FIMA: A New Fuzzy Itemsets Miner Algorithm

Ass. Prof. Dr. Hussein K. Al-Khafaji
Al-Rafidain University college/ Software Eng. Dept.

ABSTRACT

Frequent itemsets mining is the second step of Association rules mining which is the main task of Knowledge Discovery (KDD) and data mining (DM). There are three types of itemsets; Crisp (CI), Generalized (GI), and Fuzzy itemsets (FI). FI is the most recent type. This paper presents a new algorithm to mine fuzzy itemsets from large taxonomic databases depending on fuzzy taxonomies that reflect partial belongings among data items, also it depends on Item-Transaction layout, and shortest path finding between an item and its super classes. The proposed algorithm, Fuzzy Itemsets Miner Algorithm (**FIMA**) deals with the three types of fuzzy itemsets; taxonomic nodes, linguistic terms, and hedges. FIMA scans the database, under mining, only once. It excludes the need for complicated data structures, prunes the pruning steps of available algorithm, and avoids the weakness of manipulating low levels values of minimum support threshold. The algorithm performs much better than the available algorithm such that it reduces the complexity of mining FIs from exponential, $O(a^n)$, to linear order of magnitude O(n).

1. Introduction

Data Mining is often defined as finding hidden knowledge in a database [Marg00]. Data Mining has many tasks; the main task is called *Association Rules* Mining [Elis01]. Association Rules are patterns of the form $X \xrightarrow{c} Y$, such that $X \cap Y = \varnothing$, where X and Y are itemsets and c is the confidentiality of the rule. The intuitive meaning of such a rule is that *transactions in the database which contain the items in X tend to also contain the items in Y* [1]. An example of such a rule might be that "*95% of customers who purchase pens and copybooks also buy some books*"; here 95% is called the *confidence* of the rule. The *support* of the rule is the percentage of transactions that contain both X and Y. Therefore, Association Rules, as specific form of knowledge, reflect relationships among items in database, and have been widely studied in the fields of knowledge discovery and data mining. However, in many situations of real applications discovery association rules involve *uncertainty* and *imprecision*, particularly fuzziness and generalization. The need of high-level managers and decision makers for generalization has led researchers to expand the concept of association rules to so-called **G**eneralized **A**ssociation **R**ules, GAR [2]. GARs represent the relationships between basic data items, as well as between data items at all levels of related taxonomies (or interchangeably, taxonomic structures). In most cases, taxonomies (is-a hierarchies) over the items are available. An example of taxonomic structures is shown in Figure (1) [2].

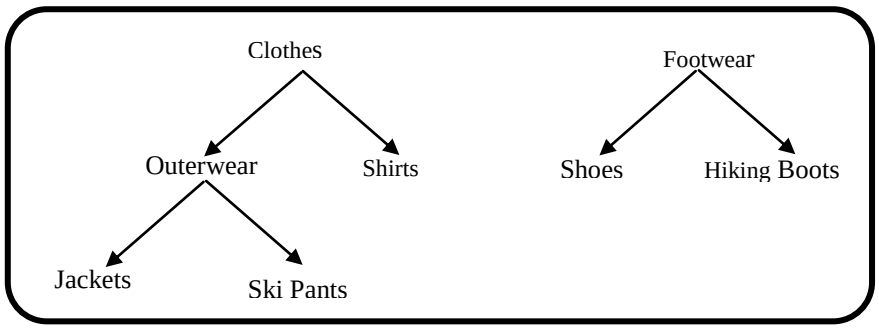

Figure (1) Example of Taxonomy represented as Directed Acyclic Graph (DAG)

An example of generalized AR is: "*50% of those who buy outerwear buy shoes*". Another rule is "*70% of those who buy clothes buy shoes*"[2]. Depending on the above taxonomy, minsup=30%, minconf=60%, and the database presented in Table(1). Table(2) and Table(3) depict the frequent itemsets and GARs respectively.

Table(1) Transactional DB

Transaction	Items Bought
100	Shirt
200	Jacket, Hiking Boots
300	Ski Pants Hiking Boots
400	Shoes
500	Shoes
600	Jacket

Table(2) Frequent Itemsets

Itemset	Support
{Jacket}	2
{Outerwear}	3
{Clothes}	4
{Shoes}	2
{Hiking Boots}	2
{Footwear}	4
{Outerwear, Hiking Boots}	2
{Clothes, Hiking Boots}	2
{Outerwear, Footwear}	2
{Clothes, Footwear}	2

Table(3) mined GARS

Rule	DConfidence	DSupport
Outerwear $\Rightarrow$ Footwear	66.6%	33%
Outerwear $\Rightarrow$ Hiking Boots	66.6%	33%
Hiking Boots $\Rightarrow$ Outerwear	100%	33%
Hiking Boots $\Rightarrow$ Clothes	100%	33%

The notions of the degree of support (Dsupport) and the degree of confidence (Dconfidence) play an important role in the association mining algorithms [1,2,3,4,5]:

$$Dsupport(X \Rightarrow Y) = ||X \cup Y|| / |T| \quad .....eq.(1)$$

$$Dconfidence\ (X \Rightarrow Y) = ||X \cup Y|| / ||X|| \quad .....eq.(2)$$

Where X and Y are itemsets with $X \cap Y = \varnothing$. T is the set of all the transactions contained in the database concerned. ||X|| is the number of the transactions in T that contain X, $\| X \cup Y \|$ is the number of the transactions in T that contain X and Y, and |T| is the number of the transactions contained in T.

