International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

An Adjacency matrix-based Multiple Fuzzy Frequent Itemsets mining (AMFFI) technique

Mahendra N Patel[1], Dr. S.M. Shah[2], Suresh B. Patel[3*]

Submitted: 06/12/2021 Accepted : 29/01/2022

Abstract: Recently, discovering helpful information from a database consisting of transactions has been a critical research topic. Several frequent itemsets mining for association rule mining, algorithms that can only handle binary databases have proposed. Transactions using numerical values, on the other hand, are ubiquitous in real-world applications. Thus, with reference to the quantitative transactional database, several algorithms were developed and "fuzzy frequent itemsets" (FFI) were discovered. Most of them just consider the term having maximum cardinality. As a result, the number of fuzzy regions processed is equal to the number of original elements. Multiple fuzzy zones of an item, on the other hand, give a better result for making a correct decision. This study presents an AMFFI-miner (Adjacency matrix-based Multiple Fuzzy Frequent Itemsets) for discovering multiple FFIs out of a quantitative transactional database. An adjacency matrix and fuzzy-list structure were designed to find multiple FFIs by scanning database only once and generates less number of candidate itemsets. Join two nodes if its co-occurrence between two fuzzy linguistics terms satisfies minimum support threshold by finding the co-occurrence between two fuzzy linguistics terms directly from the adjacency matrix, thus reducing the number of nodes joining and speeding up discovering multiple FFI. Experiments carried out to compare the suggested method's performance to that of existing methodologies based on running time, memory utilization, and the number of nodes joining.

Keywords: *Fuzzy-sets, multiple fuzzy frequent* itemsets, *multiple regions, List structure, Adjacency Matrix*

1. Introduction

Multiple Techniques belonging to "Data mining" are used for discovering the valuable "knowledge from datasets" called as (KDD)[1]. Methods of KDD are mainly classified as association rule mining (ARs) [1-3], Classification [4][5], and Clustering [6]. Amongst all of the techniques employed for mining frequent itemsets (FIs), the ARs are the most commonly used one. Apriori algorithm is first presented by Agrawal et al. [2] to mine ARs in a "level-wise" approach. It first generates candidate itemsets and applies pruning on them for finding the FIs at each level. This method requires scanning the database multiple times and generating numerous candidate itemsets, which is a time-consuming computation.

Han et al.[7] proposed a data structure called FP-tree (Frequent – Pattern Tree) to detect FIs without candidate generation using FP-growth mining technique. The FIs can discover quickly using this technique.

Quantitative databases provide higher details for analyzing and taking decision in real-world scenarios than typical binary databases. Quantitative databases built on crisp sets, on the other hand, are difficult to manage. To manage quantitative databases using fuzzy set theory, pre-established membership functions are employed for translating the quantitative values of a transaction into a representation of language concepts [8]. Hong et. al [9] proposed a level-wise strategy for mining fuzzy data to produce "fuzzy frequent itemsets"(FFIs). The maximum cardinality value is use in this method to generate frequent itemsets at each level. The maximum cardinality mechanism minimizes the cost of discovery calculation fuzzy frequent itemsets, but some information may be lost. Hong et. al [10] presented an effective strategy to discover complete fuzzy frequent itemsets using the Gradual Data-Reduction Strategy. Lin et al. then provided a number of techniques MFFIs can be mined depending on their tree topology [11-13]. Despite the fact that tree-based algorithms beat Apriori-like algorithms technique, MFFI mining still necessitates computation. Next, Lin et al. [14] proposed various algorithms, for discovering FIIs from their designed fuzzy list structure. Lin et al. [15] proposed two pruning procedures to reduce size of search space. However, the levels of calculation costs needed for discovering the multiple FFIs are still required.

This study proposes an adjacency matrix and a "fuzzy-list" structure for mining multiple FFIs called AMFFI-miner (Adjacency matrix-based multiple fuzzy frequent itemsets). In this method, first, scan the database and generate an adjacency matrix and fuzzy list. Join two nodes if its co-occurrence between two fuzzy linguistics terms is greater or equal to minimum support threshold by finding directly from the adjacency matrix, thus reducing the number of nodes joining so AMFFI-miner algorithm generates a smaller number of candidate itemsets to minimize search space. Find L2 (2-frequent itemsets) directly from the adjacency matrix. As a result, the cost of computing mining MFFIs

[1] *Ph.D. Scholar, Gujarat Technological University, Ahmedabad, India*
ORCID ID:0000-0002-4342-1070
[2] *Computer Engineering, L.D.C.E., Ahmedabad, India*
ORCID ID:0000-0002-4937-4527
[3] *Information Technology Department, GEC Gandhinagar, India*
ORCID ID:0000-0003-2861-8972
** Corresponding Author Email: mnpatel32@gmail.com*

can be greatly decreased. Experimental results prove superiority of proposed approach as compared to other prevalent techniques.

