Association Rule Mining for Both Frequent and Infrequent Items Using Particle Swarm Optimization Algorithm

MIR MD. JAHANGIR KABIR[1], SHUXIANG XU[2], BYEONG HO KANG[3], ZONGYUAN ZHAO[4]
School of Computing and Information Systems
UNIVERSITY OF TASMANIA, AUSTRALIA.
E-mail: mmjkabir@utas.edu.au[1],Shuxiang.Xu@utas.edu.au[2],Byeong.Kang@utas.edu.au[3],
Zongyuan.Zhao@utas.edu.au[4]

***Abstract*-- In data mining research, generating frequent items from large databases is one of the important issues and the key factor for implementing association rule mining tasks. Mining infrequent items such as relationships among rare but expensive products is another demanding issue which have been shown in some recent studies. Therefore this study considers user assigned threshold values as a constraint which helps users mine those rules which are more interesting for them. In addition, in real world users may prefer to know relationships among frequent items along with infrequent ones.**

The particle swarm optimization algorithm is an important heuristic technique in recent years and this study uses this technique to mine association rules effectively. If this technique considers user defined threshold values, interesting association rules can be generated more efficiently. Therefore this study proposes a novel approach which includes using particle swarm optimization algorithm to mine association rules from databases. Our implementation of the search strategy includes bitmap representation of nodes in a lexicographic tree and from superset-subset relationship of the nodes it classifies frequent items along with infrequent itemsets. In addition, this approach avoids extra calculation overhead for generating frequent pattern trees and handling large memory which store the support values of candidate item sets.

Our experimental results show that this approach efficiently mines association rules. It accesses a database to calculate a support value for fewer numbers of nodes to find frequent itemsets and from that it generates association rules, which dramatically reduces search time. The main aim of this proposed algorithm is to show how heuristic method works on real databases to find all the interesting association rules in an efficient way.

Keywords-Particle Swarm Optimization; Data mining; genetic algorithm;frequentitemsets;lexicographic tree.

I. INTRODUCTION

Data in different type of information databases including business transaction data, scientific data, financial data, medical data and so on contain hidden information. Analysing and discovering critical hidden information from these sort of databases have become one of the important issues in recent data mining techniques.A transactional database consists of set of items and inventing the relationships among these items is the main goal of association rule mining. The knowledge which comes from application of association rule mining in industrial databases is useful to explain industrial failures.

Association rule mining includes two steps: first to mine frequent itemsets from large database and second is to generate association rules or correlation relationship among a large set of data items. Nowadays, huge amounts of data are collected and stored by industries. The discovery of association rules among large amount of business transactions helps industries make decisions.

The traditional approaches reveal valid association rules by using support and confidence values of itemsets in a database. To prune the search space it uses a minimum support value as a threshold. Two main problems arise because of using such mechanisms. (1) If users set minimum support value too low then it increases the computational complexity such as generation of candidate itemsets, complexity of designing large number of tree nodes, testing of nodes and so on. Finally it generates a large number of association rules and traditional algorithms suffer poor performance because of these large numbers of rules. (2) If users set minimum support value too high, many interesting rules with low support value are missed. Such association rules are important to discover relationship among expensive items such as diamond or gold necklaces, ear rings, bracelets. These rules are also important for identifying such web documents which are identical or similar.

Recently, some researchers developed algorithms to mine association rules without a minimum support value constraint[1]–[3].The approaches proposed in these papers used confidence based pruning mechanism instead of support based pruning technique which is adopted by traditional association rule mining methods. Support free association rule mining techniques discover high, cross and low support based rules. Itemsets with high support

values are well known patterns. Patterns containing items with cross support value have poor correlation. On the other side, patterns with a low support value provide precious insights.

II. BASIC CONCEPTS AND TERMINOLOGY

Let $D = \{t_1, t_2, t_3, \ldots., t_k, \ldots t_n\}$ be the database, where $t_1, t_2, \ldots\ldots., t_n$ are the number of transactions in the database. Each transaction t_n is a set of items $I = \{i_1, i_2, \ldots., i_k, \ldots i_n\}$, where i_1 is item number 1, i_n is item number n and so on. Transaction t_n is represented as a binary vector. If $t_n[k] = 1$ it means that t_n bought the item i_k , otherwise $t_n[k] = 0$. Let X be a set of few items in I i.e. $X \subseteq I$ .The set $t_n(X) \subseteq I$ is true for all items in itemset X for transaction t_n. The support value of an item is how many times the item appears in the transaction database as a subset. The support value of an itemset is denoted by $\sigma(X) = |\{t_1(X) + t_2(X) + t_3(X) \ldots\ldots\ldots t_{n-1}(X) + t_n(X)\}|/|D|$. Here $t_n(X)$ gives the binary value. If the examined itemset X appears as a subset in a transaction t_n, then $t_n(X) = 1$, otherwise = 0. An itemset with 1 item is called 1- itemset, an itemset with k-items is called k-itemset. An itemset is called frequent if its support value is more than or equal to the user defined threshold value is denoted by min_supp(minimum support) i.e. $\sigma(X) \geq \min\_supp$. We denote the frequent itemset by FI. If an itemset X is frequent and no superset of X is frequent then we can conclude that X is a maximal frequent itemset and we denote the set of all maximal frequent itemsets by MFI. If X and Y are sets of items, the confidence value of itemsetX and Y is how many times itemset X appears as well as Y with regard to the total number of transactions in the database containing X. An association rule is an expression of the form X→Y, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. A rule is valid or strong if the support and confidence value of that rule are greater than the predefined threshold value set by the user. Most association rule mining approaches consider support-confidence framework to disclose interesting rules.