In many real world applications, the related taxonomic structures may not be necessarily ***crisp***, rather, certain fuzzy taxonomic structures reflecting partial belonging of one item to another may pertain..For example, Tomato may be regarded as being both Fruit and Vegetable, but to different degrees [4]. Fuzzy set theory was proposed by Zadeh in 1965 to model the vagueness inherent to some concepts [6]. Fuzzy set theory allows an object to belong to a set with a membership degree between 0 and 1. An example of a fuzzy taxonomic structure is shown in Figure (2).

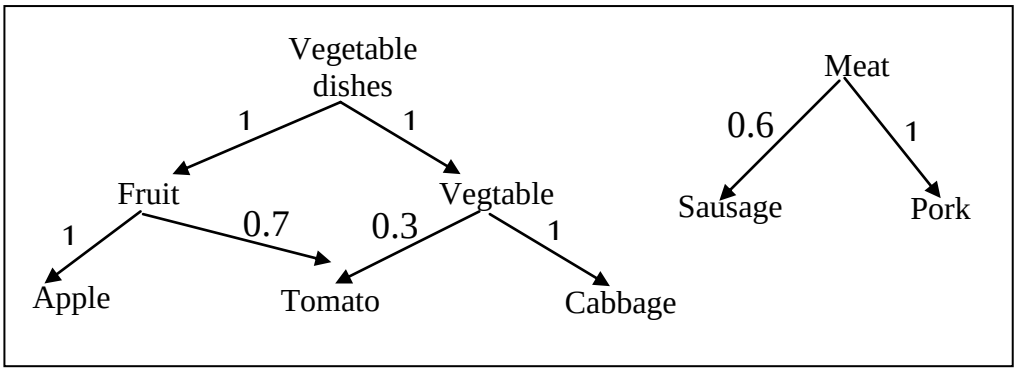

Figure (2) Example of Fuzzy Taxonomic Structures [4]

Here, a sub-item belongs to its super-item with a certain degree in [0, 1]. In fact, in such a fuzzy context, the computation of Dsupport and Dconfidence shown above can hardly be applied, but needs to be extended accordingly. Chen et al., [4], fuzzy taxonomies have been introduced for generalized association rule mining. Specifically, in analogue to the crisp case, given a transaction set T, there may exist a fuzzy taxonomic structure FG as shown in Figure (3). In Figure (3), every child-node y belongs to its parent-node x with degree μ_{xy} in [0,1]. Its computation is depending on the notions of fuzzy subclass, superclass and inheritance.

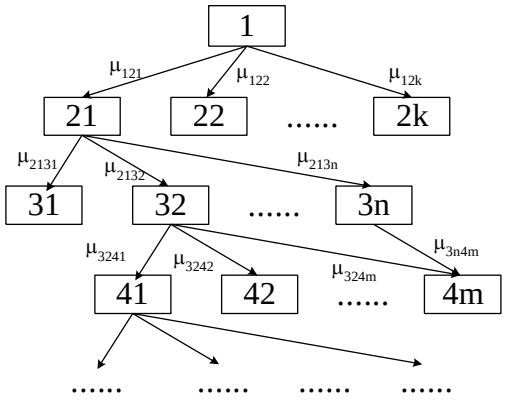

Figure (3) A fuzzy taxonomic structure[ ]

$$\mu_{xy} = \bigoplus_{\forall l: x \to y} \left(\bigotimes_{\forall e on l} \mu_{le} \right) \qquad \ldots\ldots\ldots \text{ eq.(3)}$$

Where $l: x \to y$ is one of the accesses (paths) of attributes x and y, e *on* l is one of the edges on access l, μ_{le} is the degree on the edge e on l. If there is no access between x and y, $\mu_{xy} = 0$. Notably, what specific forms of the operators to use for $\oplus$ and $\otimes$ depends on the context of the problems at hand. For the problem of FAR, ***max*** is used for $\oplus$ and ***min*** for $\otimes$.

