Aggregate Function Based Enhanced Apriori Algorithm for Mining Association Rules

Medhat H A Awadalla[1]**,and Sara G El-Far**[2]

[1] Communication, Electronics and Computers Department, **Helwan University ,
Cairo, State ZIP/Zone, Egypt**

[2] Communication, Electronics and Computers Department, **Helwan University ,
Cairo, State ZIP/Zone, Egypt**

Abstract

Association rule analysis is the task of discovering association rules that occur frequently in a given transaction data set. Its task is to find certain relationships among a set of data (itemset) in the database. It has two measurements: Support and confidence values. Confidence value is a measure of rule's strength, while support value corresponds to statistical significance. Traditional association rule mining techniques employ predefined support and confidence values. However, specifying minimum support value of the mined rules in advance often leads to either too many or too few rules, which negatively impacts the performance of the overall system. In this paper, it is proposed to replace the Apriori's user-defined minimum support threshold with a more meaningful aggregate function based on Central Limit Theorem (CLT). The paper also proposes a new function, MinAbsSup with bit mapping, which calculates a custom minimum support for each item set based on the probability of collision chance of its items. Furthermore, a modification for Apriori algorithm to accommodate this function is proposed. Experiments on large set of data bases have been conducted to validate the proposed framework. The achieved results show that there is a remarkable improvement in the overall performance of the system in terms of run time, the number of generated rules, and number of frequent items used.

Keywords: *Data Mining, Association Rule Mining, Apriori algorithm, minimum support, minimum confidence.*

1. Introduction

Association rule mining is interested in finding frequent rules that define relations between unrelated frequent items in databases, and it has two main measurements: support and confidence values. The frequent itemset is defined as the itemset that have support value greater than or equal to a minimum threshold support value, and frequent rules as the rules that have confidence value greater than or equal to minimum threshold confidence value. These threshold values are traditionally assumed to be available for mining frequent itemsets. Association Rule Mining is all about finding all rules whose support and confidence exceed the threshold, minimum support and minimum confidence values.

Association rule mining proceeds on two main steps. The first step is to find all itemsets with adequate supports and the second step is to generate association rules by combining these frequent (or) large itemsets [1-3].

In the traditional association rules mining [4-5], minimum support threshold and minimum confidence threshold values are assumed to be available for mining frequent itemsets, which is hard to be set without specific knowledge; users have difficulties in setting the support threshold to obtain their required results. Setting the support threshold too large, would produce only a small number of rules or even no rules to conclude. In that case, a smaller threshold value should be guessed (imposed) to do the mining again, which may or may not give a better result, as by setting the threshold too small, too many results would be produced for the users, too many results would require not only very long time for computation but also for screening these rules. That would explain the need to develop an algorithm to generate a minimum support, and minimum confidence values depending on the datasets in the databases.

To use association rule mining without support threshold [6-9], another constraint such as similarity or confidence pruning is usually introduced. However, the coincidental itemset problem had not been directly considered by any of these researches. There are some researches that are relevant to the coincidental itemset problem, and proposed an additional measure [10] in order to improve the support-confidence framework.

Initially, association rules mining was proposed for market basket analysis. Given a set of transactions *D*, association rule mining finds the complete set of association rules whose *support* is greater than a user-defined minimum support threshold (*min-sup*) and *confidence* greater than a user-defined minimum confidence threshold (*min-conf*). The following is a formal statement of association rule mining for transaction databases. Let $I = \{i_1, i_2. \ . \ . \ i_m\}$ be the universe of items. A set *X,I* of items is called an itemset. A transaction t = (*TID, X*) is a tuple where *TID* is a unique transaction ID and *X* is an itemset. A transaction database *D* is a set of transactions. The *count* of an itemset *X* in *D*, denoted by count(*X*), is the number of transactions in *D* containing *X*. The *support* of an itemset *X* in *D*, denoted by supp(*X*), is the proportion of transactions in *D* that contain *X*. The rule $X \rightarrow Y$ holds in the transaction set *D* with *confidence c* where c = conf($X \rightarrow Y$) and conf($X \rightarrow Y$) = supp ($X \upsilon Y$) / supp(X). The dominant theme in traditional association mining is the discovery of positive association rules in frequently occurring itemsets.

The continuation of this paper is as follows: section 2 presents the most related work to the theme of this paper. Section 3 introduces the Apriori algorithm. Section 4 presents Apriori inverse algorithm. Section 5 presents the enhanced Apriori algorithm. Sections 6 and 7 demonstrate the conducted experiments and discussions. Section 8 concludes the paper.

2. Background

A lot of association rule algorithms have been developed in the last decades [11-13], which can be classified into two categories: (1) breadth-first search (BFS) or *candidate-generation-and-test* approach such as Apriori [14], (2) depth-first search (DFS) *or pattern-growth* approach [15- 18]. With BFS the support values of all (k - 1) itemsets are determined before counting the support values of the k-itemsets. In contrast, DFS recursively descends following the tree structure defined above.