Table: 1 Purchase of Science Fiction and Books in a Supermarket

	Science fiction	**¬Science Fiction**	$\sum_{row}$
Books	300	200	500
¬Books	150	100	250
$\sum_{column}$	450	300	750

If we transform this table into its binary form it then looks like the following table:

Table: 2 Purchase of Science Fiction and Books in a Supermarket (Binary Form)

Transaction ID	**Science Fiction**	**Books**	**¬Science Fiction**	**¬Books**
1	1	1	0	0
2	1	0	0	1
3	0	1	1	0
4	0	0	1	1
…	…	…	…	…
1000	0	1	1	0
Support Value	450	500	300	250

If users set a minimum support of 30% and a minimum confidence of 65% then we get the following 8 rules.

1) Science Fiction → Books[sup = 30%, conf = 67%]
2) Books → Science Fiction [sup = 30%, conf = 60%]
3) Science Fiction → ¬Books[sup = 15% , conf = 33%]
4) ¬Books → Science Fiction[sup = 15%, conf = 60%]
5) Books → ¬Science Fiction[sup = 20%, conf = 40%]
6) ¬Science Fiction → Books[sup = 20%, conf = 67%]
7) ¬Science Fiction → ¬Books[sup = 10%, conf = 33%]
8) ¬Books → ¬Science Fiction[sup = 10%, conf = 40%]

To generate association rules based on user interest form a large database is a most time consuming task in the present day. We present a novel approach to find association rules from large databases by using the principles of particle swarm optimization algorithm.

The main contributions of this research paper are as follows: (1) We propose a new algorithm including traditional particle swarm optimization algorithm to mine association rules from frequent itemsets, (2) These frequent itemsets are searched from lexicographic trees which are based on user define threshold fitness values, and (3) Our scheme mines interesting rules not only for two or three itemsets but also for large itemsets.

III. RELATED WORKS

It is well known that Apriori algorithm generates a candidate set and tests it in a breadth fast manner. It discovers all the frequent itemsets at level k before moving to its next level (k+1). It counts the support value of each node in level k and prunes those nodes if the support values of those nodes do not satisfy a user define support value. It generates candidate itemsets at each level and scans the database so frequently that it is costly, especially when there exists a long pattern [4]. After generating frequent itemsets, Apriori algorithm generates association rules. If the confidence values of frequent itemsets are greater than the predefined minimum confidence, association rules under these frequent itemsets are generated.

Sampling algorithm[5] is proposed by Toivonen in 1996. This algorithm finds association rules by reducing database activity. Level-wise method is applied to samples by considering the minimal support threshold value. This approach generates exact association rules but in a few cases it does not guarantee that it will generate all the association rules. In most cases this approach needs one full pass over the database, but in a worst case it needs two passes.

Brin et al. proposed a new approach called dynamic itemset counting algorithm[6] in 1997. The design motivation of this algorithm is to limit accessing the main database. This approach partitions the whole database into several blocks labelled by starting point and scans the database repeatedly. This algorithm generates association rules which are normalized and it provides profound intuitive results with respect to other methods. Finally they showed the effect of real data sets on the performance of the system.

Pincer-Search algorithm [7]traverses a lattice through a bi-directional method that follows both top-down and bottom-up approaches. To find maximal frequent itemsets it applies pruning methods by the following two properties:

1) All the subsets of frequent itemsets are pruned
2) All the supersets of infrequent itemsets are pruned

This approach is used to maintain and update the data structure which is designed for this study named maximum frequent candidate set. Through bottom up approach it prune the candidate itemsets. Another important characteristic of this algorithm, it does not need to explicitly test all the frequent itemsets.

Breadth first traversal (a level by level search strategy on a search space) is applied for a MaxMiner search algorithm. To prune the branches of a tree it performs a look-ahead method. MaxMiner uses breadth first approach for limiting the number of passes over the database but look-ahead, which involves superset pruning, works better for depth first search methods[8].

DepthProject performs depth first traversal on a lexicographic tree along with variations of superset pruning. To order child nodes, it applies dynamic reordering methods. By trimming infrequent items out of each node's tail, it reduces the size of the search space. To eliminate non-maximal frequent itemsets DepthProject would require post pruning methods[9].