In addition to the fuzziness involved in taxonomic structures, a more general view of fuzziness involvement in knowledge representation and discovery could be taken from the forms of the association rules. A fuzzy association rule is considered to be of the form X⇒Y where either X or Y is a collection of fuzzy sets. An example of such rules is "*VERY Expensive cloth* ⇒ *Tropical fruit*", where linguistic terms ("Expensive cloth" and "Tropical fruit") as well as linguistic hedge ("VERY") are involved. The linguistic terms and hedges are fuzzy in nature [1, 4, 6].

The complexity of mining FAR is resides in the discovery of fuzzy itemsets, FI. Therefore, there are many researchers have suggested algorithms to mine FI such as [2, 3, 4, 5, 7, 8] which can be classified as apriori-based algorithms. They are applied on quantitative database and pre-known

membership functions. Lopez et al.[7] suggested an algorithm to mine FAR which represent the relation between the features of yeast genome. This algorithm depended on the Top-Down Frequent Parent Growth (TD-FP Growth) algorithm which requires complex data structure called FP-tree. Also this algorithm supports linguistic term fuzziness only. Chen et al., [4] proposed an algorithm to mine frequent FIs hidden in taxonomic databases, this algorithm called Extended-Apriori algorithm. It inherits all the drawbacks of the standard algorithm of mining (crisp) frequent itemsets, apriori algorithm [11], in addition to the drawbacks emerged from the nature of FAR mining problem. These drawbacks involve Database multi-scan, inability to handle dense databases, inability to handle sparse databases with low minsup threshold, generating huge number of candidate itemsets, and the dependency of apriori algorithm on complex pruning steps.

Indeed, membership function, MF, has serious persuade on the mining step of FI. Therefore some researchers have used machine learning techniques to identify the MFs [9, 10]. These algorithms suffered from the complexity of knowledge representation problem and the time consumption of learning session.

The next section will concentrate on explanation of the complexity of mining FIs and counting their supports.

2. FIs' Supports Counting

Let I be the collection of all basic data items (or the leaf-nodes in the taxonomies), and I’ be the collection of all nodes of the taxonomies. Apparently, I is a subset of I’. In the crisp case, mining crisp association rules is to discover the relationships between the elements in I with X and Y being the subsets of I; while mining crisp generalized association rules is to discover the relationships between the elements in I’ with X and Y being the subsets of I’. In

the fuzzy case, X and Y are collections of fuzzy sets on the domains of interest in terms of rule semantics [1,2,4].

FARs on taxonomic nodes are fuzzy association rules that reflect the relationships of data between the nodes of fuzzy taxonomies. Since the higher-level nodes (or interchangeably, attribute-nodes) of the fuzzy taxonomic structures are generally fuzzy sets and usually labeled by meaningful linguistic terms, the discovered so-called generalized association rules with fuzzy taxonomies are fuzzy rules. In this case, X and Y are subsets of I' where the set (I'-I) is composed of higher-level nodes which are, in general, fuzzy sets defined on lower-level nodes in I'.

In a given fuzzy taxonomy, each attribute node may be viewed as a fuzzy set on its child nodes due to the partial belongings of its child nodes. Further, each attribute node could also be viewed as a fuzzy set on the leaf-nodes (i.e., set I) as expressed in eq.(3). Specifically, an attribute node x ∈ I' may be a fuzzy set as follows [2,4]:

$$x = \{ \mu_x(a)/a \mid a \in I,\ \mu_x(a) = \mu_{xa} = \underset{\forall l:x\to a}{\oplus} (\underset{\forall eonl}{\otimes} \mu_{le}) \} \ldots\ldots\ldots \text{eq.(4)}$$

If a is an attribute value in a certain transaction $t \in T$, T is the transaction set, and x is an attribute in certain itemset X, then the degree μ_{xa} with which a belongs to x can be obtained according to eq.(5). Thus, μ_{xa} may be viewed as the degree that the transaction $\{a\}$ supports x. Further, the degree that t supports X can be obtained as follows:

$$\mu_{tX} = Support_{tX} = \min_{x\in X}(\max_{a\in t}(\mu_{xa})) \quad \ldots\ldots\ldots \text{Eq.(5)}$$

Moreover, in terms of how many transactions in T support X, the Σ*count* operator [Gpe04] is used to sum up all the degrees that are associated with the transactions in T:

$$Dsupport(X) = \sum_{\forall t\in T} count\,(Support_{tX}) / |T| = \sum_{\forall t\in T} count\,(\mu_{tX}) / |T| \quad \ldots\text{eq.(6)}$$

Hence, for a generalized association rule X⇒Y, let X∪Y = Z, then Dsupport (X⇒Y) can be obtained as follows:

$$Dsupport(X \Rightarrow Y) = \sum_{\forall t \in T} count\,(\mu_{tZ}) / |T| \quad \ldots\ldots\ldots \text{Eq.(7)}$$

Next, in an analogous manner, *Dconfidence*($X \Rightarrow Y$) can be computed as follows:

$$Dconfidence(X \Rightarrow Y) = \sum_{\forall t \in T} count\,(\mu_{tZ}) / \sum_{\forall t \in T} count\,(\mu_{tX}) \quad \ldots\ldots\ldots\ldots \text{Eq.(8)}$$

Equations (6), (7), and (8) embody the complexities arise in counting Dsupport and Dconfidence of the fuzzy itemsets and rules which require database multi-scanning. The contexts of any one of the previous algorithms are not presented in this paper, but the following example illustrates the result of mined FI, their supports and some mined FARs with the confidence and support values.