2. Related Work

Delgado et. al [16] proposed method to find fuzzy ARs for both types of databases namely relational and quantitative. Hong et. al [17] presented a novel method based on Apriori Tid data structure to get frequent patterns for increasing itemsets from quantitative databases. Hong et. al [18] used an FP-tree structure called FUFP-tree for reducing execution time in case of insertion or arrival of new data. Lin et. al [19] used the same FP-tree-like structure called FFP-tree (fuzzy frequent pattern) to discover FFIs from quantitative databases. There are some limitations which are resolved by Lin et. al [20][21]. Here, the authors proposed a compressed fuzzy frequent pattern (CFFP-tree) structure and the upper bound fuzzy frequent pattern (UBFFP-tree) structure. The CFFP-tree [20] and UBFFP-tree [21] structures, like the FFP-tree [19], use a global sorting approach for reducing the amount of tree nodes. In UBFFP [21] method used upper bound value for mining FFIS then CFFP-tree [20] method. Li et. al [22] proposed the FC-Tree structure and FCFI-miner (Fuzzy closed frequent itemsets miner) for discovering FFIS. In this method, the author used a superset pruning strategy for speeding up mining. To find complete information of all linguistic terms in the fuzzy set, authors discover MFFIs (Multiple fuzzy frequent itemsets). Hong et. al [11] proposed MFFP-tree structure and MFFP-growth mining method to discovering MFFIs. Similarly authors designed CMFFP-tree [12] and UBMFFP-tree [13] methods to generate MFFIs based on CFFP-tree [20] and UBFFP-tree [21], respectively. Lin et. al [15] designed a Fuzzy-list structure and MFFI-miner method to discover MFFIs. In this method author used two different pruning strategies for reducing the search space, running time and running space. In [23], authors used complex fuzzy list (CFL)-structure same like fuzzy list structure [15] and used type-2 membership function for discovering MFFIs. Fuzzy-set theory based various algorithms for discovering the knowledge in different application domains are also developed [24-26].

3. Problem Fundamentals and Research Gap

The Itemset-I = $\{i_1, i_2, \ldots, i_m\}$ is a finite set of m unique items. The quantitative database D contains n number of transactions such that $D = \{T_1, T_2, T_3, \ldots, T_n\}$. In which each transaction says $Tq \in D$ and $Tq \in I$. As well as each transaction has a unique identifier, that says *TID*. Each transaction *Tq* consists item with its purchase quantity value, say w_{iq}. A k length itemsets $K = \{i_1, i_2 \ldots i_k\}$ is called *k-itemsets*.

Table 1 shows a sample quantitative dataset, say *D*, consisting of seven transactions in the following example. In this example, consider minimum support $\emptyset$=1. Membership function £ shown in figure 1.

Table 1. Sample dataset (Quantitative)

Transaction ID	Item: Quantity
TID_1	A: 5; B: 10; C: 2; D: 9
TID_2	B: 8; C: 2; E: 3
TID_3	A: 5; B: 3; C: 10; E: 11
TID_4	A: 1; C: 8; D: 3
TID_5	A: 5; B: 2; C: 6
TID_6	B: 3; C: 10; D: 2; E: 2
TID_7	C: 3; E: 9

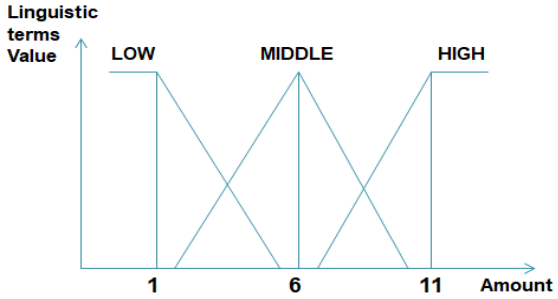


Fig. 1. Membership values

Table 2. Fuzzy dataset

Transaction ID	Linguistic terms of items
TID_1	AL-0.2 + AM-0.8, BM-0.2 + BH-0.8, CL-0.8 + CM-0.2, DM-0.4 + DH-0.6
TID_2	BM-0.6 + BH-0.4, CL-0.8 + CM-0.2, EL-0.6 + EM-0.4
TID_3	AL-0.2 + AM-0.8, BL-0.6 + BM-0.4, CM-0.2 + CH-0.8, EH-1.0
TID_4	AL-1.0, CM-0.6 + CH-0.4, DL-0.6 + DM-0.4
TID_5	AL-0.2 + AM-0.8, BL-0.8 + BM-0.2, CM-1.0
TID_6	BL-0.6 + BM-0.4, CM-0.2 + CH-0.8, DL-0.8 + DM-0.2, EL-0.8 + EM-0.2
TID_7	CL-0.6 + CM-0.4, EM-0.4 + EH-0.6

Fuzzy frequent itemsets mining method generally follow the following three steps.