MAFIA, proposed by Burdick, Calimlim, and Gehrke[10], extends the idea of DepthProject. Similar to DepthPorject, MAFIA also uses vertical bitmap representation where the support value/count of an itemset is based on AND operations among the itemsets. For example, if there are 4 items in a data tuple and the database is as follows

A	B	C	D
1	0	1	1
0	1	1	1
1	0	1	0
1	0	1	1
1	1	0	1

Figure 1: Vertical bitmap representation

Bitvectors for itemset A,B,C,D of Figure 1 are 10111, 01001, 11110, 11011 respectively. To get the support value/count of the itemset it needs to apply bitwise AND (&) operation between the bitvectors of the itemsets. For the above example, the result of bitwise AND operation of bitvectors of items A and C is 10111 & 11110, which equals to 10110. The support value or count of an item is the number of 1's in the bitvector. Here the support value of itemset {A, C} is 3. If we want to add another bitvector D with the previous result of bitwise AND operation of bitmap {A,C}, it equals to 10110 & 11011, which equals to 10010. The support value of the itemset {A,C,D} is 2. The search strategy of MAFIA integrates depth first method to traverse the tree to find maximal frequent itemsets along with effective pruning methodology. Look-ahead pruning methodology which was first used by MaxMiner is also used by MAFIA. The last checking method of MAFIA is easy to test. Without counting $A \cup C$, it allows us to conclude that {A,C} is frequent. This technique is defined as Parent Equivalence Pruning.

In[11], Gouda and Zaki proposed a novel approach called GENMAX to find maximal itemsets. In this approach they used a novel technique called Progressive Focusing. This technique maintains local maximal frequent itemsets (LMFI) which is used for making comparison with newly found frequent itemsets (FI). Non maximal frequent itemsets are identified through this step and it decreases the number of subset testing. GENMAX uses vertical representation of a database and stores transaction identifier set (TIS) for each itemset instead of bitvector. The support value of an itemset is defined by the cardinality of an itemset's TIS. Researchers of GENMAX concluded that, through experimental results this algorithm performs better than existing algorithms on different types of databases.

Bilal Alataş and Erhan Akin [12] designed an efficient genetic algorithm as a search strategy to mine both positive and negative quantitative association rules. Association rules are deduced from frequent patterns. Their approach is different than other methods. This method mined the association rules without generating frequent itemsets. The proposed genetic algorithm does not depend on minimum support and confidence value which is hard to define for a database. A new genetic operator named uniform operator is used in this approach which ensures genetic diversity.

To mine quantitative association rules researchers proposed a new algorithm which is based on genetic algorithm named QUANTMINER[13], [14]. By optimizing support and confidence value, this system dynamically identify good intervals in association rules. Researchers applied this algorithm in different data sets and showed the usefulness of this algorithm as a data mining tool.

R.J.kuo and C.W.Shih used new meta-heuristic technique name ant colony system[15] to mine large database for efficient searching of association rules. Multi-dimensional constraints are considered in this approach. In addition this approach also considered user's assign constraint. The results showed that it gave more condensed rules than Apriori algorithm. The computational time of this approach is less than Apriori algorithm. Though this system provides promising results but this system still faces some issues which need to resolve. After analysing the results it found that lots of similar rules are generated so the researchers suggested another technique like fuzzy approach to merge those similar rules into one class.

To mine association rules most researchers focused on ameliorating computational efficiency. To determine the threshold values of support and confidence which are the key factors for association rule mining task, researchers proposed a new approach which is based on particle swarm optimization technique. Suitable fitness values and their corresponding support and confidence values of identified swarms are searched through this approach. Their result showed that particle swarm optimization algorithm quickly finds suitable threshold fitness values of itemsets and quality rules are obtained through this way. Users can mine specific rules from a large database by setting support or fitness value. Since this technique free from support constraint, the main problem of this approach is users have no control over mining techniques. Apart from this their result only showed two or three dimensional rules instead of more dimensional rules which could be interesting for the policy makers of the industry[16].

IV. DATABASE REPRESENTATION

A. Bipartite Graph and Bitmap representation

If U and V are disjoint sets of vertices and E is the set of edges which connect the vertices U and V, then we can represent a bipartite graph as a triple, i.e. $G = (U, V, E)$ where $E \subseteq U \times V$.

A binary matrix is a matrix of $m \times n$, where each entry consists of a value which is either 0 or 1. Mapping between binary matrices and databases of transactions can be done in a straight forward way. Consider a database D which consists of transactions $\{t_1, t_2, \dots, t_{m-1}, t_m\}$ corresponding to rows and Items $\{i_1, i_2, \dots, i_{n-1}, i_n\}$ corresponding to columns. The database D is a $m \times n$ matrix, where each entry is defined as a_{ij}. The value of a_{ij} is 1, if transaction t_i contains item i_j otherwise it is 0. Now we are mapping each transaction as a set of items from the binary matrices.

Example 1: Consider a database D which consists of the following transactions t_1, t_2, t_3, t_4, t_5 and items i_1, i_2, i_3, i_4, where $t_1 = \{i_1, i_2, i_3\}$,$t_2 = \{i_1, i_2, i_3, i_4\}$,$t_3 = \{i_1, i_3, i_4\}$,$t_4 = \{i_1, i_3, i_4\}$ and $t_5 = \{i_1, i_2, i_3, i_4\}$. Here all the items are different. Figures 2 and 3 show the bipartite graph and the binary matrix of the database D:

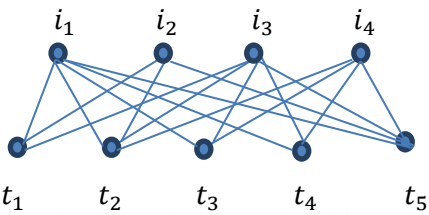

Figure 2: Bipartite Graph Representation of the Database D

	i_1	i_2	i_3	i_4
t_1	1	1	1	0
t_2	1	1	1	1
t_3	1	0	1	1
t_4	1	0	1	1
t_5	1	1	1	1

Figure 3: Binary Matrix Representation of the Database D

From Figure 2, if we map each transaction by items then the transactions are as follows:

$t_1 = \{1,1,1,0\} = 1110$

$t_2 = \{1,1,1,1\} = 1111$

$t_3 = \{1,0,1,1\} = 1011$

$t_4 = \{1,0,1,1\} = 1011$

$t_5 = \{1,1,1,1\} = 1111$

B. Lexicographic Tree

Our research problem here is to find frequent itemsets from large databases using Particle Swarm Optimization Algorithm. Itemset I consists of n items, i.e. $I = \{i_1, i_2, i_3, \ldots, i_n\}$. $X_k$represents an itemset containing k-items , where k = 1,2,…,n and $X_k \subseteq I$. If k=1, then X_k contains a 1-item, i.e., $X_1 = \{i_1\}$. If k=2, then $X_k$contains 2-items, i.e., $X_2 = \{i_3, i_4\}$, and so on. An itemset is called frequent if its support value satisfies a user defined support value and it is denoted FI. In this paper we will consider a search space which consists of all feasible solutions. A Lexicographic tree [10], [17]is the search space for this study. This tree maintains lexicographic ordering of items I in a database D. If item i occurs before item j of in a database D, then it maintains lexicographic ordering, i.e., $i \leq_L j$. If two subsets $S_1$and S_2 , where $S_1 \subseteq S_2 and\ S_1, S_2 \in S$ then it maintains the following lexicographic order: $S_1 \leq_L S_2$. There is no lexicographic ordering relationship between two subsets $S_1$and S_2, if $S_1$and S_2 are disjoint subsets.

Figure 4 shows an example of a lexicographic tree which considers lexicographic ordering for four items. The root of the tree is an empty set and each k-level contains k-items. In each level, k-itemsets maintain lexicographic ordering with the tail nodes containing items lexicographically larger than elements of the head node. The support value of the head node is more than that of the tail node. It can be seen that the nodes closer to the root are more frequent than those far from the root. There is a non-linear line (called a cut) in the tree which separates frequent itemsets from infrequent ones. The nodes which are above the cut are frequent itemsets and the elements below this cut are infrequent ones [10].

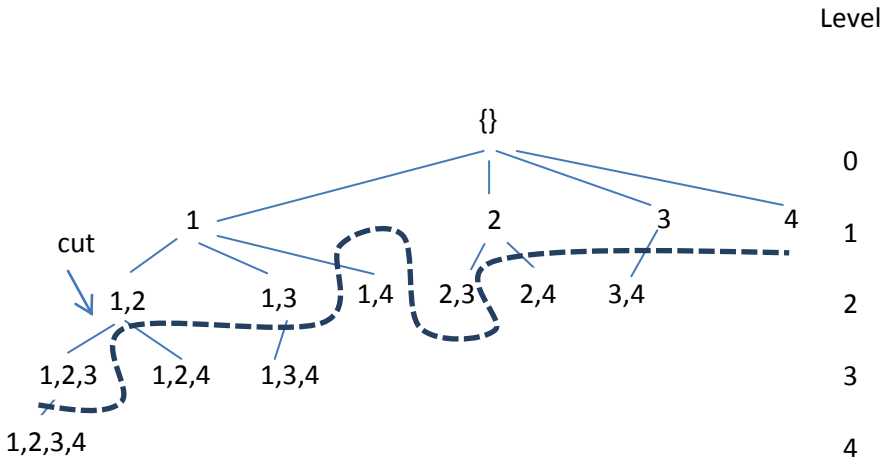

Figure 4: Lexicographic tree of four items

For this study, we consider a lexicographic tree [18] which is based on user define support values. The line is defined by a user define support value and the area above the line is referred to as a positive area and the area below the line is referred to as a negative area. All the nodes in a positive area are frequent whereas all the nodes in a negative area are infrequent. If we redesign the tree of Figure 4, the lexicographic tree of these four items would be as follows:

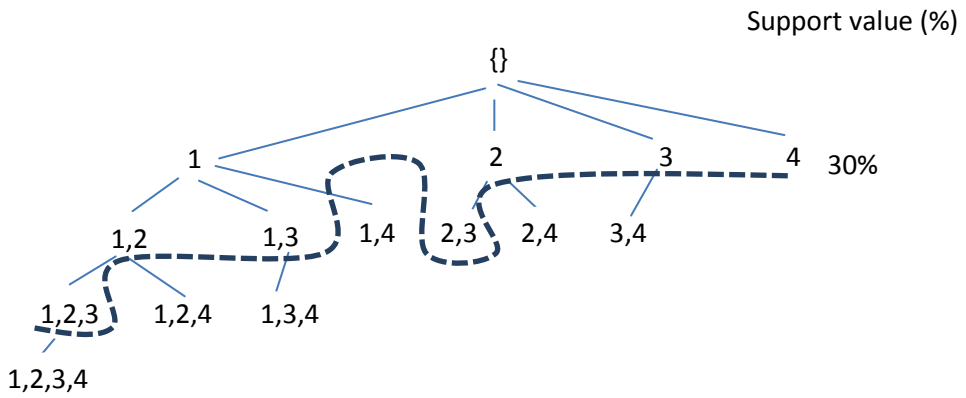

Figure 5: Lexicographic tree of four items based on a user define support value

In Figure 5, the nodes within the positive boundary area have a minimum support value which is 30%. This proposed method searches frequent nodes within a positive area and try to converge to a solution: finding all the frequent itemsets as early as possible. Figure 5 verifies the following statement, if there are 4 items, and it enumerates $2^4 - 1 = 15$ nodes including the root node. With Apriori algorithm, one would test all the nodes in a specific level and generate a candidate set. This candidate set generation needs a long time for finding maximal frequent itemsets. For example, in Figure 5 it tests the itemsets {1},{2},{3},{4} in level 0 and find that all the itemsets are frequent since these nodes meet the minimum support value. Then it goes to next level to scan the database to get the support values of {1,2},{1,3}, {1,4}, {2,3},{2,4},{3,4} and so on. On the next level it prunes the itemsets {1,4},{2,4},{3,4} since these nodes have support values which are less than the user define support value. On the other hand, unlike apriori algorithm, this algorithm does not need to test all the nodes, which saves huge amount of time even when the database is very large. For the current example, if the initially generated itemset is {1,2,3} then it scans the database and calculates the support value. If the support value of the generated itemset {1,2,3} is ≥30%, then it stores this itemset in a frequent itemset array called FI_Superset_Member. In the future it will not scan the database for {1},{2},{3},{1,2}, and {1,3} since these itemsets are the subsets of the previously generated itemset {1,2,3}. If the generated itemstes are {1},{2},{3} or {1,2} then it always checks the array FI_Superset_Member. If it finds any superset in FI_Superset_Member then this approach will discard theses subsets, which substantially reduces the time for scanning the database to calculate the support values correspondingly.

The main advantage of this approach is its ability to quickly converge to a solution, and find all the supersets in a positive boundary area closer to the cut as fast as possible. In the above example, if {1,2,3} is generated before all of its subsets ({1},{2},{3},{1,2}, {1,3}, {2,3}) and found to be a frequent itemset, then it will discard those subsets (which are also frequent itemsets). If the next generated itemset is {1,3,4} it will check the NFI_Subset_Member array and does not find any subset there. PSO based algorithm will scan the database for this itemset to find its support value and store it in NFI. In the future all the supersets of {1,3,4} will be discarded. Figure 6 shows the graphical view of the behaviour of proposed method.

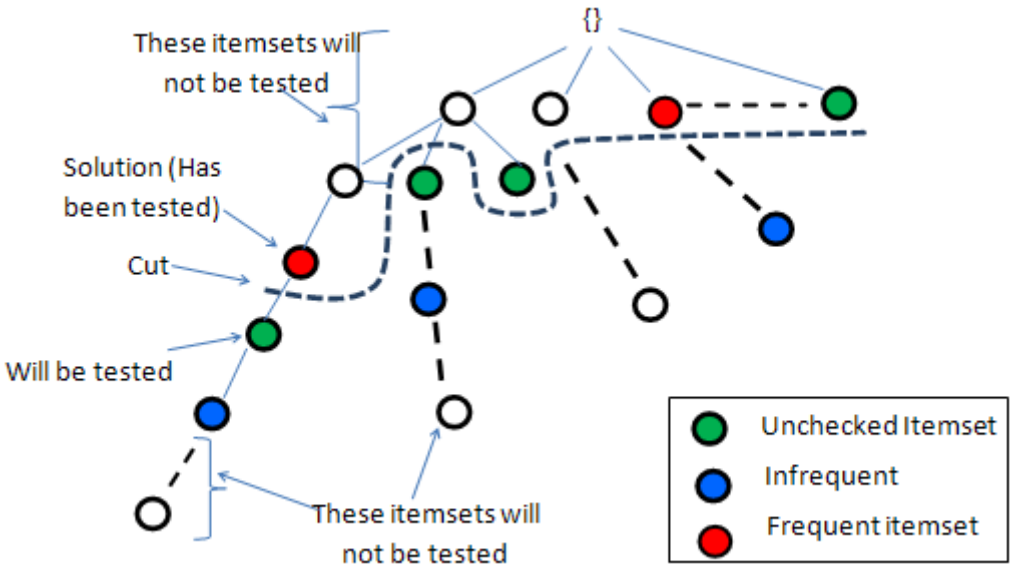

Figure 6: Behaviour of Proposed Method

V. APPLICATION OF PSO ALGORITHM TO ASSOCIATION RULE MINING

Particle swarm optimization algorithm is an evolutionary computational technique where swarm describes the behaviour of particles. It was first introduced by Kennedy and Eberhart in 1995 [19]. To get the optimum solution, this technique considers population based searching mechanism where particles change their positions in a given space with respect to time. The particles are flying in a multidimensional space to find the solution in PSO system. When particles fly in a multidimensional space, each particle considers two experiences to modify its current position. One is the best fitness value it has achieved called "pbest" and another is the best fitness value achieved by any particle of the generated population called "gbest". If $v_i(t)$ is the velocity of *i-th* particle at time *t*, then for calculating the new velocity of *i*-th particle at time *t+1,* which considers two best values pbest and gbest and it is

$$v_i(t+1) = v_i(t) + r_1\big(pbest_i - x_i(t)\big) + r_2\big(gbest - x_i(t)\big) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Where $x_i(t)$ is the position of particle *i* at time *t*, $pbest_i$ is the personal best position found by the *i-th* particle. To balance global and local search, Shi and Eberhart in 1998 introduced another method named "inertia weight". In this method the following equations are used to modify the position of a particle *I*[20].