Example(1): Given the taxonomies as shown in Figure (2), and the transactions as shown in Table (4), the frequent itemsets generated are shown in Table (5) with min-support being set to 1/3 [4].

Table (4) Transactions in a Supermarket database

Transaction #	Things Bought
1	Apple
2	Tomato, Sausage
3	Cabbage, Sausage
4	Tomato, Pork
5	Pork
6	Cabbage, Pork

Table (5) Σcount values for frequent itemsets

Frequent Itemsets	Σ*count* values
{Cabbage}	2
{Tomato}	2
{Pork}	3
{Sausage}	2
{Fruit}	2.4
{Vegetable}	2.6
{Vegetable dishes}	4.4
{Meat}	4.2
{Vegetable, Meat}	2.2
{Vegetable dishes, Meat}	2.9

Table(6) Mined FARs with minconf=50%

FAR	Confidence	Support
Vegetable⇒Meat	91%	36%
Meat⇒ Vegetable	52%	36%
Vegetable Dish⇒Meat	65%	48%
Meat⇒ Vegetable Dish	69%	48%

3. The Proposed Algorithm

This section presents a new algorithm to mine fuzzy itemsets; fuzzy itemset mining algorithm, FIMA. The input to FIMA is D', the extended database of the transaction database D. D' is generated in two steps:

1- Conversion of D to Item-Transactions layout if it is in Transaction-Items layout; and

2- Extending step, which will be explained in section(3.1).

The transaction of Item-Transactions layout has the form (*itemset, TIDLIST)*, where TIDLIST are Transaction IDentifiers of the transactions containing the itemset.

3.1 Database Extending

The fuzzy itemsets mining algorithm require extending the transactions of a database under mining. Extending the database means adding all the ancestors of an item to the transactions containing it.

This operation consumes long execution time due to:

1) Its need for removing duplicated parents.
2) Its need for re-sorting each transaction after extending and removing duplication
3) Increasing the size of each transaction that converts the database to dense type. It is known that apriori algorithm is originally inefficient in mining dense database. These drawbacks are excluded by the proposed extending-sub-algorithm presented below. To explain the complexity of extending the database, consider the database presented in Table (4) and the taxonomic structure presented in Figure (2).

Table (7) presents the extending process, and Table (8) shows the database after extending and sorting process.

Table (7) Extended Database	
Extended Transaction#	Extended Transaction's Items
1	Apple, Fruit, Vegetable_Dish
2	Tomato, Fruit, Vegetable_Dish, Vegetable, Sausage, Meat
3	Cabbage, Vegetable, Vegetable_Dish, Sausage, Meat
4	Tomato, Fruit, Vegetable_Dish, Vegetable, Pork, Meat
5	Pork, Meat
6	Cabbage, Vegetable, Vegetable_Dish, Pork, Meat

Table (8) Sorted Transaction Database	
Extended Transaction#	Extended Transaction's Items
1	Apple, *Fruit, Vegetable_Dish*
2	*Fruit, Meat*, Sausage, Tomato, *Vegetable, Vegetable_Dish*
3	Cabbage, *Meat*, Sausage, *Vegetable, Vegetable_Dish*
4	*Fruit, Meat*, Pork, Tomato, *Vegetable, Vegetable_Dish*
5	*Meat*, Pork
6	Cabbage, *Meat*, Pork, *Vegetable, Vegetable_Dish*

It is mentioned previously that the proposed algorithm, FIMA depends on Item-Transactions layout; therefore, the database must be extended vertically before presenting it to FIMA. This job is accomplished by the proposed algorithm presented in figure (4) which is called *D*atabase *E*xtending *S*ub-*A*lgorithm, *DESA*. DESA improves the extending process efficiently.

Step#3 of DESA copies frequent items of D to D'. The ancestors accommodate their ∑count values from frequent and infrequent items, therefore, step#4 represents a loop involves all the items of D to manipulate their ancestors.

