

# An Efficient Tree-based Fuzzy Data Mining Approach

Chun-Wei Lin, Tzung-Pei Hong, and Wen-Hsiang Lu

## Abstract

**In the past, many algorithms were proposed for mining association rules, most of which were based on items with binary values. In this paper, a novel tree structure called the compressed fuzzy frequent pattern tree (CFFP tree) is designed to store the related information in the fuzzy mining process. A mining algorithm called the CFFP-growth mining algorithm is then proposed based on the tree structure to mine the fuzzy frequent itemsets. Each node in the tree has to keep the membership value of the contained item as well as the membership values of its super-itemsets in the path. The database scans can thus be greatly reduced with the help of the additional information. Experimental results also compare the performance of the proposed approach both in the execution time and the number of tree nodes at two different numbers of regions, respectively.**

**Keywords:** *fuzzy data mining, fuzzy set, quantitative value, CFFP trees, CFFP-growth, fuzzy frequent patterns.*

## 1. Introduction

Depending on the classes of the knowledge derived, the mining approaches may be classified as finding association rules [1, 3-4], classification rules [14, 25], clustering rules [16, 19] and sequential patterns [2, 23], among others. Many algorithms for mining association rules from transactions were proposed [1, 3] in the past. Most of the approaches were based on the Apriori algorithm [3], which generated and tested candidate itemsets level by level. This processing way might, however, cause iterative database scans and high computational costs. Han *et al.* thus proposed the Frequent-Pattern-tree (FP-tree) structure for efficiently mining association rules without generation of candidate

itemsets [10].

Fuzzy sets have recently been widely used in many applications and caused good effects. Several fuzzy learning algorithms for inducing rules from given sets of data have been designed [11-12]. Papadimitriou *et al.* proposed an approach based on the FP-tree structure to find fuzzy association rules [21]. Each item in the quantitative transactions was transferred into two fuzzy regions through predefined membership functions. Only the local frequent fuzzy regions were kept in each transaction for the later mining process. A tree structure called fuzzy frequent pattern tree (FFPT) was then constructed tuple by tuple from the first transaction to the last one. After the FFPT was constructed, the fuzzy association rules could be thus derived based on the tree. However, the fuzzy operations were not used in their approach to generate the membership value of an itemset. Besides, only the local frequent fuzzy regions were kept to construct the FFPT. The fuzzy regions which were close to but below the predefined minimum support threshold would not be used for the construction even though their summed membership values in the transactions were large. It seemed to lose the nature of fuzzy-set processing.

In the past, we proposed a fuzzy FP-tree structure to mine fuzzy association rule [17]. In that approach, the fuzzy regions in each transaction were sorted in a descending order of their fuzzy values in that transaction. Two transactions with the same fuzzy regions but different orders were put into two different paths of the tree. The tree structure was thus loose and huge. In this paper, a novel tree structure called the compressed fuzzy frequent pattern tree (CFFP tree) is designed to keep some related information for fuzzy data mining. It is compressed because two transactions with the same fuzzy regions but different orders will be put into the same path of the tree. The node number in the tree can thus be significantly reduced.

The proposed approach first transforms quantitative values in transactions into linguistic terms [11-12]. Each item uses only the linguistic term with the maximum cardinality in the later process, thus making the number of fuzzy regions processed equal to the number of the original items. Later, the kept linguistic terms are checked for whether their membership values are larger than or equal to the predefined minimum support threshold. The terms satisfying the above condition are

---

Corresponding Author: Tzung-Pei Hong is with the Department of Computer Science and Information Engineering, National University of Kaohsiung, 700, Kaohsiung University Rd., Nanzih District, 811, Kaohsiung, Taiwan.

E-mail: tphong@nuk.edu.tw

Manuscript received 15 Dec. 2009; revised 1 Feb. 2010; accepted 12 Mar. 2010.

set as the fuzzy frequent 1-itemsets. The CFFP tree is then constructed from the transactions based on the occurrence frequencies of the fuzzy frequent 1-itemsets, which makes the tree structure tighter and more compressed. Each node in the tree has to keep the membership value of the fuzzy frequent 1-itemset within it as well as the membership values of its super-itemsets in the path. Besides, the CFFP-growth mining algorithm is also proposed to derive fuzzy frequent itemsets from the constructed CFFP tree. The algorithm is efficient because the database scan time can be greatly reduced just like the FP tree.

## 2. Review of Related Works

In this section, some related researches are briefly reviewed. They are mining algorithms for fuzzy association rules and the FP-growth algorithm.

