tree_building algorithm is presented in Algorithm 1.

**Algorithm 1** (FPPC_tree_building)

**Input:** A quantitative database QD, membership function, and minsup .

**Output:** A FPPC-tree (*FTr*), the set of frequent fuzzy 1-itemset ().

1. Convert the value of each in transaction into a fuzzy set as in (1).

2. Scan updated database containing fuzzy values to compute the support of each fuzzy item in the transaction as in (2).

3. Check if , put the in . That is .

4. Sort the frequent fuzzy items in in support decreasing order.

5. If , delete from all

6. Generate the root of the FPPC_tree and label it as "null"

7. **for** each in **{**

8. Sort the remaining fuzzy items in support decreasing order.

9. Insert the fuzzy items into FFPC_tree (this process is similar to MFFP_tree [14])

10. **}**

11. Traverses FPPC-tree to generate the *PP_Code* of each node.

**Example 2:** In the illustrative example 1, the algorithm converts the value () of all items in QD into fuzzy values and then calculates the support of each fuzzy items in the updated database as given in Tables 2 and 3.

Then, the algorithm discards all fuzzy items, whose supports are less than the minsup, and sorts the rest of fuzzy items in support decreasing order (Table 4). The updated transactions with frequent fuzzy1-itemsets are presented in Table 5.

Next, the algorithm inserts the fuzzy items in each updated transaction into FPPC_tree. Lastly, the algorithm walks through the FPPC_tree (Fig. 2) to generate PP_code of each node.

**Table 2.** Fuzzy database after converting quantitative values into fuzzy values

| TID | Fuzzy items |
|---|---|
| 1 | $\left(\frac{0.2}{I_2, Low} + \frac{0.8}{I_1, Middle}\right), \left(\frac{0.2}{I_3, Middle} + \frac{0.8}{I_3, High}\right), \left(\frac{0.8}{I_4, Low} + \frac{0.2}{I_4, Middle}\right), \left(\frac{0.4}{I_5, Middle} + \frac{0.6}{I_5, High}\right)$ |
| 2 | $\left(\frac{0.6}{I_2, Middle} + \frac{0.4}{I_1, High}\right), \left(\frac{0.8}{I_2, Low} + \frac{0.2}{I_2, Middle}\right), \left(\frac{0.6}{I_3, Low} + \frac{0.4}{I_3, Middle}\right)$ |
| 3 | $\left(\frac{0.6}{I_2, Low} + \frac{0.4}{I_2, Middle}\right), \left(\frac{0.4}{I_3, Middle} + \frac{0.6}{I_3, High}\right)$ |
| 4 | $\left(\frac{0.2}{I_2, Low} + \frac{0.8}{I_1, Middle}\right), \left(\frac{0.6}{I_2, Low} + \frac{0.4}{I_2, Middle}\right), \left(\frac{0.2}{I_3, Middle} + \frac{0.8}{I_3, High}\right), \left(\frac{0.6}{I_5, Low} + \frac{0.4}{I_5, Middle}\right)$ |
| 5 | $\left(\frac{0.8}{I_2, Middle} + \frac{0.2}{I_1, High}\right), \left(\frac{0.4}{I_3, Middle} + \frac{0.6}{I_3, High}\right), \left(\frac{0.6}{I_4, Low} + \frac{0.4}{I_4, Middle}\right)$ |
| 6 | $\left(\frac{0.8}{I_2, Low} + \frac{0.2}{I_2, Middle}\right), \left(\frac{0.6}{I_3, Middle} + \frac{0.4}{I_3, High}\right), \left(\frac{0.6}{I_4, Low} + \frac{0.4}{I_4, Middle}\right)$ |

**Table 3.** Support of fuzzy items

| Fuzzy item | Support | Fuzzy item | Support | Fuzzy item | Support |
|---|---|---|---|---|---|
| $I_1$.Low | 0.4 | $I_3$.Low | 0.6 | $I_5$.Low | 0.6 |
| $I_1$.Middle | 3.0 | $I_3$.Middle | 2.2 | $I_5$.Middle | 0.8 |
| $I_1$.High | 0.6 | $I_3$.High | 3.2 | $I_5$.High | 0.6 |
| $I_2$.Low | 2.8 | $I_4$.Low | 2.0 | | |
| $I_2$.Middle | 1.2 | $I_4$.Middle | 1.0 | | |
| $I_2$.High | 0 | $I_4$.High | 0 | | |

**Table 4.** Resulted frequent fuzzy 1-itemsets based on support calculation

| Frequent fuzzy items | Support |
|---|---|
| $I_3$.High | 3.2 |
| $I_1$.Middle | 3.0 |
| $I_2$.Low | 2.8 |
| $I_3$.Middle | 2.2 |
| $I_4$.Low | 2.0 |

### 2.3 Node-List

Zhihong Deng and Zhonghui Wang [4] presented some important definitions of Node-List and some properties associated with it. We summarize and apply these concepts to our fuzzy values as shown below.