Step#7 open a record for the ancestor of an item if there is no record for. Step#8 finds the shortest path from item i to its ancestor depending. Step#9 diminishes the complexity of ∑count operator counting to simple addition operation and this is one of the research contributions. Step#10 and step#11 construct pairs of (TID, $\mu_{TIDLIST}$ ancestor), i.e. the support value of a transaction

to the ancestor. Step#12 performs the union operation to add the TIDLIST of an item to the TIDLISTs of its ancestors.

```
1  DESA (input D, output D');
2   begin
3      copy frequent items of D to D';
4      forall i × tid(i) ∈ D do
5       begin
6          forall ancestor(i) do begin
7                 if ancestor(i) has no entry in D' open an entry for ancestor(i);
8                 μ i ancestor(i) =ShortestPath(i, ancestor(i));
9                 ∑count (ancestor(i))= ∑count (ancestor(i)) + μ i ancestor(i);
10                forall t∈ tid(i) do
11                     pair ( t, μ i ancestor(i));
12                tid(ancestor(i))= union (tid(ancestor(i)), tid(i));
13         end
14      end
15      Delete from D' infrequent ancestors;
16 end; // DESA.
```

Figure (4) DESA Steps

Step#15 removes the infrequent ancestors from the database. DESA attains many achievements comparable with the previous extending operation such as: *(1) DESA does not extend the transactions of the database but inserts new records in the database for unavailable ancestore, (2) no duplication occurs, (3) no resorting is required, (4) in addition to the fact that the density of the database is unchanged*.

Table (9) Item-Transactions Extended Database D'

Item	**(TID, Degree of Belongness)**	**∑ count**
{Fruit}	(1,1),(2,0.7),(4,0.7)	2.4
{Vegetable}	(2,0.3),(4,0.3),(3,1),(6,1)	2.6
{Vegetable_Dish}	(1,1),(2,0.7),(4,0.7),(3,1),(6,1)	4.4
{Meat}	(2,0.6),(3,0.6),(4,1),(5,1),(6,1)	4.2
{Cabbage}	(3,1),(6,1)	2
{Tomato}	(2,1), (4,1)	2
{Pork}	(4,1),(5,1),(6,1)	3
{Sausage}	(2,1),(3,1)	2

According to DESA, the Item-Transactions database of Table (4) and taxonomic structure of Figure (2) becomes as shown in Table (9). Where *item* field holds the items of taxonomic structure, i.e., TIDs field holds pairs of values. Each pair consists of a TIDLIST of the offspring and its degree of belongness to its ancestor. For example, the meaning of TIDs field in the record of fruit is: (1, 1) means there is an offspring exists in transaction#1, which has a degree of belongness to fruit equals 1. (2, 0.7) means there is an offspring exists in transaction #2, which has a degree of belongness to fruit equals 0.7, and so on.

Note that if the taxonomic structure is multi level or generalized structure the degrees of belongness are all set to 1, which can be omitted from the TIDs pairs, and in this case the ∑count=|TIDLIST| exactly as crisp itemset support counting according to the proposed algorithms. Hence DESA can be used to count the supports of Crisp, Generalized, and Fuzzy itemsets without any development or changes. This ability will empower FIMA to be used to mine crisp, generalized, and Fuzzy itemset. Also, this ability is one of the contribution of the research because there is algorithm can mine all the type of the itemsets.

3.2 Fuzzy Itemsets miner Algorithm (FIMA)

Figure (5) depicts the FIMA's steps. Where D' represents the extended database obtained from DESA, while FIDB represents the fuzzy itemsets database.

Step #3 of FIMA mines 1-itemsets. The implementation of this step can be involving the leaf nodes and/or taxonomical nodes, i.e. parent nodes. In other words, the mining of CAR can be done separately by using CAR mining algorithm or this duty can be accomplished by FIMA. Step#4 set the counter k of the itemsets lengths to 2 to start generating the fuzzy 2-itemsets. This counter will be updated after each generating process until no more itemsets to

be generated according to the loop of step#5. Step #7 invokes a function, called *fuzzy_itemsets_gen*. This function starts at step#12. It receives L_{k-1} itemsets to generate and return L_k itemsets.

```
1  FIMA(Input D', output FIDB); // FIDB Fuzzy Itemsets
DB
2  begin
3      L1 ={frequent 1-itemsets};
4      K=2;
5      While Lk-1#∅ do
6       Begin
7           Lk = Fuzzy-Itemset-Gen(Lk-1);
8           Lk = optimize(Lk)
9           K=K+1;
10       End
11 End

12 Fuzzy-Itemset-Gen(Lk-1);
13 begin
14 Ck =∅
15 For all itemsets X∈Lk-1 and Y∈Lk-1 do
16     if X1=Y1∧…∧ Xk-2 = Yk-2 ∧ Xk-1 < Yk-1 then
17      begin
18          C = union(X,Y);
19          CTID = intersect ( XTID, YTID);
20          Csupport = ∑count(C);
21          If Csupport ≥ Dsup add C to Lk
22          Else ignore C
23   End;
24 End;
```

Figure (5) FIMA steps

Step#8 accomplishes an important filtering operation to prevent the occurrences of redundant items or itemsets in next level; this filtering will diminish the execution time required to find the next level of fuzzy frequent itemsets. This process is done according to the heuristic presented in figure(6).