Step 1: Find membership value of each item:

Quantities of an item i say w_{iq} represented in a linguistic variable say Li. In natural language representation, linguistic terms Li will be ($Li_1, Li_2, \ldots, Li_h$). Here h is defined by the membership function and h= no. of fuzzy regions of an item. For example, a 3-linguistic term membership function generates High-H, Middle-M and Low-L. Similarly, a 2-linguistic term membership function generates High-H and Low-L. Based on the membership function *£,* fuzzy terms are obtained from the quantitative value. The fuzzy linguistic term is fuzzy set $f_{iq,}$ where i is an item in a transaction T_q.

$f_{iq}=\{ L_{i1} - fw_{iq1} + L_{i2} - fw_{iq2} + \ldots . + L_{ih} - fw_{iqh}\}$.

Fw_{iqk} is the fuzzy value of k-th linguistic terms of L_{ik}, $1 \leq k \leq h$, and $fw_{iqk} \subseteq [0, 1]$.

For example, item A with quantity five is represented in linguistic terms (AL- 0.2, AM- 0.8, AH- 0.0) by 3-term membership function £ used in the above example. First, apply membership function *£* to transform quantitative dataset into fuzzy set say D' of different linguistic terms for all item shown in Table 2. In fuzzy set generated linguistic terms are denoted as fuzzy itemsets. Here in example AL-0.2 consider as linguistic variable A-low with 0.2 fuzzy values, same style use for other all.

Step 2: Find scalar cardinality of each fuzzy itemsets and 1-fuzzy frequent itemsets:

The scalar cardinality of fuzzy itemset L_{ik} denoted as sup (L_{ik}). In this step, find the support of each fuzzy itemsets. This defined as follow:

$$\text{Sup}(L_{ik}) = \sum_{q=0,\ Lik \subseteq Tq \wedge Tq \epsilon D'}^{q=n} (fwiqk).$$

Where fwiqk is the fuzzy value of fuzzy item Lik and D' is a fuzzy dataset.

For example, the fuzzy value of AL is 0.2, 0.2, 1.0, and 0.2 from transactions 1, 3, 4, and 5, respectively, according to the example taken. Scalar cardinality or support of AL = 1.6, it is a summation of all of its fuzzy values. Find scalar cardinality of all fuzzy

itemsets. If scalar cardinality >= Ø, where Ø is the minimum support threshold then, save the corresponding fuzzy item into fuzzy 1-frequent itemsets (FL1). Verify for each fuzzy item sup (L_{ik}), if satisfied with minimum support threshold then store in FL1.

FL1= FL1 ∪ (sup (Lik) >= Ø).

Step 3: Finding the support value:
Fuzzy 1-frequent itemsets (FL1) generate next-level fuzzy frequent itemsets say fuzzy k-itemsets where k ≥2. Using join operation joins fuzzy items from FL1 and generates candidate set say FC2 (fuzzy 2-candidate itemsets). Consider itemset X created by joining itemset A and B from FL1. Sup(X) = Support of itemset:X, calculated by summing up the minimum fuzzy value of fuzzy itemset A and B from truncation Tq, where X ⊆ Tq and Tq ∈ D'. This defined as follow:

$$\text{Sup(X)} = X \in Li \;/\; \sum_{q=0,\ \text{X}\subseteq\text{Tq}\wedge\ \text{Tq}\epsilon D}^{q=n} min(\text{fwaql},\ \text{fwbql}).$$

From FC2 itemsets that satisfy minimum support threshold Ø then store in fuzzy 2-frequent itemsets (FL2). The same way subsequently finds fuzzy k-frequent itemsets.

4. Proposed works

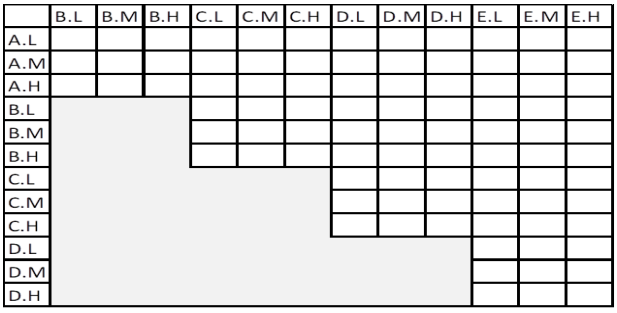
In this section, a proposed two-phase approach generates multiple fuzzy frequent itemsets. In phase 1, construct adjacency matrix and fuzzy-list from quantitative dataset D. Next, efficiently discover multiple fuzzy frequent itemsets from adjacency matrix and fuzzy-list using AMFFI-miner method. In proposed approach efficiently generates complete MFFIs by scanning the database only one time.