$$v_i(t+1) = wv_i(t) + r_1\big(pbest_i - x_i(t)\big) + r_2\big(gbest - x_i(t)\big) \quad (3)$$

Here *w* acts as an inertia weight which can be a constant or time function. Constriction factor is added in PSO technique by Clerc et al in 1999[21]. This factor increases the social interaction among the particles which is a major factor for improving the performance of PSO algorithm.

A. Itemsets Mapping to Chromosomes

The porposed method maps itemsets onto a chromosome code. Each node in the lexicographic tree represents different itemsets and all the nodes in the tree get a unique chromosome code. The main feature of choromosome coding is

1) It is easy to calculate the support value since it uses bitmap representation of the database.
2) Generate all the possible nodes.

If there are n items, it enumerates $(2^n - 1)$ itemsets or nodes in the lexicographic tree. Theoretically, This proposed approach does not need to generate, store and traverse all the nodes in its lifetime.

The length of a chromosome is fixed. If a database contains n items, then the length of all the generated chromosomes are always n. The chromosomes look like the following:

V_{item_1}	V_{item_2}	...	V_{item_k}	...	$V_{item_{n-1}}$	V_{item_n}

Figure 7: Mapping items onto chromosomes $V_{item_{1....n}} \in [0,1]$

For example: A database D contains 4 items and 1000 transactions. These 4-items can generate $(2^4 - 1) = 15$ nodes in a lexicographic tree with each node representing one or more transactional patterns among 1000 transactions of database D. Let's say, transaction k contains $item_2$and $item_4$. This transaction represents the node (2,4) and the chromosome coding of this node is:

0	1	0	1

B. Population generation

The evolution starts with randomly generated individuals called particles. Each particle class contains 5 variables named support: support value of the particle; velocity: velocity of that particle; best_support; position: particle's current position; bestposition: particle's best position it has achieved so far, it is equivalent to "pbest". Here position is the itemests in a lexicographic tree. At time t, each particle's support value is compared with each other and the particle is chosen as a best particle, the support value of that particle is close to the user defines support value. The support value of best particle is called here "gbest". All other particles change its position with respect to best particle's position. When the particles are traversed in tree, all the traversed nodes are classified into frequent and infrequent itemsets. Through this way the search space became narrow for the swarm particles.

C. Lifetime of proposed method

The lifetime of proposed method depends on the fixing of all the particles position. The search space become so narrow that the swarm particles will not get the new solution after a certain amount of time and the program will be terminated.

VI. PROCEDURES OF PROPOSED ALGORITHM

A. Algorithm Design for ARM using PSO

Input D: database; numberofParticles; min_supp: minimum support; min_conf: minimum confidence;

Output AR: Association Rules

1) Scan the database D and find the support value of all 1- itemsets and store it in a matrix named Itemset_1_support
2) Prune those 1-itemset which do not satisfy min_supp threshold
3) Let *freq* is a matrix which store all frequent 1-itemsets
4) For each 1-itemset from matrix itemsets_1_support set IR_front and calculate IR_remaining = number_of_items – IR_front, this will finally give a search space for specific 1-itemsets
5) Generate random position of the particles and the positions must be in specific 1-itemset range search space
6) Check the FI_Superset_Member and NFI_Subset_Member array for superset and subset checking of these generated particles.
7) If it finds any superset or subset of a particle in FI_Superset_Member or NFI_Subset_Member respectively, then it assigned this particle position as invalid position and go to Step 4.
8) All the support values of particles are considered and compared it to the user define threshold support value name min_supp. Support value of a particle which is close to the min_supp is assigned as best_particle.
9) IF best_particle.support<min_support

   Since the itemsets closer to the root are more frequent, so the search space of that particle is above the current position in lexicographic tree. Perform NFI_Member_Add, and if any infrequent itemsets are found then update NFI_Subset_ Member.
10) IF best_particle.support>min_support

    Since the itemsets far from the root are less frequent, so the search space of that particle is below the current position in lexicographic tree. PerformFI_Member_Add, and if any frequent itemsets are found then updateFI_Superset_Member.