From implementation viewpoint, optimization#1 can be implemented separately to be executed once after generating L2 to prevent checking the "if

statement", but the optimization operation is presented as one unit for simplicity and well-understandability.

```
Optimization (Lk )
  Begin
     If k=2
        Delete l2 which consists of an item and its ancestor; // optimization 1
     Delete any ancestors in D' that are not presented in Lk // optimization 2
  End;
```

Figure(6) Optimization Operation

Fuzzy_itemsets_gen joins each pair of (k-1)-itemsets which have similar k-2 items at their first parts but have different items at the location number (k-1), i.e., if there are two itemsets X and Y of length (K-1) such that X1=Y1, X2=Y2, …, X_{k-2}= Y_{k-2} but X_{k-1} < Y_{k-1}, then step#18 will join X and Y to generate C to be as X1X2… X_{k-2} X_{k-1} $Y_{k-1.}$ Step#19 finds the common transactions of X and Y. Step#20 calculates the ∑*count* of the ancestor. *This operation depends on any shortest path finder algorithm*. Step#21 will add the generated itemset C to L_k if its support is equal or greater than the minimum support. Recall the structure of table(9), it is mentioned previously that TIDs field of D' consists of two sub fields TID and Degree of belongness. From implementation viewpoint, the Degree sub field is assigned its value during common items finding operation or by an independent module. The support field holds the support of an ancestor. D' can be used to store frequent fuzzy itemset. For example, if there exist two 1-itemsets {Meat} and {Vegetable_Dish}, the process of generation 2-itemset from them with all its required information depending on union and common transaction finding operation as shown in Figure (7).

The TIDLIST of the itemset {Meat, Vegetable_Dish}, i.e., {2,3,4,6}, is produced by common transactions finding operation of the itemset {Meat} and

{Vegetable_Dish}, i.e., the common transaction of {1,2,3,4,6} and {2,3,4,5,6}. The ∑*count* of {Meat, Vegetable_Dish} is computed by the summation of the degrees of belongness.

Min(support(2,Meat), support(2, Vegetable_Dish)) + min(support(3, Meaat), support(3, Vegetable_Dish)) + Min(support(4,Meat), support(4, Vegetable_Dish)) + Min(support(6,Meat), support(6, Vegetable_Dish)),

= Min(0.7,0.6) + Min(1,0.6) + Min(0.7,1) + Min(1,1)

= 0.6 + 0.6 + 0.7 + 1 = 2.9.

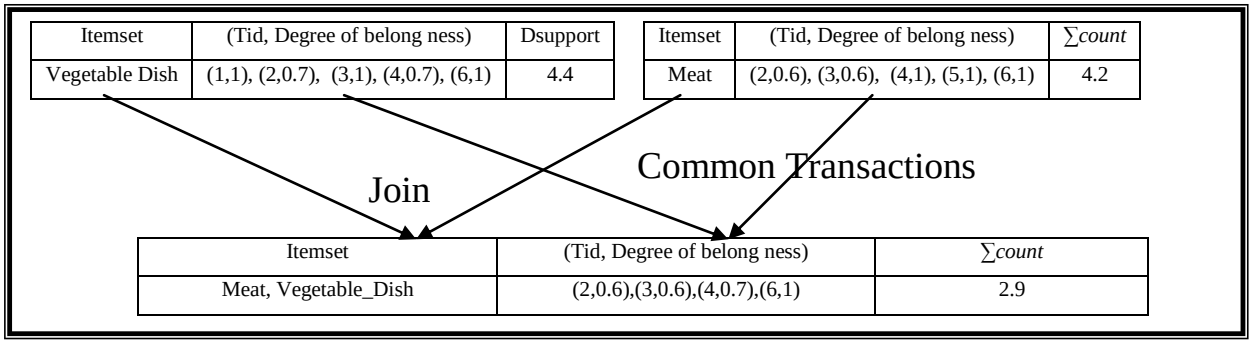

Figure (7) Fuzzy itemset generation

It is obvious that FIMA prevents extending the transactions and excludes the multi scan of the database under mining. It does not require any special data structure such as Hash Tree or FP-tree. Also, FIMA prevents generating redundant 2-itemsets depending on the optimization operations. The excluding of redundant 2-itemsets means excluding all the redundant itemsets in the next levels and excluding of the redundant association rules to be mined. To explain this point, recall Figure (2) and its database presented in table (9). According to this taxonomic structure, FIMA excludes eleven redundant 2-itemsets, which are:

{Apple, Fruit}, {Apple, Vegetable_Dish}, {Fruit, Tomato},
{Tomato, Vegetable}, {Tomato, Vegetable_Dish}, {Cabbage,
Vegetable}, {Cabbage, Vegetable_Dish}, {Fruit, Vegetable_Dish},

{Vegetable, Vegetable_Dish}, *{Meat, Sausage}*, and *{Meat, Pork}*.