Each of the algorithms is characterized by its strategy to a) traverse the search space and b) determine the support values of the itemsets as shown in figure 1. In addition an algorithm may employ specific optimizations for further speeding up. The most popular algorithm of this type is Apriori [16, 19] where the downward closure property of itemset support was introduced. Apriori makes an additional use of this property by pruning those candidates that have an infrequent subset before counting their supports. This optimization issue becomes possible because BFS ensures that the support values of all subsets of a candidate are known in advance. Apriori counts all candidates of a cardinality k together in one scan over the database. The critical part is to look for the candidates in each of the transactions. For this purpose, the work in [16] introduces a so called hash-tree structure. The items in each transaction are used to descend in the hash-tree. Whenever they reach one of its leaves, they find a set of candidates having a common prefix that is contained in the transaction. Then these candidates are searched in the transaction that has been encoded as a bitmap [16]. In the case of success, the counter of the candidate in the tree is incremented.

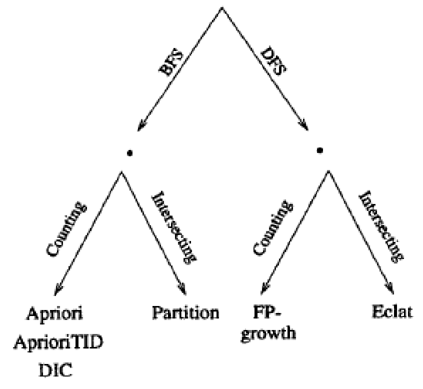

Figure 1: Systematization of the Algorithms

Apriori Tid [16] is an extension of the basic Apriori approach. Instead of relying on the raw database, Apriori Tid internally represents each transaction by the current candidates it contains. The Apriori Tid algorithm has the additional property that the database is not used at all for counting the support of candidate itemsets after the first pass. Rather, an encoding of the candidate itemsets used in the previous pass is employed for this purpose. In later passes, the size of this encoding itemsets can become much smaller than the database, thus saving much reading effort.

SETM algorithm [17] was motivated by the desire to use SQL to compute large itemsets, the candidate itemsets are generated on-the-fly during the pass as data is being read. Specially, after reading a transaction, it is determined which of the itemsets found large in the previous pass are present in the transaction. New candidate itemsets are generated by extending these large itemsets with other items in the transaction. However, the disadvantage is that this results in unnecessarily generating and counting too many candidate itemsets that turn out to be small.

DIC is a further variation of the Apriori-Algorithm [18, 19]. DIC softens the strict separation between counting and generating candidates. Whenever a candidate reaches min-supp; that is even when this candidate has not yet "seen" all transactions, DIC starts generating additional candidates based on it. For that purpose a prefix-tree is employed. In contrast to the hash-tree, each node - leaf node or inner node - of the prefix-tree is assigned to exactly one candidate respectively frequent itemset. In contrast to the usage of a hash-tree that means whenever we reach a node we can be sure that the itemset associated with this node is contained in the transaction. Furthermore interlocking support determination and candidate generation decreases the number of database scans.

The Partition-Algorithm [20, 21] uses set intersections to determine support values. As mentioned above Apriori determines the support values of all (k - 1) candidates before counting the k-candidates. The problem is that partition of course wants to use the Tid-lists of the frequent (k - 1) itemsets to generate the Tid-lists of the k-candidates. Obviously the size of those intermediate results easily grows beyond the physical memory limitations of common machines. To overcome this Partition, it splits the database into several chunks that are treated independently. The size of each chunk is chosen in such a way that all intermediate Tid-lists fit into main memory. After determining the frequent itemsets for each database chunk, an extra scan is necessary to ensure that the locally frequent itemsets are also globally frequent.

Counting occurrences assumes candidate sets of a reasonable size. For each of those candidate sets, a database scan is performed. Apriori that relies on BFS scans the database once for every candidate size k. When using DFS the candidate sets consist only of the itemsets of one of the nodes of the tree. Obviously, scanning the database for every node would results in tremendous overhead. The simple combination of DFS with counting occurrences is therefore of no practical relevance.

In [22] a fundamentally new approach called FP- growth was introduced. In a preprocessing step, FP- growth derives a highly condensed representation of the transaction data, so called FP- tree. The generation of the FP- tree is done by counting occurrences and DFS. In contrast to former DFS - approaches, FP-growth does not follow the nodes of the tree, but directly descends to *some part* of the itemsets in the search space. In a second step FP- growth uses the FP-tree to derive the support values of all frequent itemsets.

In [23] the algorithm ECLAT is introduced, that combines DFS with Tid-list intersections. When using DFS it suffices to keep the Tid-lists on the path from the root down to the class currently investigated in memory. That is, splitting the database as done by Partition is no longer needed. ECLAT employs an optimization called "fast intersections". Whenever two Tid-lists are intersected then the only interest is in the resulting Tid-list if its cardinality reaches min-supp. In other words, each intersection should be broken off as soon as it is sure that it will not achieve this threshold. ECLAT originally generates only frequent itemsets of size > 3. ECLAT had been modified to mine also the frequent 1 and 2 itemsets by calling it on the class that contains the 1 itemsets together with their Tid-lists as mentioned in [23].

In addition, in [23] algorithms that mine only the maximal frequent itemsets are introduced, e.g. Max-ECLAT. An itemset X is maximal frequent if for every frequent itemset $Y\ X \subseteq Y \rightarrow Y = X$ holds. These algorithms were not considered because although it is straight forward to derive the set of all frequent itemsets from the maximal frequent itemsets, this does not hold for the corresponding support values. Without those, it is not able to derive rule confidences and therefore not generating association rules. In the following sections, Apriori and Apriori Inverse algorithms will be focused for qualitative comparative study.