4.1. Phase 1(Adjacency matrix and fuzzy list construction):

Fuzzy list construction takes place in the first phase. Algorithm 1 shows the construction of the Adjacency matrix and fuzzy list for 2-fuzzy itemsets.
Items arranged in ascending order in transaction Tq, Tq ∈ D. Let us considers 3-term membership function £. First, construct adjacency matrix says AdjMat (M) of size (m *3) X (M *3). Here, m= total no. of items as per D. Here required matrix size is three-time more than the number of items (m). Matrix size is based on membership function. Use the 2-term membership function; then, the matrix size is two times more than the number of items.
AdjMat (M) = (m *t)X (m*t). Here m= total no. of items as per D and t = no. of fuzzy region as per membership function.
Scan transaction Tq and applying membership function £ to transform quantitative dataset into a fuzzy dataset of that transaction whose TID is q. Generate a pair of transformed fuzzy itemsets for different fuzzy variables from transaction Tq. Calculate the minimum fuzzy value of each pair, and then update the adjacency matrix's correspondence cell value by adding it and inserting the associated fuzzy list.
AdjMat (Li, Lj) =AdjMat (Li, Lj) +min (fwiq, fwjq)
Where Li and Lj are fuzzy items whose fuzzy value fwiq and fwjq respectively.
Construct the fuzzy list for Li and Lj if not exist. This fuzzy list inserted transaction id q (TID of Tq) and minimum fuzzy value of pair as min (fwiq, fwjq).

Algorithm 1: Construction of the Adjacency matrix and fuzzy list for 2-fuzzy itemsets
Input: Quantitative dataset D, No. of Items M
Output: Adjacency matrix AM and Fuzzy-list FL
*Step 1: Initialize Matrix AM for (M * no of the fuzzy region) X (M * no of the fuzzy region)*
Step 2: Initialize TID=1
Step 3: Read line L from D
Step 4: Repeat through step 7 while L is not the end of file D
Step 5: Apply membership function on each item's quantitative value in L and create fuzzy linguistic terms fl[] of all items.
Step 6: Store fuzzy value in adjacency matrix AM for all co-occurrences of fuzzy linguistic terms
Define FV= min (fuzzy value of fl[i], fuzzy value of fl[j])
AM (fl[i], fl[j]) + = FV
Create Fuzzy-List of " ' fl[i] '+' fl[j]' "if not exist
Create element with (TID, FV) & insert into Fuzzy-List of " ' fl[i] '+' fl[j]' "
Step 7: Increment TID by 1 and Read Next Line from D into L
Step 8: Finished.

Let us consider an example for the quantitative dataset as table 1 and membership function as fig 1 for 5 different items from A to E. The corresponding constructed the adjacency matrix shown in fig 2.
Scan first transaction (A: 5, B: 10, C: 2, D: 9) from dataset D and apply the membership to create fuzzy set (AL-0.2 + AM-0.8, BM-0.2 + BH-0.8, CL-0.8 + CM-0.2, DM-0.4 + DH-0.6) as shown in table 2. So created pair of transformed fuzzy itemsets are AL-BM, AL-BH, AL-CL, AL-CM, AL-DM, AL-DH, AM-BM, AM-BH, AM-CL, AM-CM, AM-DM, AM-DH, BM-CL, BM -CM, BM -DM, BM-DH, BH-CL, BH -CM, BH -DM, BH-DH, CL-DM, CL-DH, CM-DM, and CM-DH. Update all pair co-occurrences into adjacency matrix with minimum fuzzy value as shown in fig 3. Initially, there is no fuzzy-list created, therefor construct fuzzy-list for each pair and TID=1 and minimum fuzzy value is inserted. In figure 4 shown some pairs (AL-BM, BM -DM, CL-DH) Fuzzy-list. The same procedures followed for the next transaction. After reading all transaction adjacency matrix (M) looks like shown in figure 5 and figure 6 shown some pair (AL-BM, BM -DM, CL-DH) Fuzzy-list.



	B.L	B.M	B.H	C.L	C.M	C.H	D.L	D.M	D.H	E.L	E.M	E.H
A.L												
A.M												
A.H												
B.L												
B.M												
B.H												
C.L												
C.M												
C.H												
D.L												
D.M												
D.H												

Fig. 2. Adjacency Matrix

	B.L	B.M	B.H	C.L	C.M	C.H	D.L	D.M	D.H	E.L	E.M	E.H
A.L		0.2	0.2	0.2	0.2			0.2	0.2			
A.M		0.2	0.8	0.8	0.2			0.4	0.6			
A.H												
B.L												
B.M				0.2	0.2			0.2	0.2			
B.H				0.8	0.2			0.4	0.6			
C.L								0.4	0.6			
C.M								0.2	0.2			
C.H												
D.L												
D.M												
D.H												

Fig. 3. Adjacency Matrix after a 1st-row scan

A.L-B.M		B.M-D.M		C.L-D.H	
Tiid	Fuzzy Value	Tiid	Fuzzy Value	Tiid	Fuzzy Value
1	0.2	1	0.2	1	0.6