### A. Mining Algorithms for Fuzzy Association Rules

In the past, Srikant *et al.* proposed a quantitative association rule mining method to find association rules by partitioning the quantitative database and transforming the problem into binary one [24]. Besides, several mining approaches based on the fuzzy set theory have also been proposed. Chan *et al.* proposed an F-APACS algorithm to mine fuzzy association rules [5]. They first transformed quantitative attribute values into linguistic terms and then used the adjusted difference analysis to find interesting associations among attributes. Kuok *et al.* proposed a fuzzy mining approach to handle numerical data in databases and to derive fuzzy association rules [15]. At nearly the same time, Hong *et al.* proposed a fuzzy mining algorithm to mine fuzzy rules from quantitative databases [11-12]. Some related researches are still in progress [6-7, 20, 22]. Most of the above approaches are based on the Apriori approach. In this paper, we thus adopt the tree structure to speed up the execution of the fuzzy mining process. Because the transaction data has been stored in the tree structure, the database scan time can be greatly reduced.

### B. The FP-growth Algorithm

Han *et al.* proposed the Frequent-Pattern-tree structure (FP-tree) and FP-growth algorithm for efficiently mining frequent itemsets without generation of candidate itemsets [10]. The mining algorithm consisted of two phases. The first phase focused on constructing the FP-tree from a database, and the second phase focused on deriving frequent patterns from the FP-tree. Three steps were involved in FP-tree construction. The database was first scanned to find all the items with their counts. The items with their supports equal to or larger than a predefined minimum support were then selected

as frequent 1-itemsets (items). Next, the frequent items were sorted in a descending order of their frequencies. At last, the database was scanned again to construct the FP tree according to the sorted order of frequent items tuple by tuple from the first transaction to the last one.

After the FP tree was constructed from a database, the FP-growth mining algorithm was then executed to find all frequent itemsets [10]. A conditional FP tree was generated for each frequent item, and the frequent itemsets with the processed item could be recursively derived from the FP-tree structure. Several other algorithms based on the FP-tree structure have been proposed [8, 13, 18, 26].

## 3. The Proposed CFFP-tree Construction Algorithm

The proposed construction algorithm for building a CFFP tree from a quantitative database is described in this section. The notation used in the proposed algorithm is first stated below.

### A. Notation

$D$	a quantitative database;
$n$	the number of transactions in $D$ ;
$T$	the $i$ -th transaction in $D, 1 \leq i \leq n$ ;
$m$	the number of items in $D$ ;
$I_j$	the $j$ -th item, $1 \leq j \leq m$ ;
$v_{ij}$	the quantitative value of $I_j$ in $T_i$ ;
$h_j$	the number of fuzzy regions for $I_j$ ;
$R_{jl}$	the $l$ -th fuzzy region of $I_j, 1 \leq l \leq h_j$
$f_{ijl}$	the membership value of $v_{ij}$ in region $R_{jl}$ ;
$count_{jl}$	the count of the fuzzy region $R_{jl}$ in $D$ ;
$max-count_j$	the maximum count value among the fuzzy regions of $I_j$ ;
$max-R_j$	the fuzzy region of $I_j$ with $max-count_j$ ;
$o(max-R_j)$	the occurrence number of the fuzzy region $max-R_j$ ;
$s$	the predefined minimum support threshold.

### B. The Proposed Algorithm

The proposed approach integrates the fuzzy-set concepts and the FP-tree-like approach to efficiently find the fuzzy frequent itemsets from the quantitative transactions. The CFFP-tree construction algorithm is first designed to build the tree structure for the fuzzy frequent 1-itemsets. Each node in the tree structure keeps a fuzzy frequent 1-itemset, its membership value, and the membership values of its super-itemsets in the path according to the intersection operator, which is the minimum operator here. An array called *minval\_Ary* is then attached to a node to keep the membership values

of its super-itemsets in the path. The CFFP-tree construction algorithm is stated as follows.

**The CFFP-tree construction algorithm:**

**INPUT:** A quantitative database consisting of  $n$  transactions and  $m$  items, a set of membership functions, and a predefined minimum support threshold  $s$ .

**OUTPUT:** A constructed CFFP tree.

**STEP 1:** Transform the quantitative value  $v_{ij}$  of each item  $I_j$  in the  $i$ -th transaction into a fuzzy set  $f_{ij}$  represented as  $(f_{ij1}/R_{j1} + f_{ij2}/R_{j2} + \dots + f_{ijn}/R_{jhj})$  using the given membership functions, where  $h_j$  is the number of fuzzy regions for  $I_j$ ,  $R_{jl}$  is the  $l$ -th fuzzy region of  $I_j$ ,  $1 \leq l \leq h_j$ , and  $f_{ijl}$  is  $v_{ij}$ 's fuzzy membership value in region  $R_{jl}$ . Note that  $f_{ijl}/R_{jl}$  represents the membership value of the region  $R_{jl}$  is  $f_{ijl}$ .