3. Apriori Algorithm

The most influential algorithm Apriori [7, 15, 16, 19], generates the *k*-candidate by combining two frequent (*k*-1) itemsets. The Apriori algorithm employs a bottom-up, *breadth-first* searching that generates all frequent itemsets, which is feasible with sparse datasets such as market-basket data, where the frequent patterns are very short.

However, the performance of these algorithms degrades incredibly in some application domains such as genome data where there are many, long frequent patterns, because they perform as many passes over the database as the length of the longest frequent pattern. This incurs high I/O overload for iteratively scanning large database.
The Apriori algorithm needs scanning the whole data set and examine the itemsets multiple of times, which is very time consuming process.

The algorithm Apriori
```
L1= {large1- itemset};
For (k=2; $L_{k-1} \neq \varphi$; k++)
{
$C_k$=Apriori ($L_{k-1}$);
For all transactions t $\epsilon$ D
{
Ct=subset ($C_k$,t);
For all candidates c ϵ Ct
        c.count++;
$L_k$ = {c ϵ $C_k$ | c.count >= minsup}
}
Return ( $\cup_k L_k$ )
        }
```

L_k: Set of large k-itemsets (those with minimum support). Each member of this set has two fields: (1) itemset, and (2) support count. C_k: Set of candidate k-itemsets (potential large itemsets). Each member of this set has two fields: (1) itemset and (2) support count.

The Apriori-gen function takes as an argument L_{k-1}, the set of all large *(k-1)* itemsets. It returns a superset of the set of all large k-itemsets. First, in the join step, L_{k-1} joins with L_{k-1} to obtain a superset of the final set of candidates C_k. The union p U q of itemset p, q U L_{k-1} is inserted in C_k if they share *k-2* first items.
For a transaction set {1: A, C, D; 2: B, C, E; 3: A, B, C, E; 4: B, E}, itemsets and its corresponding support count will be as follows:
{A: 2; B: 3; C: 3; D: 1; E: 3}. Assume Minimum support = 50% which equivalent to support count =2. By pruning the infrequent then frequent itemsets and their corresponding support count are {A: 2; B: 3; C: 3; E: 3}.

4. Apriori Inverse

The Apriori-Inverse algorithm [24] is based on a level-wise search. On the first pass through the database, an inverted index is built using the unique items as keys and the transaction IDs as data. At this point, the support of each unique item (1- itemsets) in the database is available as the length of each data chain.

To generate *k*-itemsets under max-sup, the (k – 1) itemsets are extended in precisely the same manner as Apriori to generate candidate k- itemsets. That is, a (k – 1) itemset i_1 is turned into a *k* itemset by finding another (k – 1) itemset i_2 that has a matching prefix of size (k – 2), and attaching the last item of i_1 to i_2. For example, the 3 - itemsets {1, 3, 4} and {1, 3, 6} can be extended to form the 4 - itemset {1, 3, 4, 6}, but {1, 3, 4} and {1, 2, 5} will not produce a 4 - itemset due to their prefixes failing to match at the second item.

These candidates then are checked against the inverted index to ensure that they at least meet a minimum absolute support requirement and are pruned if they do not, (the length of the *intersection* of a data chain in the inverted index provides support for a *k*-itemset with *k* larger than 1).
The process continues until no candidate itemsets can be generated, and then association rules are formed in the usual way.
It should be clear that Apriori-Inverse finds all perfectly sporadic rules, since we have simply inverted the downward-closure principle of the Apriori algorithm; rather than all subsets of rules being over min-sup, all subsets are under max-sup. Since making a candidate itemset longer cannot increase its support, all extensions are viable *except* those that fall under the minimum absolute support requirement. Those exceptions are pruned out and are not used to extend itemsets in the next round. For example, let *D* be {{1, 2, 3, 4}, {1, 3, 5}, {1, 3, 5, 7}, {1, 6, 8}, {2, 3, 4, 6}, {3, 6, 7, 8},{3, 6, 8}, {6, 9}}. The *Idx* from *D* where {*item*:[*tid-list*]} is {{1:[1, 2, 3, 4]}, {2: [1, 5]}, {3:[1, 2, 3, 5, 6, 7]}, {4: [1, 5]}, {5: [2, 3]}, {6: [4, 6, 7, 8]}, {7:[3, 6]}, {8:[4, 6, 7]}, {9:[8]}}. Given a maximum support of 25% and supposing that the minimum absolute support value is 2, *S1* will be {2, 4, 5, 7}. Items below the minimum absolute support value would not be considered for extension. Thus, item 9, which had the support of 1, was pruned out. The itemsets then are extended to {{2, 4}, {2, 5}, {2, 7}, {4, 5}, {4, 7}, {5, 7}}, but *S2* only contains itemset {2, 4}, because the other itemsets have support below the minimum absolute support value and so are pruned out.

Because we are dealing with candidate itemsets with low support, the chance that an itemset appears due to noise or just by coincidence is higher than for candidate itemsets with higher support. Itemsets that occur within the database due to coincidence do not add any meaningful information and, therefore, should not be considered when we are searching for rare itemsets using Apriori-Inverse. The minimum absolute support value is used to filter out these candidate items. The value varies for different candidates; the minimum absolute support value for items that have a higher support is generally higher. The minimum absolute support value is dependent solely on the support of the individual items.