In the case of Apriori-based algorithms, these redundant 2-itemsets will pass to the next level causing emerging many redundant 3-itemsets and so on.

4. Experiment Result

FIMA was tested by public databases [http://mips.gsf.de/genre/proj/yeast] (DB2), [http://www.yeastgenome.org] (DB2), and FAM95 from UCLA statistics data sets archive website [http://www.stat.ucla.edu/data/fpp] (DB3). The mined FIs of first and second databases were compared with the part of the results achieved by [9] and presented on [http://www.biomedcentral.com/content/1471_s2.pdf]. Indeed, the semantic of the databases is irrelative to the goal of the research, the research concentrated on the mining of these databases. Also, FIMA was used to mine three crisp databases presented on (www.ecn.purdue.edu/kddcup).

Unfortunately, there are no codes of other approaches to compare the complete result and the execution times; therefore the complexity of the algorithm is analyzed by using the order of magnitude which elucidated the outperforming of FIMA over these approaches. Anyway figure (8) depicts the mining time of the DB1, DB2, and DB3 according to different minimum support values.

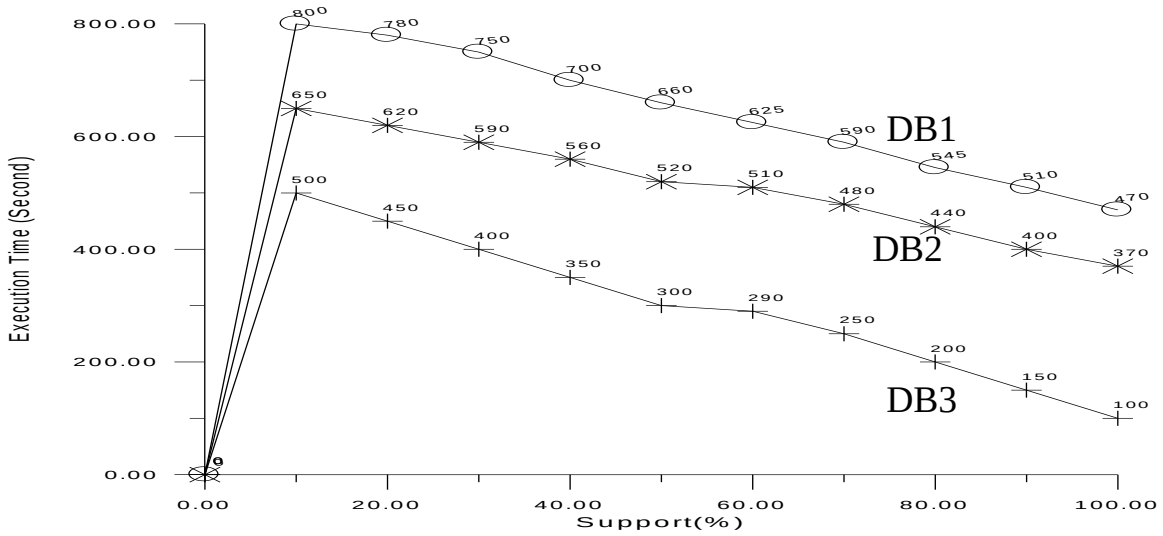

Figure (8) execution Time of DB1, Db2, and DB3

5. Discussion and Conclusion

This section involves combined points of discussions and conclusions depending on the analyzing and applying the proposed algorithm.

1. As explained previously, the GAR concept expands CAR concept, and since the FAR concept expands both GAR and CAR, so the proposed algorithm is applied on fuzzy and crisp databases to test its speed and scalability. The research presents a very important by-product result that is FIMA can be used for mining the three kinds of itemsets CI, GI, and FI. Therefore this algorithm can be regarded as a new algorithm to mine multi level, GI, CI, and FI. Logically, if FIMA is applied on taxonomic database with degrees of belongness are set to 1, and then FIMA will mine GI or multi level itemsets. Similarly, if it's applied on crisp database and in this case there is no belongness degrees between the items, i.e., equal zero, FIMA will mine CI successfully.
2. FIMA scans the database one time only by selecting the frequent items with their transactions and generates all levels of the frequent itemsets depending on simple operations done on the items and their transactions. Avoiding multi-scans over the database diminishes the mining time and converts the mining complexity from exponential to linear complexity; this complexity can easily be computed by using big-O notation. The big-O notation of FIMA is O(n) while the big-O notation of basic algorithm is $O(a^n)$.
3. FIMA requires no learning sessions as the algorithm presented in [9,10].
4. FIMA requires no special data structures such as hash tree [4] and FP-tree [7]. From implementation point of view FIMA requires two database fields to hold the itemsets and their transactions. It is better to select a type for these fields to hold very long strings which is available in some database management systems. Anyway, long data type can be used efficiently.