**Table 5.** Updated transactions with frequent fuzzy1-itemsets

| TID | Fuzzy items |
|---|---|
| 1 | , |
| 2 | , |
| 3 | ,, |
| 4 | , |
| 5 | , |
| 6 | ,, |

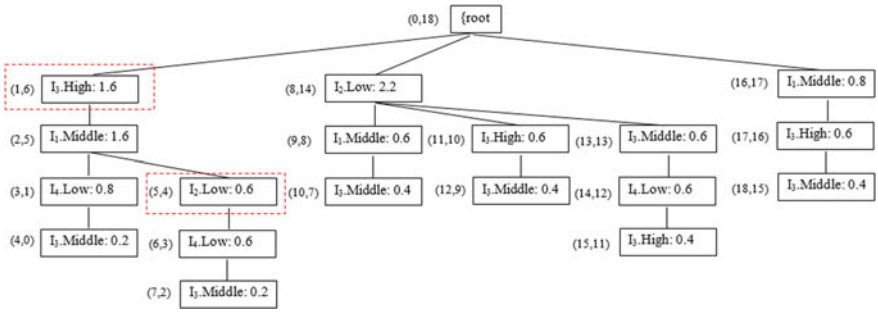


**Fig. 2.** FPPC_tree created from QD' with δ = 30%

**Definition 5:** The PP_code of each node in a FPPC_tree is $C_i$ = .

**Example 3:** The highlighted nodes $N_1$ and $N_2$ in Fig. 2 have the PP_codes: $C_1$ = < 1, 6, 1.6 > and $C_2$ = < 5, 4, 0.6>

**Property 1:** Node $N_1$ is called an ancestry of node $N_2$, if and only if and

**Definition 6:** Let $C_1$ be and $C_2$ be, $C_1$ is an ancestry of $C_2$ if and only if $pr_1 < pr_2$ and $po_1 > po_2$.

**Definition 7:** The Node-List of a fuzzy item is a chain of PP_codes of nodes representing the fuzzy items in the FPPC_tree. The PP_codes are sorted in *fpre_code* increasing order. Each PP_code is indicated by.

**Example 4:** The Node-List of $I_1$.*Middle* includes three nodes: Figure 3 shows the Node-List of fuzzy frequent 1-itemsets in the above example.

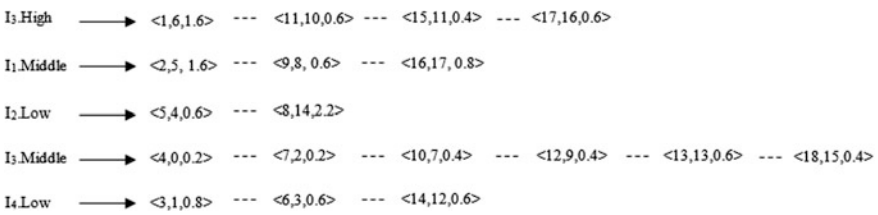


**Fig. 3.** Node-List of frequent fuzzy items

After building the FPPC_tree, we traverse the FPPC_tree in *fpre_code* order. For each node $N_i$, we insert into the Node-List of the item specified by $N_i$ and gain the Node-List of each fuzzy frequent 1-itemset. The construction of the Node-List of all frequent fuzzy 1-itemsets is shown in Algorithm 2.

**Algorithm 2: FNodelist_Generation**

**Input:** FPPC-tree (FTr) and $F_1$ (the list of frequent fuzzy 1-itemsets)

**Output**: $NL_1$ (the set of Node list of frequent fuzzy 1-itemsets)

1. Assume that $NL_1[k]$ is the node list of an $k^{th}$ element in $F_1$.

2. Traverse all nodes $N_i$ in the FPPC_tree by

3. **if** ($N_i$. *f_fuzzy_term* = $F_1[k]$. *f_fuzzy_term*)

4. insert into $NL_1[k]$;

5. **return** ;

**Property 2:** Given and, if $N_1$. *fpre_code* < $N_2$. *fpre_code*, then $N_1$. *fpost_code* < $N_2$. *fpost_code*.

**Proof:** If, by traversing across the tree, $N_1$ is traversed before $N_2$. But $N_1$ cannot be the ancestry of $N_2$ because they have the same *f_fuzzy_term*, so $N_1$ must locate the left side of $N_2$ on the tree. In the contrary traversal, we also traverse the tree from left to right, so.

**Property 3:** If the Node-List of fuzzy item is, then the support of a fuzzy item is.

**Example 5:** The Node-List of $I_1$. *Middle* includes, and the support of $I_1$. *Middle* is *3.0.* All fuzzy items are true to this property.