**STEP 2:** Calculate the scalar cardinality  $count_{jl}$  of each fuzzy region  $R_{jl}$  in all the transactions as:

$$count_{jl} = \sum_{i=1}^n f_{ijl}.$$

**STEP 3:** Find  $max-count_j = \text{Max}_{l=1}^{h_j}(count_{jl})$  for  $j = 1$  to  $m$ , where  $h_j$  is the number of fuzzy regions for item  $I_j$  and  $m$  is the number of items. Let  $max-R_j$  be the region with  $max-count_j$  for item  $I_j$ . It will then be used to represent the fuzzy characteristic of item  $I_j$  in the later mining process.

**STEP 4:** Check whether the value  $max-count_j$  of a fuzzy region  $max-R_j$ ,  $j = 1$  to  $m$ , is larger than or equal to the predefined minimum count  $n*s$ . If a fuzzy region  $max-R_j$  satisfies the above condition, put the fuzzy region with its count in  $L_j$ . That is:

$$L_j = \{max-R_j \mid max-count_j \geq n*s, 1 \leq j \leq m\}.$$

**STEP 5:** While executing the above steps, also find the occurrence number  $o(max-R_j)$  of each fuzzy region in the quantitative database.

**STEP 6:** Build the Header\_Table by keeping the fuzzy regions in  $L_j$  in the descending order of their occurrence numbers.

**STEP 7:** Remove the fuzzy regions of the items not existing in  $L_j$  from the transactions of the transformed database. Sort the remaining fuzzy regions in each transaction according to the order of the fuzzy regions in the Header\_Table.

**STEP 8:** Initially set the root node of the CFFP tree as *root*.

**STEP 9:** Insert the transactions in the transformed database into the CFFP tree tuple by tuple. The following two cases may exist.

**Substep 9-1:** If a fuzzy region  $max-R_j$  in the currently processed  $i$ -th transaction has appeared at the corresponding path of the CFFP tree, add the membership value  $f_{ijl}$  of the region  $max-R_j$  in the transaction to the node with  $max-R_j$ . Besides, calculate

the membership values of the super-itemsets of  $max-R_j$  in the path by the intersection operator and add the values to the corresponding elements of the array (called *minval\_Ary*) in the node.

**Substep 9-2:** Otherwise, add a new node with  $max-R_j$  to the end of the corresponding path and set the membership value  $f_{ijl}$  of the region  $max-R_j$  in the currently processed  $i$ -th transaction as the value in the node. Besides, calculate the membership values of the super-itemsets of  $max-R_j$  in the path by the intersection operator and set the values to the corresponding elements of the array (called *minval\_Ary*) in the node. At last, insert a link from the node of  $max-R_j$  in the last branch to the current node. If there is no such a branch, insert a link from the entry of  $max-R_j$  in the Header\_Table to the current node.

After STEP 9, the final CFFP tree is built. In STEP 9, a corresponding path is a path in the tree which corresponds to the fuzzy regions to be processed in a transaction according to the order of fuzzy regions appearing in the Header\_Table.

**C. An Example**

In this session, an example is given to illustrate how to construct a CFFP tree from a quantitative database. Assume the quantitative database shown in Table 1 is used as the example. It consists of 6 transactions and 5 items denoted A to E. The number behind an item represents the amount of the item purchased.

Table 1. The quantitative database in the example.

TID	Items
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

Assume the predefined minimum support threshold is set at 30% and the fuzzy membership functions shown in Figure 1 are used for all the items.

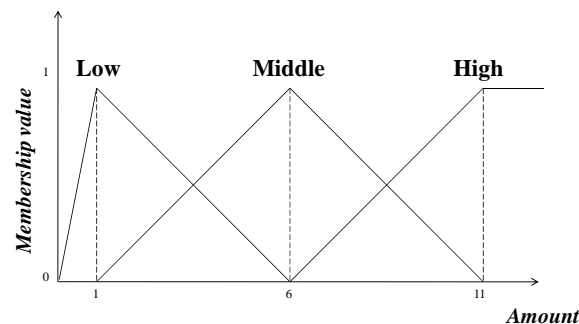


Figure 1. The membership functions used in the example.