Fig. 4. Fuzzy-list after a 1st-row scan

	B.L	B.M	B.H	C.L	C.M	C.H	D.L	D.M	D.H	E.L	E.M	E.H
A.L	0.4	0.6	0.2	0.2	1.2	0.6	0.6	0.6	0.2	0	0	0.2
A.M	1.4	0.8	0.8	0.8	1.2	0.8	0	0.4	0.6	0	0	0.8
A.H	0	0	0	0	0	0	0	0	0	0	0	0
B.L				0	0	0	0	0	0	0	0	0
B.M				0	1.2	1.2	0.6	0.2	0	0.6	0.2	0.6
B.H				0.8	1	0.8	0.4	0.4	0.2	1	0.6	0.4
C.L							0	0.4	0.6	0.6	0.8	0.6
C.M							0.8	0.8	0.2	0.4	0.8	0.6
C.H							1.2	0.6	0	0.8	0.2	0.8
D.L										0.8	0.2	0
D.M										0.2	0.2	0
D.H										0	0	0

Fig. 5. Adjacency Matrix after all row scan

A.L-B.M		B.M-D.M		C.L-D.H	
Tiid	Fuzzy Value	Tiid	Fuzzy Value	Tiid	Fuzzy Value
1	0.2	1	0.2	1	0.6
3	0.2	6	0.2		
5	0.2				

Fig. 6. Fuzzy list after all row scan

4.2. Phase 2(AMFFI-miner to mine MFFIs):

In case of quantitative transaction data, the proposed AMFFI algorithm improved Apriori-like [16][17], FP-tree-like [12][13],[18-22], and the fuzzy-list-like [15][23] methods in finding FFIs. To reduce the huge no. of candidate generation, we utilize the upper triangular of adjacency matrix M.

In this phase, mine MFFIs from the adjacency matrix (M) row-by-row using fuzzy lists generated in phase1.

Scan the row from M and identify the cell with value >= Ø. Fuzzy-lists with RowNumber-ColumnNumber of an identified cell fetched from fuzzy-lists and declared as fuzzy 2-frequent itemsets say FL2 of this row. Subsequently, generate fuzzy k-frequent itemsets say FL_k (K>2) recursively by intersection operation by TIDs on FL_{k-1}. Use the binary search method to find combined fuzzy lists quickly. The fuzzy list (FL_k) for k-frequent itemsets (k>2) is constructed by combining existing fuzzy list of FL_{k-1}. Newly constructed fuzzy list consist elements those have common Tid in existing fuzzy list.

To reduce the search space and candidate set, not join fuzzy itemsets that cannot generate its superset that knows directly from adjacency matrix M. I.e., considering running example where Ø=1, read the second row from M. Here, the row number is A.M whose cells BL and CM satisfied the min support value. Thus, get FL2 from this row is AM-BL and AM-CM. By joining this possible superset is AM-BL-CM from FL2.However, the proposed method does not join this because it knows that generated superset does not satisfy the min support value. Here AM-BL and AM-CM are fuzzy frequent itemsets. Its superset AM-BL-CM is possible or not checked by BL row and CM column cell value. If it is greater or equal to min_support value, it may be possible; otherwise, not. In our example, this value is zero, which does not satisfy the Ø means its superset is impossible. Extensions of these are not fuzzy frequent itemsets, so discarded them before joining. This way minimizes join operation so vast reducing the candidate set, ultimately improving the running time performance. The AMFFI and AMFFI-miner methods show in Algorithm 2 and Algorithm 3, respectively.

Algorithm 2: AMFFI method

Input: AM: adjacency Matrix; FLs: the fuzzy-list; Minsupport
Output: MFFIs
Step 1: Initializations
for each row in AM do
Initialize fuzzy-list
L.FL ← null;
Step 2: for each cell in row
If cell value >= minsupport
Get fuzzy-list of (F[row.id][cell.id]) from FLs into temp
Add temp to L.FL
Call AMFFI-miner with L.FL and row_id

Algorithm 3: AMFFI-miner

Input: AM adjacency Matrix; FLs, fuzzy-list; Minsupport, row_id (fuzzy linguistic term)
Output: MFFIs
Step 1: for each fuzzy-list A in FLs do
Step 2: if SUM.A.if>= minsupport then
MFFIs ← A ∪ MFFIs.
Step 3:temp.FLs ← null;
Step 4: for each fuzzy-list B after A in FLs do
If A.ITEM = B.ITEM then
Continue;
If AM [A.ITEM][B.ITEM] >= minsupport then
temp.FL ← temp.FLs + Construct (A, B);
Step 5: AMFFI-Miner (temp.FLs);
Step 6: Return MFFIs.

5. Evaluation of Experimental Results

Here we present the performance analysis of proposed method as compared to MFFI-Miner [15]. In [15] authors, make a comparison of its own method MFFI-miner with other two methods: GDF [10] and UBMFFP tree [21]. The proposed AMFFI and MFFI-miner methods coded using Java. The results are analyzed on two real-life datasets, chess [27], mushroom [27], as well as one synthetic dataset, T10I4D100k [27]. In the datasets, the quantities of items give randomly in the 1 to 20 intervals. The experimental results analyzed in terms of runtime, join count and memory utilization.

