MaRFI: Maximal Regular Frequent Itemset Mining using a pair of Transaction-ids

G. Vijay Kumar
Scholar, Computer Science and Engineering Department
Acharya Nagarjuna University
Guntur, India.
gvijay_73@yahoo.co.in

Dr. V. Valli Kumari
Professor, Department of CS & SE
AU College of Engineering
Visakhapatnam, India
vallikumari@gmail.com

Abstract—Frequent pattern mining is the fundamental and most dominant research area in data mining. Maximal frequent patterns are one of the compact representations of frequent itemsets. There is more number of algorithms to find maximal frequent patterns that are suitable for mining transactional databases. Users not only interested in occurrence frequency but may be interested on frequent patterns that occur at regular intervals. A frequent pattern is regular-frequent, if it occurs at less than or equal to user given maximum regularity threshold. Occurrence behaviour (regularity) of a pattern may be considered as important criteria along with occurrence frequency. There is no suitable algorithm to mine maximal regular-frequent patterns retrieving at once in transactional databases also satisfies downward closure property. Thus we are introducing a new single-pass algorithm called *MaRFI* (*Ma*ximal *R*egular *F*requent *I*temset) which mines maximal regular-frequent patterns in transactional databases using pair of transaction-ids instead of using item-ids. Our experimental results show that our algorithm is efficient in finding maximal regular-frequent patterns.

Keywords-maximal frequent itemset; regular patterns; single-pass; transactional database; down ward closure property; transaction-ids;

I. INTRODUCTION

Mining various interesting patterns is an important and challenging area in data mining and knowledge discovery research. Discovering frequent patterns is one of the most vital fields in association rule mining. An itemset is frequent if its support is not less than the user given minimum support threshold. Apriori Algorithm [1][2] was the fundamental algorithm to mine frequent patterns from static databases and was introduced by Agarwal et al., in 1993 which requires *k* number of scans to generate *k*-itemset. Researchers had improved the quality of association rule mining by introducing a huge number of algorithms and their mutations which are proposed on the basis of Apriori Algorithm. Association rule mining process is divided into two steps. In the first step it finds the frequent itemsets whose support threshold is greater than or equal to the given support measure. In the second step it generates strong association rules from the frequent itemsets.

Han et al., in 2000 introduced a high compact support-descending FP-tree and FP-growth algorithm [3] to mine frequent itemsets without generating candidate sets which requires only two database scans. A large number of patterns are normally generated when support threshold is set to low, and most of them are found to be insignificant depending on the application or user constraint. As a result several techniques have been proposed recently to reduce the desire result set by some of the user interesting parameters such as closed [4], k-most [5], maximum length [6] etc., are found to be useful in discovering frequent patterns of special interest among users. The other user interesting time interval parameter may be a regular pattern. Users may perhaps be interested on frequent patterns that occur at regular intervals. For example, a web site administrator to improve web page design may be interested in regularly visited web pages rather than on the heavily hit web pages for a specific period of time. Tanbeer et al., [7] proposed regular pattern mining in 2008 by constructing RP-tree and FP-growth approach to mine regular patterns over transactional databases. An item set is regular if its regularity threshold is not greater than the user given maximum regularity threshold. The idea of maximal frequent itemset was proposed in the year 1998, an itemset is maximal frequent if its support is frequent and it should not be a subset by any other frequent itemset. In this paper we are introducing a new single-pass algorithm called *MaRFI* to mine maximal regular-frequent itemset in transactional databases which uses transaction-ids instead of using

item-ids. An itemset is maximal regular-frequent if it satisfy the two measures (i.e., support and regularity) and it should not have any other superset that satisfies the above two measures in the given database.

The rest of the paper is organized as follows. In Section 2 we discuss about the related work and in Section 3 we define the problem of finding maximal regular-frequent patterns in transactional databases. In Section 4 we describe the process of mining maximal regular-frequent patterns with transaction-ids. In Section 5 we present our results and finally we conclude the paper in Section 6.

II. RELATED WORK

Maximal Frequent Itemset (MFI) was first introduced in the year 1998, Bayardo et al., proposed MaxMiner Algorithm [8] for mining the maximal element sets. MaxMiner uses a breadth-first traversal of the search space, and also it reduces database scans by employing a look-ahead to prune the branches of the tree i.e., it involves in superset pruning. MaxMiner also employs item re-ordering heuristic to increase the effectiveness of superset-freqency pruning. MaxMiner uses the original horizontal database so it can perform the same number of passes over a database as Apriori does. Agrawal et al., introduced DepthProject Algorithm [9] uses depth first search of a lexicographic tree and also used counting method based on transaction projections along with its branches to find long itemsets. This projection is equivalent to the horizontal version of the tidsets at a given node in the search tree. The DepthProject algorithm also uses the look-ahead pruning method with item reordering and returns a superset of the maximal frequent itemset and also require post-pruning to eliminate non-maximal patterns.