In this example, amounts are represented by three linguistic terms as *Low*, *Middle* and *High*. Thus, three fuzzy membership values are produced for each item according to the predefined membership functions. Note that the proposed approach also works when the membership functions of the items are not the same and the function numbers are arbitrarily given. The proposed approach first constructs the CFFP tree from the given transactions as follows.

**STEP 1:** The quantitative values of all the items in the transactions are represented as fuzzy sets using the given membership functions in Figure 1. Take the item *A* in the first transaction as an example to illustrate the process. The amount “5” for *A* is converted into the fuzzy set  $(0.2/A.Low)+(0.8/A.Middle)$  using the given membership functions in Figure 1. This step is repeated for the other items, and the results are shown in Table 2.

Table 2. The fuzzy sets transformed from the data in Table 1.

TID	Items
1	$(0.2/A.Low)+(0.8/A.Middle)$ , $(0.2/C.Middle)+(0.8/C.High)$ , $(0.8/D.Low)+(0.2/D.Middle)$ , $(0.4/E.Middle)+(0.6/E.High)$
2	$(0.6/A.Middle)+(0.4/A.High)$ , $(0.8/B.Low)+(0.2/B.Middle)$ , $(0.6/C.Low)+(0.4/C.Middle)$
3	$(0.6/B.Low)+(0.4/B.Middle)$ , $(0.4/C.Middle)+(0.6/C.High)$
4	$(0.8/A.Middle)+(0.2/A.High)$ , $(0.4/C.Middle)+(0.6/C.High)$ , $(0.6/D.Low)+(0.4/D.Middle)$
5	$(0.2/A.Low)+(0.8/A.Middle)$ , $(0.8/B.Low)+(0.2/B.Middle)$ , $(0.2/C.Low)+(0.8/C.Middle)$
6	$(0.6/A.Low)+(0.4/A.Middle)$ , $(0.2/C.Middle)+(0.8/C.High)$ , $(0.8/D.Low)+(0.2/D.Middle)$ , $(0.8/E.Low)+(0.2/E.Middle)$

**STEP 2:** The scalar cardinality of each fuzzy region in all the transactions is calculated as the *count* value of the region. Take the fuzzy region *A.Low* as an example. Its scalar cardinality is  $(0.2 + 0.0 + 0.0 + 0.0 + 0.2 + 0.6)$ , which is 1.0. This step is repeated for the other regions, with the results shown in Table 3.

Table 3. The counts of fuzzy regions.

Item	Count	Item	Count
<i>A.Low</i>	1.0	<i>D.Low</i>	2.2
<i>A.Middle</i>	3.4	<i>D.Middle</i>	0.8
<i>A.High</i>	0.6	<i>D.High</i>	0.0
<i>B.Low</i>	2.2	<i>E.Low</i>	0.8
<i>B.Middle</i>	0.8	<i>E.Middle</i>	0.6
<i>B.High</i>	0.0	<i>E.High</i>	0.1
<i>C.Low</i>	0.8		
<i>C.Middle</i>	2.4		
<i>C.High</i>	2.8		

**STEP 3:** The fuzzy region with the maximum count among the three possible regions of each item is found. Take item *A* as an example to illustrate the process. Its counts for the three different fuzzy regions are *Low*:1.0,

*Middle*:3.4, and *High*:0.6, respectively. Since the count for *Middle* is the maximum among the three values, the region *Middle* is thus used to represent the item *A* in the later mining process. This step is repeated for the other items. Thus, region *Low* is chosen for items *B*, *D*, and *E*, and region *High* is chosen for item *C*.

**STEP 4:** The counts of the fuzzy regions selected in STEP 3 are checked against the predefined minimum count, which is  $6*30\% (= 1.8)$ . Since the count values of *A.Middle*, *B.Low*, *C.High* and *D.Low* are larger than 1.8, these four fuzzy regions are put in the set of  $L_1$ , which will be used to construct the CFFP tree later. After that,  $L_1 = \{A.Middle:3.4, B.Low:2.2, C.High:2.8, D.Low:2.2\}$ .

**STEP 5:** The occurrence number of each fuzzy region in  $L_1$  is also calculated. For example, the fuzzy region *A.Middle* appears in transactions 1, 2, 4, 5, and 6. Its occurrence number is thus 5. The results are shown in Table 4.

Table 4. The counts and occurrence numbers of the fuzzy regions.

Item	Count	Occurrence Number
<i>A.Middle</i>	3.4	5
<i>B.Low</i>	2.2	3
<i>C.High</i>	2.8	4
<i>D.Low</i>	2.2	3

**STEP 6:** The fuzzy regions in  $L_1$  are then sorted in the descending order of their occurrence numbers and are put in the Header\_Table.