4.1 Minimum Absolute Support Value

When searching for rare itemsets, two circumstances are considered: occurrences of itemsets due to some non-random process that is generating them or occurrences of itemsets by coincidence. It is important to determine this, as itemsets that have a low support but high confidence that seem interesting may be occurring by chance and should be considered as noise.

Clearly, it makes sense only to consider candidate itemsets that appear together more often than coincidence. Coincidence is defined in this manner: for *N* transactions in which antecedent *A* occurs in *a* transactions and consequent *B* occurs in *b* transactions, the probability that *A* and *B* will occur together exactly *c* times by chance can be calculated. It is referred to this as probability of collision chance Pcc. It can be calculated using equation (1). The probability that *A* and *B* will occur together exactly *c* times is:

$$P_{cc}(c|N,a,b) = \frac{\binom{a}{c}\binom{N-a}{b-c}}{\binom{N}{b}} \quad (1)$$

This equation is the usual calculation for exact probability of a 2×2 contingency table. Now, we want the least number of collisions above which Pcc is smaller than some small value *p* (usually 0.001). This is:

$$MinAbsSup(N,a,b) = \{m \mid \sum_{i=0}^{m} Pcc(i|N,a,b) \geq 1 - p\} \quad (2)$$

This formula amounts to invert the usual sense of Fisher's exact test [25]. Usually a 2×2 contingency table is provided and a p-value calculated; however, here we are providing two of the four values and a p-value and calculating the minimum value to complete the table.

5. Enhanced Apriori Algorithm

Here we developed an algorithm that converts the database into array of zeros and ones, bitmap. Calculate the support value for each element, and then the minimum support (min-supp) value. Minimum support value is calculated based on aggregation functions, Simple Mean, Mean Square Error, and Standard Deviation. Infrequent items are the items that have low support value that lies on the extra small region defined by Standard Deviation parameter.
After pruning elements (non-frequent items) which have support value below the min-supp value, we use frequent items to generate rules, and then calculate the confidence value for each rule, and the minimum confidence (min-conf) value. Minimum confidence value is calculated based on Simple Mean, Mean Square Error, and Standard Deviation. Prune rules which have confidence below the min-conf. (pruning non-frequent rules). Again, generate next pass rules, combining frequent rules with frequent items. The algorithm is illustrated in the flowchart shown in Figure2; Figure 3 illustrates calculating elements support count for every single item in the database, while Figure 4 illustrates the function that calculates minimum support threshold value.

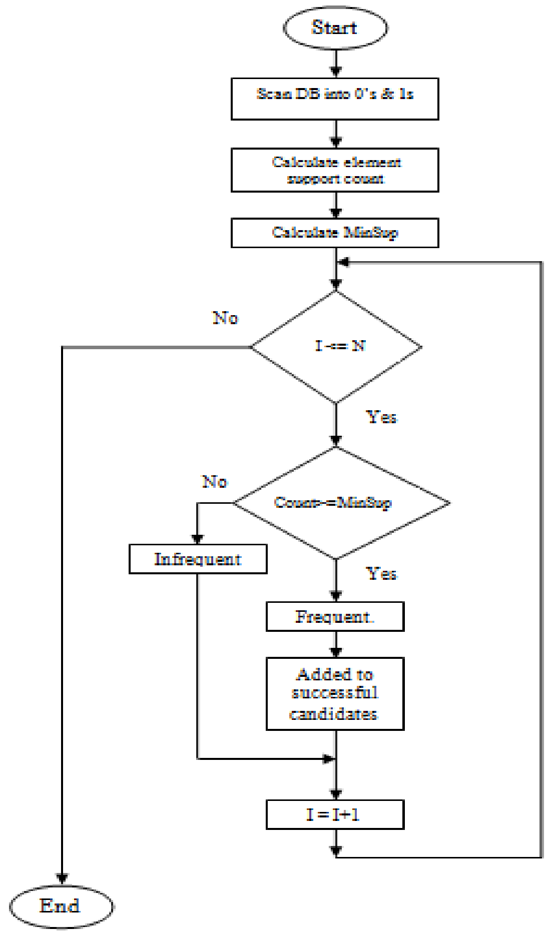

Figure2. The proposed enhanced Apriori Algorithm

Deterministic-function pruning based enhanced Apriori algorithm

Input: Transaction database *D*

Output: Non-coincidental frequent itemsets

```
For all transactions $t \in D$
{
Ct=subset ($C_1$, t);
For all candidates $c \in Ct$
        c.count++
}
L1= Min_sup(C1);
For (k=2; $L_{k-1} \neq \varphi$;k++)
{
$C_k$=Apriori ($L_{k-1}$);
For all transactions $t \in D$
{
Ct=subset ($C_k$,t);
For all candidates $c \in Ct$
        c.count++
$L_k$ = {MinSup(Ck)}
```

```
    }
    Return ($\cup_k L_k$ )
}
```