Han et al., [3] introduced FPgrowth algorithm and a tree based data structure called FP-tree which is a compact representation of all the transactions in the database. This tree uses a recursive divide-and-conquer method and a database projection approach to mine long patterns. On the other hand it is not feasible when pattern length is long since it enumerates all frequent patterns. Burdick et al., introduced Mafia [10] which uses three pruning strategies to remove non-maximal sets to obtain maximal frequent itemset. The first pruning is look-ahead pruning strategy which is used like in MaxMiner. The second is to check if a new set is subsumed by an existing maximal set. The last technique checks if $t(X) \subseteq t(Y)$. Mafia uses vertical bit-vector data format, compression and projection of bitmaps to improve the performance. Mafia mines a superset of MFI and requires a post-pruning step to eliminate non-maximal patterns just like in DepthProject algorithm. GenMax [11][12] was introduced by Gouda and Zaki which is used backtracking search for enumerating all maximal patterns. It uses number of optimizations to quickly prune away a large portion of the subset search space. The data representation in GenMax is vertical format representation. It also uses a progressive focussing technique to remove non-maximal patterns and uses diffset propogation to perform fast frequency counting. MaxMiner and MafiaPP which is an extented version of Mafia are efficient in some datasets like *mushroom* dataset rather than GenMax. So the authors improved the algorithm by using an optimized intersection-based method that was able to run in the same time as MaxMiner and MafiaPP.

The Pincer-Search Algorithm [13] used horizontal format to constructs the candidates in a bottom-up approach like Appriori, and at the same time it also starts a top-down search for maintaining a candidate set of maximal patterns. This is useful in reducing the number of database scans by eliminating non-maximal patterns early, but the overhead of maintaining this is very high. SmartMiner algorithm [14] finds exact maximal frequent itemset from large databases. This algorithm uses global and local tail information to supplement dynamic reordering and it does not require superset checking which is very expensive. SmartMiner at each step passes the tail information to lead a small search tree for new maximal frequent itemset. FPMax algorithm [17] used to mine only for MFI which is an extension of FP-Growth algorithm. FP-tree is also used in this algorithm to store the support count information of the entire database. To keep track for all MFI they used another tree structure called MFI-tree to reduce the search time and the number of subset testing operations. Sunil Joshi et.al., [18][19] used Apriori-like algorithm in which first they generated transaction pair with longest common subsequence of items by using transaction-ids and prune the empty pairs which having none of the element in common using dynamic function. In the next step they find for longest common subsequence from 3 transaction-ids and delete the empty sets. The process continues until the maximal pattern detects after deleting all the empty sets. The authors say that this approach is very fast when itemsets are much larger than the transactions.

All the above algorithms cannot be directly applied for mining the regular-frequent patterns from a transactional database because these algorithms consider only support threshold and do not consider regularity threshold. Our algorithm also uses transaction-ids but only a pair of transaction-ids for mining maximal regular-frequent itemsets requires only one database scan. Our algorithm works efficiently in all aspects i.e., irrespective of length of an itemset or the number of transactions in a database. The process of MaRFI algorithm is explained in the fourth section with couple of examples.

III. PROBLEM DEFINITION

In this section we describe the concepts of *regular-frequent* pattern mining and define the necessary definitions of the problem to obtain *maximal regular frequent Itemsets* in transactional databases.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n distinct items. A set $X = \{i_j, \ldots, i_k\} \subseteq I$, where $j \leq k$ and $j, k \in [1, n]$ is called an itemset or a pattern. A transaction $t = (tid, Y)$ is a couple where each transaction has a unique identifier *tid* is a transaction-id and Y is a pattern. Let size (t) be the size of t, i.e., the number of items in Y. A transaction database *TDB* over I is a set of transactions $T = \{t_1, \ldots, t_m\}$, $m = |TDB|$ is the size of *TDB*, i.e., the total number of transactions in *TDB*. If $X \subseteq Y$, which means that t contains X or X occurs in t and denoted as t_j^X, $j \in [1, m]$. Therefore, $T^X = \{t_j^X, \ldots, t_k^X\}$, $j \leq k$ and $j, k \in [1, m]$ is the set of all transactions where pattern X occurs in *TDB*.

A. Frequent pattern X

The total number of transactions in a *TDB* that contains pattern X is called the support of X i.e., $Sup(X)$. Hence $Sup(X) = |T^X|$, where $|T^X|$ is the size of T^X. The pattern X is said to be frequent if its support is greater than or equal to user given minimum support threshold i.e., $Sup(X) \geq min\_sup(\delta)$.