**STEP 7:** The fuzzy regions not existing in  $L_1$  are removed from the transactions in Table 2. The remaining fuzzy regions in each transaction are then sorted according to the order in the Header\_Table. After this step, the updated transactions are shown in Table 5.

Table 5. The updated transactions after STEP 7.

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

**STEP 8:** The root of the CFFP tree is initially set as *root*.

**STEP 9:** The updated transactions in Table 5 are used to construct the CFFP tree tuple by tuple from the first transaction to the last one. Each node consists of not only the membership value of the fuzzy region within it but also the membership values of its super-itemsets in the path by the intersection operator. Take the first transaction as an example to illustrate the construction

process. In this example, the first transaction (0.8/A.Middle, 0.8/C.High, 0.8/D.Low) in Table 5 is first processed. A new node with the item A.Middle and the membership value 0.8 is then created and attached to the root. Since item A.Middle is the first item in the path, it is unnecessary to find the membership values of its super-itemsets from the transaction.

Next, the node of the item C.High with its membership value 0.8 is created and attached to the node A.Middle. In this example, the only super-itemset of item C.High in the path is (A.Middle, C.High). Its membership value is then calculated as  $(A.Middle \cap C.High)$ , which is  $0.8 \cap 0.8 (= 0.8)$ . The super-itemset with its membership value is then kept in the *minval\_Ary* of the node with C.High. After that, the node of the item D.Low with its membership value 0.8 is created and attached to the node C.High. Its super-itemsets include (A.Middle, D.Low), (C.High, D.Low) and (A.Middle, C.High, D.Low). Their membership values are then calculated as  $(A.Middle \cap D.Low)$ , which is  $(0.8 \cap 0.8) (= 0.8)$ ,  $(C.High \cap D.Low)$ , which is  $(0.8 \cap 0.8) (= 0.8)$ , and  $(A.Middle \cap C.High \cap D.Low)$ , which is  $(0.8 \cap 0.8 \cap 0.8) (= 0.8)$ . The membership values are then kept in the array of *minval\_Ary* attached to node D.Low. Besides, three links from the Header\_Table to the nodes of A.Middle, C.High and D.Low are generated because it is the first time for building the nodes with the items in the tree. The results after the first transaction is processed are shown in Figure 2.

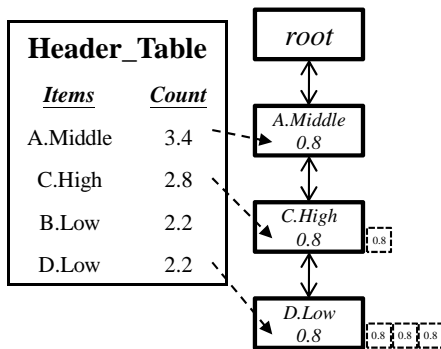


Figure 2. The CFFP tree after the first transaction is processed.

The same process is then executed for the other transactions. After all the six transactions are processed, the resulting Header\_Table and the constructed CFFP tree are shown in Figure 3.

After STEP 9, a complete CFFP tree has been constructed and can be used for helping derive all the fuzzy frequent itemsets. The proposed mining algorithm called CFFP-growth and based on the CFFP tree is then stated in the next section.

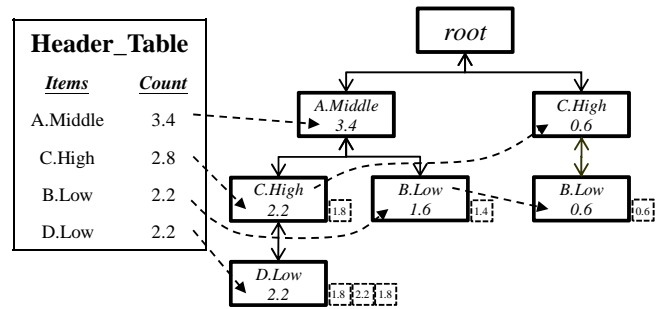


Figure 3. The finally constructed CFFP tree.

### 4. The Proposed CFFP-growth Mining Algorithm

After the CFFP tree is constructed, the desired frequent fuzzy itemsets can then be derived by the proposed CFFP-growth mining algorithm. It is described below.

#### A. The Mining Algorithm

The fuzzy frequent items in the Header\_Table are processed one by one and bottom-up. The nodes containing the currently processed item are first found from the CFFP tree. The fuzzy frequent itemsets can then be derived efficiently and effectively from the nodes. The mining algorithm is stated as follows.