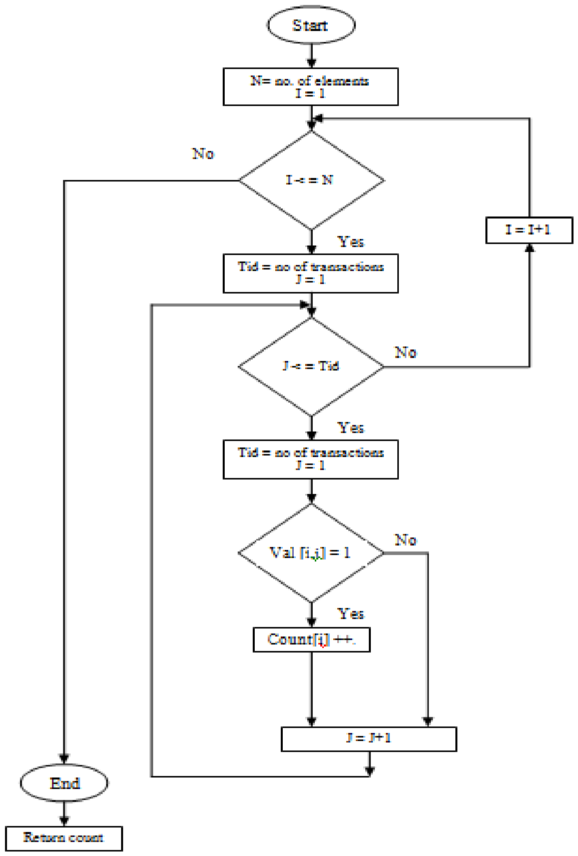

Figure3. Calculating Element Support Count

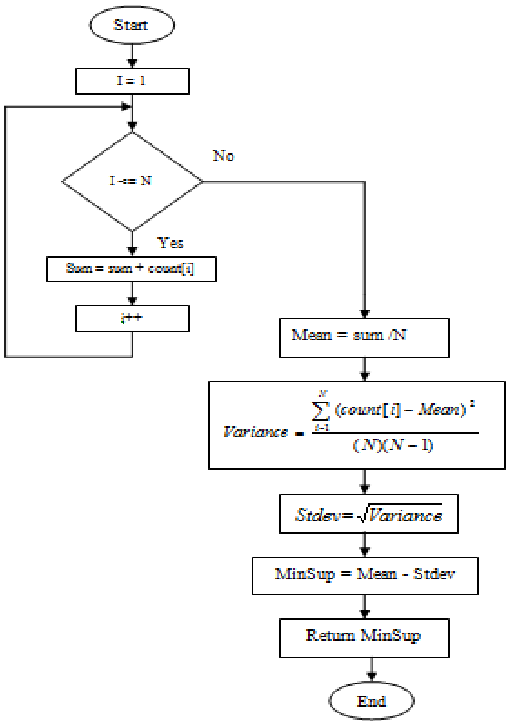

Figure4. Calculating Minimum Support Threshold Value

6. Experiments and Discussion

For an example of a basket market database of 9 transactions shown in Table 1 [8] is repeated for the sake of qualitative comparison.

Table 1: Basket Market Dataset

Tid	Items
1	Book- CD - Video
2	CD – Game
3	CD- DVD
4	Book – CD – Game
5	Book – DVD
6	CD- DVD
7	Book – DVD
8	Book – CD – DVD
9	Book – CD – Video – DVD

Turning it into an array of zeros and ones would produce Table 2.

Table 2: Bit Map

Tid	book	CD	Video	Game	DVD
1	1	1	1	0	0
2	0	1	0	1	0
3	0	1	0	0	1
4	1	1	0	1	0
5	1	0	0	0	1
6	0	1	0	0	1
7	1	0	0	0	1
8	1	1	1	0	1
9	1	1	0	0	1

Applying the Algorithm on the previous bitmap would give the results shown in Table 3.

Table 3: Illustrative Example Result

1	1	1	0	0
0	1	0	1	0
0	1	0	0	1
1	1	0	1	0
1	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	1	1	0	1
1	1	0	0	1

Sup count	6	7	2	2	6
Min support	2.191681084				
Status	frequent	frequent	Non frequent	Non frequent	Frequent

By pruning non frequent itemsets:
After pruning non frequent items as shown in Table 4, then combining frequent elements to produce 2-itemset on the first iteration, and then find out Min-confidence to determine frequent/Non-frequent rules.

Table 4: Frequent Bit Itemsets

Tid	book	CD	DVD
1	1	1	0
2	0	1	0
3	0	1	1
4	1	0	0
5	1	1	1
6	0	1	1
7	1	0	1
8	1	1	1
9	1	1	1

Combining frequent Itemsets to generate rules as follow and shown in Table 5.

Table 5: Rules Generation of frequent Items

	Book – CD	DVD - Book	CD – DVD
	1	0	0
	0	0	0
	0	0	1
	1	0	0
	0	1	0
	0	0	1
	0	1	0
	1	1	1
	1	1	1
Rule Confidence	0.67	0.67	0.57
Min Confidence	0.57		
Status	Frequent	frequent	Frequent

We can conclude three rules of the 1st iteration which are:

- Rule#1: 66.667% of transactions that contains Book also contains CD.
- Rule#2: 66.667% of transactions that contains Book also contains DVD.
- Rule#3: 57% of transactions that contains CD also contain DVD.