5.1. Runtime Analysis:

The implemented3-term fuzzy linguistic AMFFI and MFFI-miner [15] were evaluated with different minimum support thresholds to compare execution running time. The output of execution running time evaluated on chess dataset shown in Fig 7, mushroom dataset shown in Fig 8 and T10I4D100k dataset shown in Fig 9.

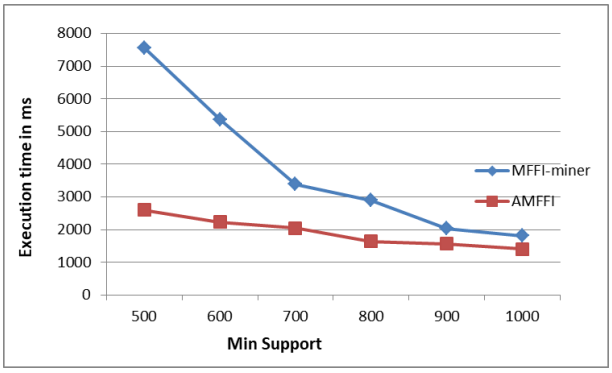


Fig. 7. Execution time comparisons: Chess dataset

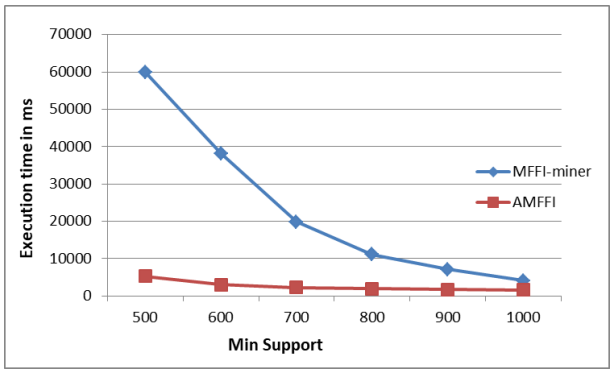


Fig. 8. Execution time comparisons: Mushroom dataset

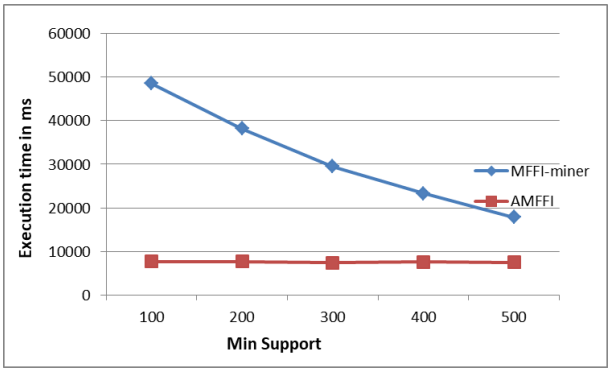


Fig. 9. Execution time comparisons: T10I4D100k dataset

MFFI-miner [15] outperforms the GDF [10] and the UBMFFP tree [21] in terms of running time. The proposed AMFFI method outperforms MFFI-miner method. Thus it proves to be the fastest amongst the three namely MFFI-miner [15], GDF [10] and UBMFFP tree [21]. The AMFFI method also exhibits its robustness when a lower minimum support threshold taken.

5.2. Join counts Analysis:

In this section, performance evaluated for the number of join count that occurs when generating MFFIs. The number of join counts for the chess dataset shown in Fig 10, mushroom dataset in Fig 11, and T10I4D100k dataset in Fig 12.

The result shows that the AMFFI method generates fewer join counts (candidate itemsets). It also observed that the most remarkable performance given the AMFFI method is the number of join counts. As compared to state-of-the-art methods, the proposed AMFFI method generates fewer candidate itemsets.