**Definition 8:** (relation) Given two frequent fuzzy items and (. when if and only if is before in.

## 3 Proposed NFFP Algorithm for Mining Frequent Fuzzy Itemsets

In this section, we introduce the NFFP algorithm and present two main contributions in this paper: (1) improve Node-List intersection of frequent fuzzy k-itemsets from [4] and (2) propose the new approach using Node-List structure to find the frequent fuzzy itemsets; this approach helps lessen the memory utilization requirements, since storing FPPC_tree during frequent fuzzy itemset mining process is not needed.

### 3.1 Node-List Intersection

In paper [4], the author proposed the Code_intersection method for determining the Node-List of k-itemsets which is only consistent with crisp values. In this case, PPC_tree is constructed according to itemsets arranged in their descending order of frequency. If $i_1$is before $i_2$in $L_1$then all nodes of $i_1$ are always ancestry of nodes of $i_2$. So author only check whether Node-List of is ancestry of Node-List of $i_2$. This work proposes an improved method that offers the Node-List intersection of frequent fuzzy k-itemsets to obtain the Node-List of frequent fuzzy (k + 1) itemsets.

**Definition 9:** (Node-List of frequent fuzzy 2-itemsets).
Suppose that two different fuzzy frequent items in which (is before in), and their Node-List respectively are: and. For any PP_code and. If and has the ancestry—descendant relationship of PP_Code, then insert the descending PP_Code into the Node-List of.

**Example 6:** The Node-List of two itemsets $I_1$.Middle and $I_2$.Low is shown in Fig. 4

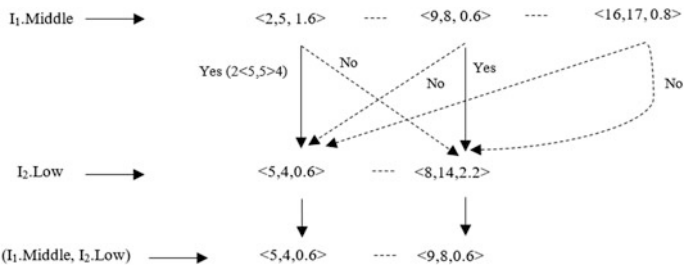


**Fig. 4.** Node-List of two items ($I_1$.Middle, $I_2$.Low)

Based on Definition 9, we generalize to the concept of the Node-List of a frequent fuzzy *k*-itemsets ($k \geq 3$) in below.

**Definition 10:** (Node-List of frequent fuzzy k-itemsets): Let be an frequent fuzzy itemsets (), the Node-List of be, the Node-List of be. For any PP_code and. If and has the ancestry—descendant relationship of PP_Code, then insert the descendant PP_Code into the Node-List of.

**Example 7:** We have known that the Node-List of ($I_1$.Middle, $I_2$.Low) and ($I_1$.Middle, $I_4$.Low) is < 5, 4, 0.6 >, < 9, 8, 0.6 > and < 3, 1, 0.8 >, < 6, 3, 0.6 >. The Node-List of ($I_1$.Middle, $I_2$.Low, $I_4$.Low) is shown in Fig. 5.

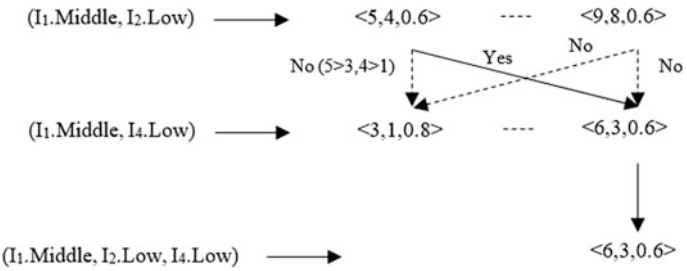


**Fig. 5.** Node-List of two items ($I_1$.Middle, $I_2$.Low, $I_4$.Low)

**Property 4:** For any Node-List of fuzzy k-itemsets, represented by, the support of P is.

**Proof:** With k = 1: In accordance with Property 3, the conclusion is true.
With: The designed algorithm sorts the fuzzy items in a in their supports decreasing order in each transaction for building the FPPC_tree structure and uses - norm in fuzzy set such as (as the intersection operator so we can obtain the minimum values between fuzzy regions from support of the child nodes. Given any fuzzy k-itemsets have the Node-List then according to Definitions 9 and 10, is a child node. So the support of P following the Definition 4 is.

The improved Node-List intersection method is presented in Algorithm 3.