On the 2nd iteration:

A single rule produced that 50% of transactions that contain Book-CD also contain DVD

Based on the achieved results as shown in Table 6, the developed algorithm is faster than Apriori as it reduced the number of frequent itemsets, number of rules to be checked by iteration, and number of iterations.

Table 6: Comparative study between the developed and Apriori algorithms

	Apriori (min supp = 0, Min conf= 50%)	**Developed Algorithm**
No of frequent items	**5**	**3**
No of rules to be checked 1st iteration	**10**	**3**
No of successful rules on the 1st iteration	**4 (video>>DVD ‘ Week Rule’)**	**3**
No of rules to be checked 2nd iteration	**7**	**1 (Conf = 50%)**
No of successful rules on the 2nd iteration	**3 (2 week rules ’book-CD>video’)**	**--**
No of rules to be checked 3rd iteration	**3**	**--**
No of successful rules on the 3rd iteration	**1 (book-CD-video>>DVD) conf100%**	**--**
Total rule no.	**8**	**4**

Also, an Apriori may produce huge number of rules which are redundant rules and considered to be week rules; in our illustrated example the Apriori produced rules such as:

Video >> DVD (conf 50%)
Book-video >> DVD (conf 50%)
CD-Video >> DVD (conf 50%)

Which are week, uninteresting rules.

7. Discretizing continuous valued attributes based on probability of collision:

Furthermore, we compared the performance of the proposed MinAbsSup function, Probability of collision, and the standard Apriori algorithm with bitmapping technique and probability of collision on dataset from the UCI Machine Learning Repository [25].

Lenses database: Database for fitting contact lenses, with multivariate characteristics, and categorical attributes. It has four attributes (Age, Prescription, Astigmatic, and Tear Production Rate) with no missing data, and 24 instant. The data set is shown below in Table 7.

Table 7: Lenses Data Base

T-ID	Age	Prescription	Astegmatic	Tear Rate	Lenses
1	Young	Myope	No	Reduced	No Lense
2	Young	Myope	No	Normal	Soft Lense
3	Young	Myope	Yes	Reduced	No Lense
4	Young	Myope	Yes	Normal	Hard Lense
5	Young	hypermetrope	No	Reduced	No Lense
6	Young	hypermetrope	No	Normal	Soft Lense
7	Young	hypermetrope	Yes	Reduced	No Lense
8	Young	hypermetrope	Yes	Normal	Hard Lense
9	pre-presbyopic	Myope	No	Reduced	No Lense
10	pre-presbyopic	Myope	No	Normal	Soft Lense
11	pre-presbyopic	Myope	Yes	Reduced	No Lense
12	pre-presbyopic	Myope	Yes	Normal	Hard Lense
13	pre-presbyopic	hypermetrope	No	Reduced	No Lense
14	pre-presbyopic	hypermetrope	No	Normal	Soft Lense
15	pre-presbyopic	hypermetrope	Yes	Reduced	No Lense
16	pre-presbyopic	hypermetrope	Yes	Normal	No Lense
17	presbyopic	Myope	No	Reduced	No Lense
18	presbyopic	Myope	No	Normal	No Lense
19	presbyopic	Myope	Yes	Reduced	No Lense
20	presbyopic	Myope	Yes	Normal	Hard Lense
21	presbyopic	hypermetrope	No	Reduced	No Lense
22	presbyopic	hypermetrope	No	Normal	Soft Lense
23	presbyopic	hypermetrope	Yes	Reduced	No Lense
24	presbyopic	hypermetrope	Yes	Normal	No Lense

Coding the four attributes has been done as follows:

1. Age of the patient: (1) young, (2) pre-presbyopic, (3) presbyopic
   2. Spectacle prescription: (1) myope, (2) hypermetrope
   3. Astigmatic: (1) no, (2) yes
   4. Tear production rate: (1) reduced, (2) normal

Table 8: Lenses Database Bitmapping

T-ID	Age			Prescription		Astegmatic		Tear Rate		Lenses		
	1	2	3	1	2	1	2	1	2	1	2	3
1	1	0	0	1	0	1	0	1	0	0	0	1
2	1	0	0	1	0	1	0	0	1	0	1	0
3	1	0	0	1	0	0	1	1	0	0	0	1
4	1	0	0	1	0	0	1	0	1	1	0	0
5	1	0	0	0	1	1	0	1	0	0	0	1
6	1	0	0	0	1	1	0	0	1	0	1	0
7	1	0	0	0	1	0	1	1	0	0	0	1
8	1	0	0	0	1	0	1	0	1	1	0	0
9	0	1	0	1	0	1	0	1	0	0	0	1
10	0	1	0	1	0	1	0	0	1	0	1	0
11	0	1	0	1	0	0	1	1	0	0	0	1
12	0	1	0	1	0	0	1	0	1	1	0	0
13	0	1	0	0	1	1	0	1	0	0	0	1
14	0	1	0	0	1	1	0	0	1	0	1	0
15	0	1	0	0	1	0	1	1	0	0	0	1
16	0	1	0	0	1	0	1	0	1	0	0	1
17	0	0	1	1	0	1	0	1	0	0	0	1
18	0	0	1	1	0	1	0	0	1	0	0	1
19	0	0	1	1	0	0	1	1	0	0	0	1
20	0	0	1	1	0	0	1	0	1	1	0	0
21	0	0	1	0	1	1	0	1	0	0	0	1
22	0	0	1	0	1	1	0	0	1	0	1	0
23	0	0	1	0	1	0	1	1	0	0	0	1
24	0	0	1	0	1	0	1	0	1	0	0	1

And: Lenses fitting
1: the patient should be fitted with hard contact lenses,
2: the patient should be fitted with soft contact lenses,
3: the patient should not be fitted with contact lenses.