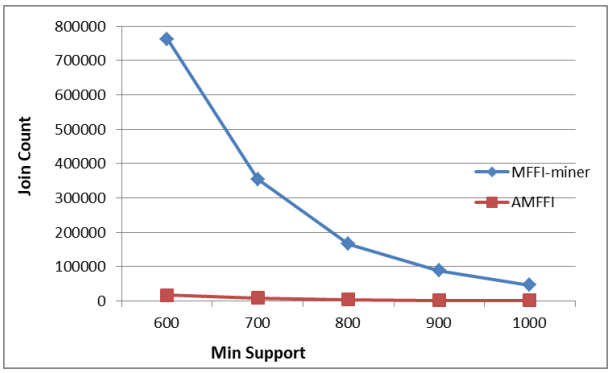


Fig. 10. Number of joint count comparisons on Chess dataset

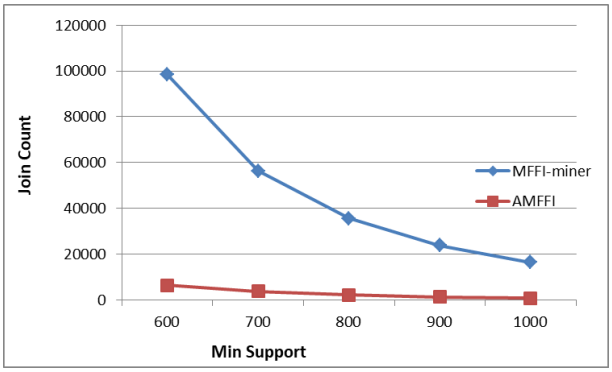


Fig. 11. Number of joint count comparisons on Mushroom dataset

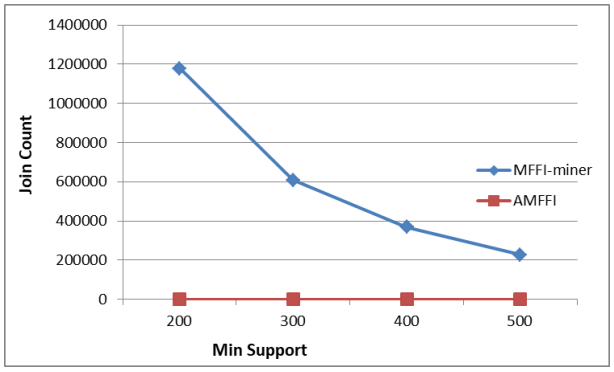


Fig. 12. Number of joint count comparisons on T10I4D100k dataset

5.3. Memory Usage Analysis:

Here, performance evaluated concerning the utilization of memory when evaluating experiments. The memory usage shows for the chess dataset in Fig 13, mushroom dataset in Fig 14 and T10I4D100k dataset in Fig 15.

The result shows that on the chess and mushroom dataset AMFFI method requires less memory than the existing MFFI-miner method. It also observed that on the synthetic T10I4D100k dataset AMFFI method requires more memory than the MFFI-miner method. From doing other experiments with different datasets, we can derive a special case where number of items in a dataset is higher than 1000, the proposed AMFFI will need a higher memory.

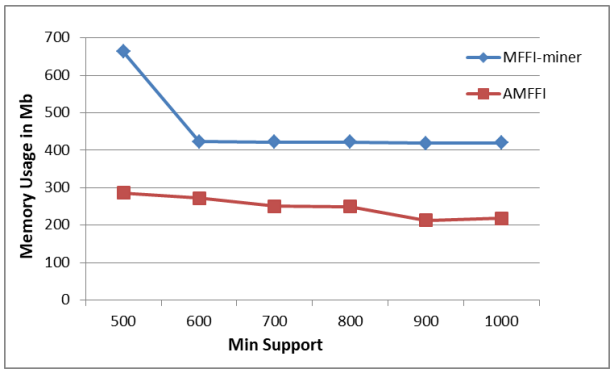


Fig. 13. Memory usage comparisons on Chess dataset

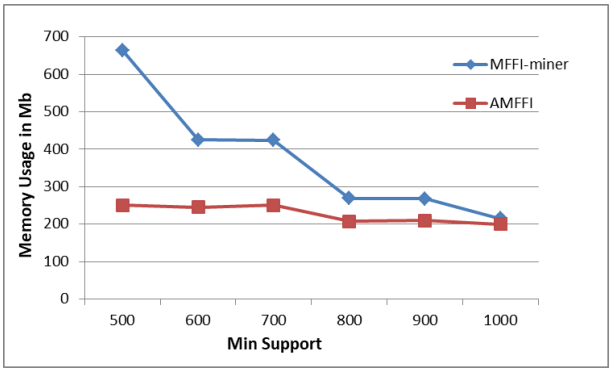


Fig. 14. Memory usage comparisons on Mushroom dataset

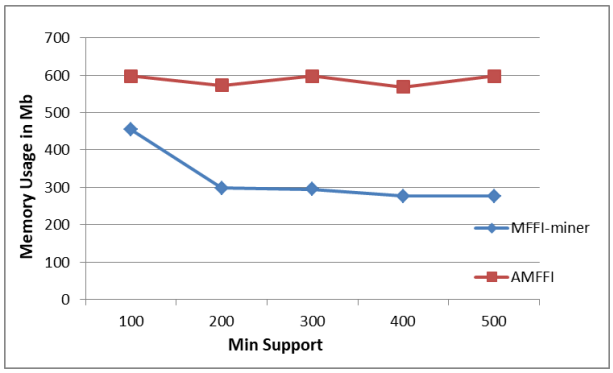


Fig. 15. Memory usage comparisons on T10I4D100k dataset

6. Conclusions:

With an objective to mine multiple fuzzy frequent itemsets efficiently, this paper presents an adjacency matrix as data structure and the AMFFI method. In comparison to the state-of-the-art methods, It generate fewer candidate itemsets by using an efficient search strategy. In addition, it generates fewer join counts (candidate itemsets), which in turn increasing the execution time performance. As far as the memory requirement is concerned, it is directly proportionate to the number of items in the dataset as discussed as a special case in section 5.3, in all other scenarios it utilizes comparatively less memory.