**Algorithm 3: FNodelist_Intersection**

**Input**: and where , are the node list of two frequent fuzzy k-itemsets.

**Output**: , the node list of frequent fuzzy (k+1) itemsets.

```
1. for
2.          for
3.            if  {
4.                    if
5.                       insert  into ;
6.            }
7.            else {
8.                                 if
9.                              insert  into ;
10.                   }
11.         }
12. }
13. return
```

### 3.2 Mining Frequent Fuzzy Itemsets Using Node-List

According to the main idea of process sequence, the proposed NFFP algorithm includes four main steps: (1) construct FPPC tree and identify all fuzzy frequent 1-itemsets, (2) construct Node-List of fuzzy frequent 1-itemsets, (3) perform Node-List intersection of k-1-itemsets to build Node-List of k-itemsets, and (4) mining all frequent fuzzy itemsets.

**Algorithm 4: NFFP**

**Input**: A quantitative database QD, membership function, and minsup .

**Output**: The complete set of frequent fuzzy itemsets (FFIs)

1. Call *FPPC_tree_building (D,)* to generate FPPC – tree (R), $F_1$

2. Call *FNodelist_Generation (R, $F_1$)*;

3. Call *Find _FFI (, $F_1$, $NL_1$)*;

4. Return FFIs

**Procedure Find_FFI** (, $F_1$, $NL_1$)

1. **For do begin**
2. **For all** , **where** , {
3.
4. **If** all *k*-1 subsets *of P* are in {
5. = FNodelist_Intersection (, );

6. Calculate ; // Use Property 4
7. **If** {
8. ;
9. ;

10. **}**
11. }
12. }
13. Delete
14. }
15.

**Example 8:** Table 6 shows the final results of the frequent fuzzy itemsets in illustrative example.

**Table 6.** Final results of frequent fuzzy itemsets

| Fuzzy frequent 1-itemsets | Support | Fuzzy frequent 2-itemsets | Support |
|---|---|---|---|
| $I_3$.High | 3.2 | {$I_3$.High, $I_1$.Middle} | 2.2 |
| $I_1$.Middle | 3.0 | {$I_3$.High, $I_4$.Low} | 1.8 |
| $I_2$.Low | 2.8 | | |
| $I_3$.Middle | 2.2 | | |
| $I_4$.Low | 2.0 | | |

## 4 Conclusion

In this paper, the proposed algorithm NFFP uses a FPPC_tree to store the quantitative database with descending order membership values. Based on the FPPC_tree, we detect the Node-List of each frequent fuzzy item. Then, NFFP algorithm obtains Node-Lists of the frequent fuzzy (k + 1) itemsets by intersecting the Node-Lists of frequent fuzzy k-itemsets and then extracts the fuzzy frequent (k + 1)-itemsets. The advantage of this algorithm is that the FPPC_tree is used to generate pre–post code for each node to get the Node-List of each frequent fuzzy item, and after that, it will be deleted so it can reduce the memory usage requirements. However, suppose that the data used in this paper is static, in the real applications, data may be changed with time and automatically inserted into database. In the future works, we will attempt to solve this problem of fuzzy data mining.

## References

1. Agrawal R., Srikant R.: Fast algorithms for mining association rules. In proceedings of 20th International Conference on Very Large Databases, Santiago, Chile (1994).
2. Deng Zhihong, Wang ZhongHui and Jiang JiaJian: A new algorithm for fast mining frequent itemsets using N-lists. Science China Information Sciences, Vol. 55 No.9, (2012) 2008–2030.
3. Farah Hanna AL-Zawaidah, Yosef Hasan Jbara: An improved algorithm for mining association rules in large database. World of Computer Science and Information Technology Journal (WCSIT), Vol. 1, No. 7, (2011) 311–316.
4. Zhihong Deng, Zhonghui Wang: A new fast method for mining frequent patterns. International Journal of Computational Intelligence Systems, Vol.3, No. 6, (2010) 733–744.
5. De Cock, M., Cornelis, C., Kerre, E.E.: Fuzzy Association Rules A Two-Sided Approach. In: FIP, (2003) 385–390.
6. Reza Sheibani, Amir Ebrahimzadeh, Member, IAUM: An Algorithm For Mining Fuzzy Association Rules. Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I, (2008) 486–490.
7. Yan, P., Chen, G., Cornelis, C., De Cock, M., Kerre, E.E.: Mining Positive and Negative Fuzzy Association Rules. In: KES, Springer, (2004) 270–276.

8. Tzung-Pei Hong, Chun-Wei Lin and Tsung-Ching Lin: The MFFP-tree fuzzy mining algorithm to discover complete linguistic frequent itemsets. Computational Intelligence, Vol.0, No.0, (2012).
9. Chun-Wei Lin, Philippe Fournier-Viger, Tzung-Pei Hong: A fast algorithm for mining fuzzy frequent itemsets. Journal of Intelligent and Fuzzy Systems (2015).