#### The CFFP-growth mining algorithm:

**INPUT:** The constructed CFFP tree, its corresponding Header\_Table, and the predefined minimum support threshold *s*.

**OUTPUT:** The fuzzy frequent itemsets.

**STEP 1:** Process the fuzzy regions in the Header\_Table one by one and bottom-up by the following steps. Let the currently processed region be *max-R<sub>j</sub>*.

**STEP 2:** Find all the nodes with the fuzzy region *max-R<sub>j</sub>* in the CFFP tree through the links.

**STEP 3:** Extract the fuzzy itemsets and their membership values from the array of *minval\_Ary(K)* stored in each node *K* extracted in STEP 2.

**STEP 4:** Sum the membership values of the same fuzzy itemsets together.

**STEP 5:** If the summed membership value of a fuzzy itemset in STEP 4 is larger than or equal to the predefined minimum count (*n\*s*), add *max-R<sub>j</sub>* to the fuzzy itemset and output it as a fuzzy frequent itemset.

**STEP 6:** Repeat STEPs 2 to 5 for another fuzzy region until all the fuzzy regions in the Header\_Table are processed.

After STEP 6, all the fuzzy frequent itemsets can be derived from the constructed CFFP tree. Below, an example is given to illustrate the algorithm.

B. An Example

For the constructed CFFP tree in Figure 3, the proposed CFFP-growth mining algorithm finds the fuzzy frequent itemsets as follows.

**STEP 1:** The fuzzy regions in the Header\_Table are processed one by one and bottom-up. In this example, the fuzzy regions are processed in the order of *D.Low*, *B.Low*, *C.High* and *A.Middle*. The fuzzy region *D.Low* is first processed by the following steps.

**STEP 2:** The nodes with the currently processed fuzzy region *D.Low* in the CFFP tree are found. In this example, there is only one node in the CFFP tree containing the fuzzy region *D.Low*.

**STEP 3:** The fuzzy itemsets and their membership values are extracted from the array of *minval\_Ary* stored in each extracted node. For the node *D.Low*, the fuzzy itemsets and memberships values extracted are  $\{(A.Middle, D.Low):1.8\}$ ,  $\{(C.High, D.Low):2.2\}$  and  $\{(A.Middle, C.High, D.Low):1.8\}$ .

**STEP 4:** The membership values of the same itemsets are summed together. In this example, since only one node is extracted for *D.Low*, no summation needs to be done.

**STEP 5:** In this example, the number of transactions is 6 and the minimum support threshold is 0.3. The minimum count is thus calculated as  $6 \times 0.3$ , which is 1.8. The membership values of the itemsets generated in STEP 3 are then checked against the minimum count 1.8. For *D.Low*, all the generated itemsets in STEP 3 satisfy the condition and are output as the fuzzy frequent itemsets.

**STEP 6:** The above steps are repeated for another fuzzy region until all the fuzzy regions in the Header\_Table are processed. The final fuzzy frequent itemsets with their membership values are shown in Table 6.

Table 6. The updated transactions after STEP 7.

1-Itemsets	Count	2-itemsets& 3-itemsets	Count
<i>A.Middle</i>	3.4	<i>A.Middle, C.High</i>	1.8
<i>B.Low</i>	2.2	<i>A.Middle, D.Low</i>	1.8
<i>C.High</i>	2.8	<i>C.High, D.Low</i>	2.2
<i>D.Low</i>	2.2	<i>A.Middle, C.High, D.Low</i>	1.8

### 5. Experimental Results

The experiments were performed in Java on an AMD Athlon PC with a 3.0G Hz processor and 1G main memory, running the Microsoft Windows XP operating system. A real dataset called *mushroom* was used in the experiments [9]. Random quantitative values from the range [1, 13] were assigned to the items in the transactions in a uniform distribution. The quantitative

database was then transferred into fuzzy regions by the predefined membership functions. Two and three membership functions for each item were tested, respectively. Figure 4 shows the execution time for the proposed CFFP-growth mining algorithm, which included the tree-construction phase and the mining phase, in the two numbers of fuzzy regions. The minimum support threshold was set at from 10% to 30%, with 5% increment each time.

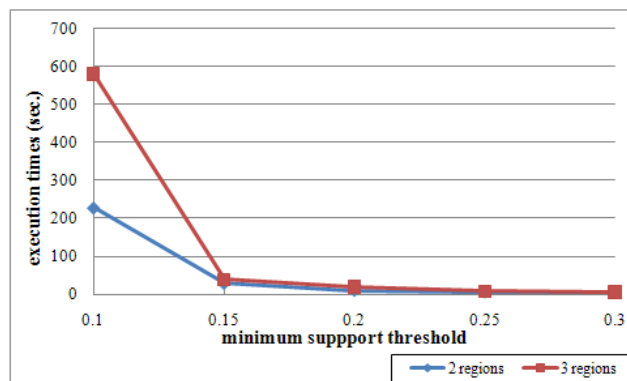


Figure 4. The comparison of the execution time in the two different numbers of fuzzy regions.