References

[1] R. Agrawal, S. Member, T. Imielinski, and A. Swami, "Database Mining: A Performance Perspective" *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 914–925, 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules" *The International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns" *The International Conference on Data Engineering*, pp. 3-14, 1995.

[4] M. Antonelli, P. Ducange, F. Marcelloni, and A. Segatori, "A novel associative classification model based on a fuzzy frequent pattern mining algorithm" *Expert Syst. Appl.*, vol. 42, no. 4, pp. 2086–2097, 2015.

[5] K. Hu, Y. Lu, L. Zhou, and C. Shi, "LNAI 1711 - 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.

[6] P. Berkhin, "A Survey of Clustering Data Mining Techniques" *Grouping Multidimensional Data*, pp. 25-71, 2006.

[7] J. Han and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" *Data Mining and Knowledge Discovery*, vol. 8. no. 1, pp. 53–87, 2004.

[8] J. Friedman and L. A. Z. Fuzzy, "Similarity relations and fuzzy orderings" *fuzzy sets Information and Control*, vol. 8, pp. 338–353, 1965.

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

[10] Hong, Tzung-Pei, "An Effective Gradual Data-Reduction Strategy for Fuzzy Itemset Mining," *Int. J. Fuzzy Syst.*, vol. 15, no. 2, pp. 170-181, 2013.

[11] T.-P. Hong, C.-W. Lin, and A. T.-C. Lin, "The MFFP-tree fuzzy mining algorithm to discover complete linguistic frequent itemsets" *Computational Intelligence*, vol. 30, no. 1, pp. 145–166, 2014.

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

[13] J. C. W. Lin, T. P. Hong, T. C. Lin, and S. T. Pan, "An UBMFFP tree for mining multiple fuzzy frequent itemsets," *Int. J. Uncertainty, Fuzziness Knowlege-Based Syst.*, vol. 23, no. 6, pp. 861–879, 2015.

[14] J. C. W. Lin, T. Li, P. Fournier-Viger, and T. P. Hong, "A fast Algorithm for mining fuzzy frequent itemsets," in *Journal of Intelligent and Fuzzy Systems*, vol. 29, no. 6, pp. 2373–2379, 2015.

[15] J. C. W. Lin, T. Li, P. Fournier-Viger, T. P. Hong, J. M. T. Wu, and J. Zhan, "Efficient Mining of Multiple Fuzzy Frequent Itemsets," *Int. J. Fuzzy Syst.*, vol. 19, no. 4, pp. 1032–1040, 2017.

[16] M. Delgado, N. Marín, D. Sánchez, and M. A. Vila, "Fuzzy association rules: General model and applications," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 2, pp. 214–225, 2003.

[17] T. P. Hong, C. S. Kuo, and S. L. Wang, "A fuzzy AprioriTid mining algorithm with reduced computational time," *Appl. Soft Comput. J.*, vol. 5, no. 1, pp. 1–10, 2004.

[18] T. P. Hong, C. W. Lin, and Y. L. Wu, "Incrementally fast updated frequent pattern trees," *Expert Syst. Appl.*, vol. 34, no. 4, pp. 2424–2435, 2008.

[19] C. W. Lin, T. P. Hong, and W. H. Lu, "Linguistic data mining with fuzzy FP-trees," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4560–4567, 2010.

[20] W. H. L. Lin, Chun Wei, Tzung Pei Hong, "An efficient tree-based fuzzy data mining approach," *Int. J. Fuzzy Syst.*, vol. 12, no. 2, pp. 150–157, 2010.

[21] C. W. Lin and T. P. Hong, "Mining fuzzy frequent itemsets based on UBFFP trees," *J. Intell. Fuzzy Syst.*, vol. 27, no. 1, pp. 535–548, 2014.

[22] H. Li, Y. Zhang, M. Hai, and H. Hu, "Finding Fuzzy Close Frequent Itemsets from Databases," *Procedia Comput. Sci.*, vol. 139, pp. 242–247, 2018.

[23] Lin, J. C. W., Wu, J. M. T., Djenouri, Y., Srivastava, G., & Hong, "Mining multiple fuzzy frequent patterns with compressed list structures" *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1-8, 2020

[24] S. Kar and M. M. J. Kabir, "Comparative analysis of mining fuzzy association rule using genetic algorithm" *The International Conference on Electrical, Computer and Communication Engineering*, pp. 1–5, 2019.

[25] D. K. Srivastava, B. Roychoudhury, and H. V. Samalia, "Fuzzy association rule mining for economic development indicators" *Int. J. Intell. Enterp.*, vol. 6, no. 1, pp. 3–18, 2019.

[26] L. Wang, Q. Ma, and J. Meng, "Incremental fuzzy association rule mining for classification and regression," *IEEE Access*, vol. 7, pp. 121095–121110, 2019.

[27] "Frequent Itemset Mining Dataset Repository" *http://fimi.ua.ac.be/data*