5. Many experiments are done on FIMA with different memory sizes. FIMA shows its appetite for memory. Indeed, FIMA needs to load two itemsets with their transactions in main memory to generate a higher level itemset with their transactions set and the DSupport. The maximum size of such itemsets depends on the features of the database under mining such as its density and the number of items.
6. The early excluding of the redundant itemsets diminishes the execution time due to the preventing of generating redundant itemsets in the next levels.
7. An interesting future work is adopting the principal of data sampling or parallel distribution to increase the scalability of FIMA.

6.References

[1] Triantaph Tlou and G. Felici, "**Data mining and Knowledge discovery approaches based on Rule Induction Techniques**", Massive Computing Series, Springer, Heidelgerg, Germany, PP 459-493, 2006.

[2] J. Han and M Kamber, "**Data Mining Concept and Techniques**", 2nd ed., Morgan Kaufman, 2006.

[3] Guoqing Chen, Peng Yan, and Etienne E. Kerre, **"Computationally Efficient Mining for Fuzzy Implication-Based Association Rules In Quantitative Databases"**, ISSN 0308-1079, 2004.

[4] Guoqing Chen, Qiang Wei, **"Fuzzy association rules and the extended mining algorithms"**, Information Science 147, 2002, pages 201-228 , www.elsevier.com/locate/ins , April 2002.

[5] M. De , C. Cornelis, and E. E. Kerre,**"Fuzzy Association Rules: A Two-Sided Approach"**, Ghentn Uni., Karijgslaan 281(S9), B-9000 Gent, Belgium 2003.

[6] Zadeh LA, "**A Computational Approach to Fuzzy Quantifiers in Natural Languages**", Compter Math. Applications, 1983.

[7] Francisco J Lopez, Armando Blanco1, Fernando Garcia1, Carlos Cano1 and Antonio Marin, "**Fuzzy association rules for biological data analysis: A case study on yeast",** Department of Computer Science and AI and Department of Genetics, University of Granada, 18071, BMC Bioinformatics, 2008.

[8] Keith C., Chan, Wai-ho Au, "**An Efficient Algorithm For Discovery Fuzzy Rules In Relational Databases**", Universia TV, http://biblotica.universia.net/autor/, 2009.

[9] José Gacto, Francisco Herrera, Jesús Alcalá-Fdez, Rafael Alcalá, "**Learning the membership function contexts for mining fuzzy association rules by using genetic algorithms**", Department of Computer Science and Artificial Intelligence, University of Granada, C/Daniel Saucedo Aranda, 18071Granada, Spain , May 2008.

[10] T. Hong, C. Chen, Y. Wu, Y. Lee, "**A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions**", Soft Comput. (2006).

[11] R. Agrawal, R. Srikant, "**Fast algorithms for mining association rules**", in: International Conference on Very Large Data Bases, Santiago de Chile, Chile, 1994, pp. 487–499.

خوارزمية جديدة لتعدين مجاميع العناصر المضببة

*أ.م.د. حسين كيطان الخفاجي
كلية الرافدين الجامعة

الخلاصة

تعدين مجاميع العناصر الكبيرة هي المرحلة الاولى والاكثر تعقيدا من مراحل تعدين قواعد الارتباط المخبئة في قواعد البيانات والتي بدورها تعد من أهم مهام تعدين البيانات. هنالك ثلاثة أنواع منها؛ مجاميع العناصر الكبيرة الواضحة، مجاميع العناصر الكبيرة المعممة، و مجاميع العناصر الكبيرة المضببة. الأخيرة هي الأحدث في أدبيات المجال والأكثر أهمية كون المشاكل الحقيقية مضببة في الغالب. هذا البحث يقدم خوارزمية جديدة لاستخراج مجاميع العناصر الكبيرة من قواعد البيانات التصنيفية المضببة. الخوارزمية تعتمد على التصنيف المضبب، الذي يشير الى انتماء العناصر الجزئي إلى تصانيفها الفوقية، وعلى الترتيب العمودي لقواعد البيانات، وعلى الهيكل الشبكي لتمثيل البيانات. الخوارزمية تتعامل مع الأنواع الثلاثة للمجاميع المضببة: العقد التصنيفية، العبارات اللغوية، والوشيعية. الخوارزمية تتفحص قاعدة البيانات مرة واحدة فقط، لا تحتاج إلى هياكل بيانات إضافية مثل أشجار الاختزال، لا تولد مجاميع مرشحة لذا فإنها لا تحتاج إلى خطوات التشذيب الموجودة في الخوارزمية المعتمدة على خوارزمية البديهة، وتتعامل مع خطوط عتبة منخفضة القيم. إن أداء الخوارزمية يتفوق على نظيرتها الحالية حيث حولت مشكلة التعدين من مشكلة أسية إلى مشكلة خطية التعقيد.