    // The search space for swarm particles become narrow through step 9 and 10 and all the solutions would be near the cut which was shown in lexicographic tree
11) All other particles follow the position of best particles and change their positionrandomly to avoid local optima.
12) For each frequent k-itemset$I \in FI_Superset_Member$

    If $c \geq \min\_conf$ generate association rules for this itemset

B. Mining the superset in a positive boundary area

For an itemset X, if there is any subset of X in FI_Superset_Member, then this method is called to replace that subset by its superset X. This method is also applicable if X is a new frequent item with no subset in FI_Superset_Member.

```
// Invocation: FI_Member_Add(I_F, FI_Superset_Member)
1. If any subset of I_F is in FI_Superset_Member
2.     Delete the Subset of I_F
3.     Add I_F in FI_Superset_Member
4. Else add I_F in FI_Superset_Member
```

C. Mining the subset in negative boundary area

For an itemset X, if there is any superset of X in NFI_Subset_Member, then this method is called to replace that superset by its subset X. This method is also applicable if X is a new infrequent item and it has no superset in NFI_Subset_Member.

```
// Invocation: NFI_Member_Add(I_IF, NFI_Subset_Member)
1. If any superset of I_IF is in NFI_Subset_Member
2.     Delete the Superset of I_IF
3.     Add I_IF in NFI_Subset_Member
4. Else add I_IF in NFI_Subset_Member
```

D. Pruning Methods

```
// Invocation: Check_Member_for_Item (I, FI_Superset_Member, NFI_Subset_Member)
1. If any superset of I is in FI_Superset_Member
2.     Discard I
3. Else if any subset of I is in NFI_Subset_Member
4.     Discard I
5.     Else scan the database to calculate support value for I
6.     If support value ≥ user define support value
7.       Invoke FI_Member_Add
8.     Else invoke NFI_Member_Add
```

Check_Member_for_Item function incorporates three techniques:

1) Superset Checking Techniques
   Checking to see whether a given chromosome is a superset in a positive boundary area. Further pruning happens if a given itmeset is not a superset in the positive boundary area.
2) Subset Checking Techniques
   Checking to see whether a given chromosome is a subset in a negative boundary area. Further pruning happens if a given itmeset is not a subset in the negative boundary area.
3) Unchecked itemset checking techniques:
   If an itemset is neither a superset in a positive boundary area and nor a subset in a negative boundary area, then this itemset is referred to as an "unchecked" itemset and needs to be tested. For this unchecked itemset, this algorithm scans the database and sets the itemset in FI_Superset_Member or NFI_Subset_Member according to the user define support value.

VII. EXPERIMENTAL RESULTS

The experiments were performed on an Intel(R) core i5-3210M CPU @2.50GHz, 4 GB RAM running on Windows 7 Enterprise. The algorithm was written in C++ language. Microsoft Visual Studio 2012 was used to compile the code. Initially we tested our algorithm ona small database. It contains 5 items for a large number of transactions. We considered a few parameters for this experiment which was shown in Table III.

TABLE III. PARAMETERS FOR ASSOCIATION RULE MINING ALGORITHM USING PSO

Specification	Value
Number of particles	2
User define support value	40%
User define confidence value	50%
Selection of the path	Randomly select the next path by following the movement message of gbest and pbest

For selecting the next path of a particle, particle depends on the movement message of "gbest" and the direction of current "pbest" value. For this experiment after getting "gbest" and "pbest" value, particle changes its position randomly and updates its "pbest" accordingly. This random position change will help avoid the "local optima" problem. Table IV shows the frequent itemsets that are generated under a user define support value. From this result we see that, itemsets {1,2,3,4}, {1,3,4,5} contain maximum items. Item column in Table IV means for which item the position is fixed. For item 1, the particle's search space considers all the superset which will be generated from 1. For item 3, the search space contain {3,4}, {3,5} and {3,4,5} positions. If we look at the results we will see that for item 1, the generated frequent itemsets are {1,2,3,4},{1,3,4,5}. Whereas for item 2, the generated frequent itemset is {2,3,4} which is the subset of the generated frequent itemsets of item 1. This approach considers the pruning strategies that, all the subset of frequent itemset will be pruned. If we do not subdivide the whole search space by the item number then we could miss some interesting rules. For example, the confidence value of itemset {1,2,3,4} is 40%. On the otherhand, the confidence value of itemset {2,3,4} which is the subset of itemset {1,2,3,4} is 70%. That is,itemset {2,3,4} can generate strong rule. For this reason we subdivide the whole search space by the item number. Otherwise it could miss some interesting rules.

TABLE IV. FREQUENT ITEMSETS WITH SUPPORT AND CONFIDENCE VALUE

Database	Records	Items	Support (%)	Item	Frequent Itemsets	Confidence	Remarks
1000×5	1000	5	40	1	{1,2,3,4}, {1,3,4,5}	{40%},{40%}	
				2	{2,3,4}	{70%}	
				3	{3,4,5}	{40%}	
				4	{4,5}	{50%}	
				5	{5}		Single item, does not generate rules

Table V shows the generated strong rules which satisfy the user define support and confidence values. This mining approach for each item number generates frequent and infrequent itemsets. Table VI shows the infrequent itemsets. Users can generate association rules from infrequent ietmsets.

TABLE V. GENERATED STRONG RULES

Database	Frequent Itemsets	Confidence	Remarks
1000×5	{1,2,3,4}, {1,3,4,5}	{40%},{40%}	No rules generated
	{2,3,4}	{70%}	2→3,4 2,3→4
	{3,4,5}	{40%}	No rules generated
	{4,5}	{50%}	4→5
	{5}		Single item, does not generate rules

TABLE VI. INFREQUENT ITEMSETS

Database	Support (%)	Item	Infrequent Itemsets
1000×5	<40%	1	{1,2,3,4,5}
		2	{2,3,4,5}
		3	None
		4	None
		5	None

VIII. CONCLUSIONS AND SUMMARY

In this paper we propose a new approach based on Particle Swarm Optimization Algorithm to mine association rules for both frequent and infrequent itemsets in an efficient way. The experimental results demonstrate several advantages of our algorithm in comparison with other existing algorithms.