Data set can be discretized and converted to simpler form to increase the speed of data processing by bitmapping as shown in Table 8.

When Apriori with MinAbsSup is compared against Apriori, the reduction in the number of rules (with all possible consequent lengths) generated is drastic. The reduction ranges from a factor of 15 to 60809, depending on the particular dataset. By setting the arbitrary threshold too low, we may be flooded with many trivial rules. We would need wade through the rules to find those that may be of some interest. However setting the support too high, we may miss out useful rules. To take the Lenses dataset as an example, normal Apriori finds 83 rules. The list below shows a subset of the rules found using normal Apriori with its summation of probabilities of collision, and the Absolute Minimum Support value. We concentrate on this particular subset because they contain a similar consequent. The rest of the rules in the subset were not found as the itemsets could not be differentiated from noise.

Based on bitmap and probability of collision, where N is the total number of transaction (N= 24), c is the number of particular times items A, and B occur together in the database, *a* is the number of times item A appears in the database (*a* = sup(A)), and *b* is the number of times item B appears in the database (*b* = sup(B)). Pcc – refer to Equation (1) - represents the probability that A, and B occur together exactly *c* times.

Addressing all different cases would produce sets of successful itemsets combination are produced supported by the summation of probabilities of collision, and the Absolute Minimum Support value.

The achieved rules are listed below:

$\{Age = 1\} \rightarrow \{Perception = 1\}$ 0.998, 7
$\{Age = 1\} \rightarrow \{Perception = 2\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Perception = 1\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Perception = 2\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Perception = 1\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Perception = 2\}$ 0.998, 7
$\{Age = 1\} \rightarrow \{Astigmatic = 1\}$ 0.998, 7
$\{Age = 1\} \rightarrow \{Astigmatic = 2\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Astigmatic = 1\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Astigmatic = 2\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Astigmatic = 1\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Astigmatic = 2\}$ 0.998, 7
$\{Age = 1\} \rightarrow \{Tear\ Production\ Rate = 1\}$ 0.998, 7
$\{Age = 1\} \rightarrow \{Tear\ Production\ Rate = 2\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Tear\ Production\ Rate = 1\}$ 0.998, 7
$\{Age = 2\} \rightarrow \{Tear\ Production\ Rate = 2\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Tear\ Production\ Rate = 1\}$ 0.998, 7
$\{Age = 3\} \rightarrow \{Tear\ Production\ Rate = 2\}$ 0.998, 7
$\{\text{Spectacle prescription} = 1\} \rightarrow \{\text{Astigmatic} = 1\}$ 0.998, 9
$\{\text{Spectacle prescription} = 1\} \rightarrow \{\text{Astigmatic} = 2\}$ 0.998, 9

{Spectacle prescription = 2} → {Astigmatic = 1} 0.998, 9
{Spectacle prescription = 2} → {Astigmatic = 2} 0.998, 9
{Spectacle prescription = 2} → {tear production rate = 3} 0.998, 9
{Spectacle prescription = 2} → {tear production rate = 3} 0.998, 9
{Spectacle prescription = 2} → {tear production rate = 3} 0.998, 9
{Spectacle prescription = 2} → {tear production rate = 3} 0.998, 9
{Astigmatic = 1} → {tear production rate = 3} 0.998, 9
{Astigmatic = 1} → {tear production rate = 3} 0.998, 9
{Astigmatic = 1} → {tear production rate = 3}0.998, 9
{Astigmatic = 1} → {tear production rate = 3} 0.998, 9
{Spectacle prescription = 1} → {Lenses = 3} 0.9957, 10
{Spectacle prescription = 2} → {Lenses = 3} 0.9957, 10
{Astigmatic = 1} → {Lenses = 3} 0.9957, 10
{Astigmatic = 2} → {Lenses = 3} 0.9957, 10
{Tear production rate = 1} → {Lenses = 3} 0.9957, 10
{Tear production rate = 2} → {Lenses = 3} 0.9957, 10

By applying these rules with its significant MinAbsSup Value, we can prune all unnecessary and unsuccessful rules and obtain a few successful rules that really present valuable information in the database. From this particular grouping Apriori with MinAbsSup, it finds that [{ tear production rate = 1} → {Lenses = 3} 0.9957, 10] is a successful Rule.

8. Conclusions

In this paper, we developed a function model which replaces user defined minimum support threshold value of standard Apriori. This function calculates a custom minimum support for each itemset based on the itemset's statistics, CLT, preventing coincidental rules from being generated. The achieved simulated results showed that the proposed function efficiently finds minimum number of rules which are non-coincidental without using arbitrary support thresholds.

Furthermore, on another approach this paper applied bitmapping technique into a statistical probability of collision algorithm proposed on Apriori-Inverse.

The rules generated by normal Apriori should not be considered as the most compact set of rules. In order to obtain a compact set of rules, some forms of post-pruning method to eliminate trivial and redundant rules have been presented. The achieved results show that MinAbsSup applied with bitmapping reduces the time and space requirements.