It was obvious to see from Figure 4 that more execution time was required for three regions than for two regions in the five different minimum support thresholds. This was because three regions would generally cause more fuzzy regions to process than two fuzzy regions. However, more fuzzy regions do not necessarily need more execution time. This is because the number of transferred fuzzy regions depends on the predefined membership functions. Besides, the quantitative values of items also affect the number of transferred fuzzy regions. Next, experiments were made to show the number of tree nodes in the two numbers of regions. The results are shown in Figure 5.

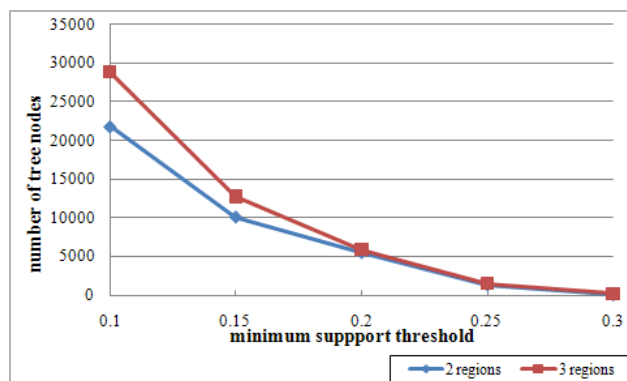


Figure 5. The comparison of the tree nodes in the two different numbers of fuzzy regions.

It could be also seen from Figure 5 that three regions

kept more tree nodes than two regions in the five different minimum support thresholds. The reason was the same as that mentioned above.

## 6. Conclusion and Future Work

In this paper, a novel tree structure called the compressed fuzzy frequent pattern tree (CFFP tree) has been designed. The tree can keep related mining information such that the database scans can be greatly reduced. A tree construction algorithm is also proposed to build the tree from a quantitative database. It first transforms quantitative values in transactions into linguistic terms and then finds the ones with the maximum cardinalities for mining. The construction process is similar to the FP-tree-like processing for building a tight tree structure except each node in the tree has to keep the membership value of the fuzzy region in the node as well as the membership values of its super-itemsets in the path. Besides, the CFFP-growth mining algorithm is also proposed to derive the fuzzy frequent itemsets from the constructed CFFP tree. Without level-wise generation of candidate itemsets, the fuzzy frequent itemsets can be derived efficiently and effectively from the CFFP tree. Experimental results compare the performance of the proposed algorithm in the execution time and the number of tree nodes in two different numbers of fuzzy regions. The overhead of the proposed approach is the additional array stored in each node. It is, however, tolerable because only fuzzy frequent 1-itemsets, instead of all the ones, will be used to build the tree.

When compared to the original FP tree algorithm, the proposed one can process quantitative transactions, instead of only binary transactions. The proposed approach will, however, spend more execution time than the original FP-tree approach because the former will need to transfer the quantitative values into fuzzy sets.

The database is assumed static in this paper. In real-world applications, data may be dynamically inserted into or deleted from a database. In the future, we will attempt to handle the maintenance problem of fuzzy data mining when the transactions are inserted, deleted or modified. Besides, how to define a minimum support threshold is an open problem in data mining. If the minimum support threshold is set low, too many rules are derived; on the contrary, if the threshold is set high, too little information is obtained. In the future, we will also try to design appropriate strategies to set the threshold according to user preferences and profiles.

## References

[1] R. Agrawal and R. Srikant, "Fast algorithms for

mining association rules in large databases," *The 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994.