1) It generates frequent and infrequent itemsets near the cut in the lexicographic tree.

2) Previous researchersalso applied PSO for association rule mining,however, their studies showed the mining results for only two or three items. This approach can mine association rules for more than three items if it satisfies user defined threshold confidence values. In addition, this approach considers user define threshold values which helps users mine those rules which are interesting to them.

2) It shows the power of using a heuristic algorithm for generating association rules for frequent itemsets along with infrequent ones from a lexicographic tree.

3) The experimental analysis of this approach shows the effect of generations of particles in a search space and pruning all the subsets and supersets in both positive and negative boundary areas, which dramatically reduces search space and cost of counting support value of itemsets.

The goal of the future research is to apply this algorithm for real large databases to see the effect of this approach. From the testing results it can be concluded that, for a small number of items random generation of population can give good results. For large number of items, hybrid PSO which can merge genetic algorithm for population generations could give more interesting results which would be the focusing point for further studies.

References

[1] E. Cohen, M. Datar, S. Fujiwara, and I. C. Society, “without Support Pruning,” IEEE Trans. Knowl. Data Eng., vol. 13, no. 1, pp. 64–78, 2001.

[2] K. Wang, D. W. Cheung, and F. Y. L. Chin, “Mining Confident Rules Without Support Requirement *,” in Proceedings of the tenth international conference on Information and knowledge management, 2001, pp. 89–96.

[3] H. Xiong and P.-N. Tan, “Mining strong affinity association patterns in data sets with skewed support distribution,” Third IEEE Int. Conf. Data Min., pp. 387–394, 2003.

[4] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” in 20th International Conference on Very Large Data Bases, 1994, pp. 487–499.

[5] H. Toivonen, “Sampling Large Databases for Association,” in Proceeding of the 22nd VLDB Conference, 1996, pp. 134–145.

[6] S. Birn, R. Motwani, J. . Ullman, and S. Tsur, “Dynamic Itemset Counting and Implication Rules for Market Basket Data,” in Proceeding of the ACM SIGMOD, 1997, pp. 255–264.

[7] D.-I. Lin and Z. M. Kedem, “Pincer-Search: A New Algorithm for Discovering the Maximal Frequent Set,” in 6th International Conference on Extending Database Technology.

[8] R. J. Bayardo, “Efficiently Mining Long Patterns from Databases,” ACM SIGMOD, pp. 85–93, 1998.

[9] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad, “Depth first generation of long patterns,” Proc. sixth ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’00, vol. 2, pp. 108–118, 2000.

[10] D. Burdick, M. Calimlim, and J. Gehrke, “MAFIA: a maximal frequent itemset algorithm for transactional databases,” Proc. 17th Int. Conf. Data Eng., no. X, pp. 443–452.

[11] K. Gouda and M. J. Zaki, “GenMax : An Efficient Algorithm for Mining,” Data Min. Knowl. Discov., vol. 11, no. 3, pp. 223–242, 2005.

[12] B. Alataş and E. Akin, “An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules,” Soft Comput., vol. 10, no. 3, pp. 230–237, Apr. 2005.

[13] A. Salleb-aouissi, C. Vrain, C. Nortet, X. Kong, and D. Cassard, “QuantMiner for Mining Quantitative Association Rules,” Mach. Learn. Res., vol. 14, no. 1, pp. 3153–3157, 2013.

[14] A. Salleb-aouissi, C. Vrain, and C. Nortet, “QuantMiner : A Genetic Algorithm for Mining Quantitative Association Rules,” in 20th International Joint Conference on Artificial Intelligence, 2007, pp. 1035–1040.

[15] R. J. Kuo and C. W. Shih, “Association rule mining through the ant colony system for National Health Insurance Research Database in Taiwan,” Comput. Math. with Appl., vol. 54, no. 11–12, pp. 1303–1318, Dec. 2007.

[16] R. . Kuo, C. . Chao, and Y. . Chiu, “Application of Particle Swarm Optimization to Association Rule Mining,” Appl. Soft Comput., vol. 11, no. 1, pp. 326–336, 2011.

[17] J. Huang, Y. Che-tsung, and C. Fu, “A Genetic Algorithm Based Searching of Maximal Frequent Itemsets,” in International conference on artificial intelligence, 2004.

[18] M. M. J. Kabir, S. Xu, B. H. Kang, and Z. Zhao, “A Novel Approach to Mining Maximal Frequent Itemsets Based on Genetic Algorithm,” in Accepted in International Conference on Information Technology and Applications (ICITA), 2014.

[19] R. C. Eberhart, “Particle Swarm Optimization : Developments , Applications and Resources,” in Proceeding of the 2001 Congress on Evolutionary Computation, 2001, pp. 81–86.

[20] Y. Shi and R. . Eberhart, “A Modified Particle Swarm Optimizer,” in IEEE International Conference on Evolutionary Computation, 1998, pp. 69–73.

[21] M. Clerc, “The swarm and the queen: towards a deterministic and adaptive particle swarm optimization,” in Proceeding of the Congress of Evolutionary Computation, 1999, pp. 1951–1957.