A custom minimum support for each itemset has been calculated based on the itemset's probability of chance collision, preventing coincidental rules from being generated. MinAbsSup associated with bitmapping efficiently finds non-redundant rules which are non-coincidental by setting a suitable threshold without using arbitrary support thresholds.

References:

[1] H. Mahgoub,"Mining association rules from unstructured documents" in Proc. 3rd Int. Conf. on Knowledge Mining, ICKM, Prague, Czech Republic, Aug. 25- 27, 2006, pp. 167-172.

[2] S. Kannan, and R. Bhaskaran "Association rule pruning based on interestingness measures with clustering". International Journal of Computer Science Issues, IJCSI, 6(1), 2009, pp. 35-43.

[3] M. Ashrafi, D. Taniar, and K. Smith "A New Approach of Eliminating Redundant Association Rules". Lecture Notes in Computer Science, Volume 3180, 2004, pp. 465 – 474.

[4] P. Tang, M. Turkia "Parallelizing frequent itemset mining with FP-trees". Technical Report titus.compsci.ualr.edu/~ptang/papers/par-fi.pdf, Department of Computer Science, University of Arkansas at Little Rock, 2005.

[5] M. Ashrafi, D. Taniar, and K. Smith "Redundant Association Rules Reduction Techniques". Lecture Notes in Computer Science, Volume 3809, 2005, pp. 254 -263.

[6] M. Dimitrijevic, and Z. Bosnjak "Discovering interesting association rules in the web log usage data". Interdisciplinary Journal of Information, Knowledge, and Management, 5, 2010, pp.191-207.

[7] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo.: Fast discovery of association rules- In Advances in Knowledge Discovery and Data Mining (1996).

[8] Z. HONG-ZHEN, C. DIAN-HUI, and, Z. DE-CHEN "Association Rule Algorithm Based on Bitmap and Granular Computing". AIML Journal, Volume (5), Issue (3), September, 2005.

[9] K. Yun Sing "Mining Non-coincidental Rules without a User Defined Support Threshold". 2009.

[10] C. Yin-Ling and F. Ada Wai-Chee "Mining Frequent Itemsets without Support Threshold: With and without Item Constraints". 2004.

[11] S. Brin, R. Motwani, and C. Silverstein "Beyond market baskets: generalizing association rules to correlations". SIGMOD Rec. 26(2), 1997, Pp. 265–276.

[12] R. Meo, R "Theory of dependence values". ACM Trans. Database Syst. 25(3), Pp. 380–406, 2000.

[13] X. Wu, C. Zhang, and S. Zhang "Efficient mining of both positive and negative association rules". ACM Trans. Inf. Syst. 22(3), Pp. 381– 405, 2004.

[14] J. Han, J. Pei, and Y. Yin.: "Mining Frequent Patterns without Candidate Generation". 2000.

[15] R. Agrawal, T. Imielinski, and A. Swami, ^{a}Mining Association Rules Between Sets of Items in Large Databases,º Proc. 1993 ACM SIGMOD Int'l Conf. Management of Data, pp. 207- 216, Washington, D.C., May 1993.

[16] R. Agrawal and R. Srikant, ^{a}Fast Algorithms for Mining Association Rules,º Proc. 1994 Int'l Conf. Very Large Data Bases, pp. 487-499, Santiago, Chile, Sept. 1994.

[17] Pramod S., O.P. Vyas: Survey on Frequent Item set Mining Algorithms. International journal of computer applications, 2010, pp. 86-91.

[18] Sergey Brin, Rajeev Motwani: Dynamic itemset counting and implication rules for market basket data 1997

[19] R. Agrawal, Heikki Mannila Fast Discovery for Mining Association Rules

[20] J. Hipp, U. Güntzer, and U. Grimmer.: Algorithms for Association Rule Mining: General Survey 2001.

[21] Ashok Savasere. Edward Omiecinski. Shamkant Navathe. An Efficient Algorithm for Mining Association Rules in Large Database. 21th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1995.

[22] Zaki M.,et al, New Algorithms For Fast Discovery of Association Rules, In KDD97. Technical report, University of Rochester Rochester, NY, USA, 1997.

[23] HONG-ZHEN ZHENG, DIAN-HUI CHU, DE-CHEN ZHAN : Association Rule Algorithm Based on Bitmap and Granular Computing. AIML Journal, Volume (5), Issue (3), September, 2005, pp. 51-54.

[24] Yun Sing Koh, Nathan Rountree, Richard O'Keefe: Finding Non-Coincidental Sporadic Rules Using Apriori-Inverse. International Journal of Data Warehousing & Mining, 2(2), 38-54, 2006.

[25] Weisstein, E. (2005). Fisher's exact test. MathWorld — A Wolfram Web Resource. Retrieved October 5, 2005.

Medhat Awadalla is an assistant professor at Electrical and Computer Engineering Department, Sultan Qaboos University. He obtained his PhD from university of Cardiff, UK. Msc and Bsc from Helwan university, Egypt. His research interest includes cloud computing, sensor networks, high performance computing and real time systems.

Sara Elfar is an research student at college of Engineering, Helwan University. She had obtained her BSc in computer science and engineering from university of Helwan, Egypt. Her research interest includes Data Mining, high performance computing, and embedded systems.