- [2] R. Agrawal and R. Srikant, "Mining sequential patterns," *The International Conference on Data Engineering*, pp. 3-14, 1995.
- [3] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," *International Conference on Management of Data*, pp. 207-216, 1993.
- [4] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 914-925, 1993.
- [5] K. C. C. Chan and W. H. Au, "Mining fuzzy association rules," *The 6th International Conference on Information and Knowledge Management*, pp. 209-215, 1997.
- [6] M. Delgado, N. Marin, D. Sanchez, and M. A. Vila, "Fuzzy association rules: General model and applications," *IEEE Transactions on Fuzzy Systems*, vol. 11, pp. 214-225, 2003.
- [7] D. Dubois, E. Hüllermeier, and H. Prade, "A systematic approach to the assessment of fuzzy association rules," *Data Mining and Knowledge Discovery*, vol. 13, pp. 167-192, 2006.
- [8] C. I. Ezeife and Y. Su, "Mining incremental association rules with generalized fp-tree," *The 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pp. 147-160, 2002.
- [9] B. Goethals, Frequent itemset mining dataset repository. Available: <http://fimi.cs.helsinki.fi/data/>.
- [10] 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, pp. 53-87, 2004.
- [11] T. P. Hong and J. B. Chen, "Finding relevant attributes and membership functions," *Fuzzy Sets and Systems*, vol. 103, pp. 389-404, 1999.
- [12] T. P. Hong, C. S. Kuo, and S. L. Wang, "A fuzzy aprioritid mining algorithm with reduced computational time," *Applied Soft Computing*, vol. 5, pp. 1-10, 2004.
- [13] T. P. Hong, C. W. Lin and Y. L. Wu, "Incrementally fast updated frequent pattern trees," *Expert Systems with Applications*, vol. 34, issue 4, pp. 2424-2435, 2008.
- [14] K. Hu, Y. Lu, L. Zhou, and C. Shi, "Integrating classification and association rule mining: A concept lattice framework," *The 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, pp.

- 443-447, 1999.
- [15] C. M. Kuok, A. Fu, and M. H. Wong, "Mining fuzzy association rules in databases," *SIGMOD Record*, vol. 27, pp. 41-46, 1998.
- [16] B. Lent, A. Swami, and J. Widom, "Clustering association rules," *The 13th International Conference on Data Engineering*, pp. 220-231, 1997.
- [17] C. W. Lin, T. P. Hong, and W. H. Lu, "Mining fuzzy association rules based on fuzzy fp-trees," *The 16th National Conference on Fuzzy Theory and Its Applications*, pp. 11-16, 2008.
- [18] C. W. Lin, T. P. Hong and W. H. Lu, "The Pre-FUFP Algorithm for Incremental Mining," *Expert Systems with Applications*, vol. 36, pp. 9498-9505, 2009.
- [19] F. Liu, Z. Lu, and S. Lu, "Mining association rules using clustering," *Intelligent Data Analysis*, vol. 5, pp. 309-326, 2001.
- [20] J. Luo and S. M. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," *International Journal of Intelligent Systems*, vol. 15, pp. 687-703, 2000.
- [21] S. Papadimitriou and S. Mavroudi, "The fuzzy frequent pattern tree," *The 9th WSEAS International Conference on Computers*, pp. 1-7, 2005.
- [22] W. Shitong, K. F. L. Chung, and S. Hongbin, "Fuzzy taxonomy, quantitative database and mining generalized association rules," *Intelligent Data Analysis*, vol. 9, pp. 207-217, 2005.
- [23] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," *The 5th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 3-17, 1996.
- [24] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," *SIGMOD Record*, vol. 25, pp. 1-12, 1996.
- [25] Y. Sucahyo and R. Gopalan, "Building a more accurate classifier based on strong frequent patterns," *Lecture Notes in Computer Science*, vol. 3339, pp. 1036-1042, 2005.
- [26] Q. Yong, L. Yong Jie, and X. Qing Song, "An improved algorithm of mining from fp-tree," *International Conference on Machine Learning and Cybernetics*, pp. 1665-1670, 2004.



learning, artificial intelligence, and social computing.



**Tzung-Pei Hong, Ph.D.:** He received his B.S. degree in chemical engineering from National Taiwan University in 1985, and his Ph.D. degree in computer science and information engineering from National Chiao-Tung University in 1992. He was in charge of the whole computerization and library planning for National University of Kaohsiung in Preparation from 1997 to 2000 and served as the first director of the library and computer center in National University of Kaohsiung from 2000 to 2001, as the Dean of Academic Affairs from 2003 to 2006 and as the Vice President from 2007 to 2008. He is currently a professor at the Department of Computer Science and Information Engineering and at the Department of Electrical Engineering. He has published more than 300 research papers in international/national journals and conferences and has planned more than fifty information systems. He is also the board member of more than twenty journals and the program committee member of more than one hundred and fifty conferences. His current research interests include parallel processing, machine learning, data mining, soft computing, management information systems, and www applications.



**Wen-Hsiang Lu, Ph.D.:** He received his B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan. He is an Associate Professor in the Department of Computer Science and Information Engineering at National Cheng Kung University, Tainan, Taiwan. His current research focuses on web mining, information retrieval, natural language processing, and medical informatics.