B. Regularity of frequent pattern X

Let t_j^X and t^X_{j+1}, $j \in [1, (m - 1)]$ be two consecutive transactions where frequent pattern X appears. The variation between these two successive transactions can be defined as a period of X, say p^X (i.e., $p^X = t^X_{j+1} - t^X$, $j \in [1, (m - 1)]$). For ease, to calculate the period of a pattern, we consider the first transaction in the *TDB* as null i.e., $t_{first} = 0$ and the last transaction is the m^{th} transaction i.e., $t_{last} = t_m$. Let for a T^X, P^X be the set of all periods of X i.e., $P^X = \{p_1^X, \ldots, p_r^X\}$, where r is the total number of periods in P^X. Then the regularity of a frequent pattern X can be denoted as $Reg(X) = max\{p_1^X, \ldots, p_r^X\}$. A frequent pattern X is said to be regular frequent if its regularity is less than or equal to user given maximum regularity threshold i.e., $Reg(X) \leq max\_reg(\lambda)$.

C. Maximum Regular-Frequent pattern X in Transactional Databases

An itemset X is a maximal regular-frequent itemset if it is regular-frequent and should not be a subset of any other regular-frequent itemset in the given database.

IV. MINING MAXIMAL REGULAR FREQUENT ITEMSET

In this section we describe the mining process of maximal regular-frequent itemset in transactional databases using transaction-ids which requires only one database scan. Mining maximal itemsets at once reduces lot of time and space. We explain the process of mining maximal regular-frequent itemsets with two example transactional databases. Why we are taking two examples because to see how deep our algorithm searches for finding maximal regular frequent patterns in transactional databases. In the first example our algorithm finds maximal itemset with in two steps after retrieving common itemsets from the transaction pairs where as in the second example it extends few more steps to find maximal regular-frequent itemsets from the database. Let us consider Table 1 as our example-1 transactional database, taken from [15].

In the first example our process first formulates all the possible combinations of pairs of transaction-ids (of size two) and extracts the common items of those pairs of transactions from Table1 of transactional database. i.e., transaction-id pairs {(1, 2), (1, 3), . . . ,(1, n), (2, 3), (2, 4), . . . , (2, n), (3, 4), . . . , (n - 2, n), (n - 1, n)} are used to find the common items present in them. The number of possible combinations of a database consisting of n number of transactions would be $\sum$ n-1 and is equal to n(n-1)/2. Simultaneously the length of itemset of each pair of transactions is also measured. These lengths of itemset i.e., number of items in each combination are considered in hierarchical and in sorting order to find the maximal regular frequent itemset. All the pairs of transactions with common itemset and their lengths are shown in Table2. Let us consider the user given support threshold δ is 3 and user given regularity threshold λ is 3.

TABLE I. TRANSACTIONAL DATABASE1 TDB1

Tid	Itemset
1	a, f, b
2	a, d, e, b, c
3	b, d, a
4	a, e
5	d, e, b, a
6	c, f
7	f, c, a
8	a, c, d, b, e, f
9	a, b

From these obtained pairs of transactions, using itemset length we first consider the largest itemset. If there exists more than one itemset with the largrest value, we consider them one by one in any order as the order do not makes any difference. By comparing the chosen itemset with the other itemset of the same length, algorithm

MaRFI-Algorithm

Input: TDB, δ, λ
Output: Maximal regular-frequent itemset(s)

```
Algorithm MaRFI( )
{  //T = {t: t=(tid, Itemset)}
   // n = |T| total number of transactions in the database
   // TP = {(tp, tp^X): tp =< t_i, t_j> tp^X = common itemset in t_i and t_j}
   // TP^X is the set of transaction pairs containing itemset X
   //XTP^X is the set of all transactions containing itemset X
   // K = {k^X : k^X = |TP^X| } k^X is the length of itemset X
   // procedure to obtain transaction pairs and the common itemset
   TP = X = {ϕ};
   for i = 1 to n - 1
    {  for j = i + 1 to n
          {
              TP = TP ∪ < t_i, t_j>;
              TP^X = t_i^X ⋂ t_j^X ;  // common itemset in t_i and t_j ;
              k^X = |TP^X|;
            }
      }
   // procedure to find maximal regular frequent itemset
    k_max = MAX{K};
    repeat
    {
     if  TP^X != {ϕ}
       {  TTP^X = {ϕ};
            for each tp^X  of length k_max   // transaction pairs containing X
          {
              TTP^X = TTP^X ∪ tp^X;   // transactions containing X (duplicates are eliminated)
            }
            X_sup = | TTP^X |;
            X_reg = regularity(TTP^X);
            if ((X_sup < δ_min_sup) or ( X_reg > λ_max_reg ) )
               {
                  delete TP^X ; // if X is not frequent or not regular then delete X
                }
        }
            X_max = X_max - 1;
   }    until ( X_max > 0);
}// end of MaRFI
```

TABLE II. COMMON ITEMSETS FROM THE PAIR OF TRANSACTION-IDS WITH ITEMSET LENGTH

Pair of Tids	Common Itemset	Itemset Length	Pair of Tids	Common Itemset	Itemset Length
1, 2	a, b	2	3, 7	a	1
1, 3	a, b	2	3, 8	a, b, d	3
1, 4	a	1	3, 9	a, b	2
1, 5	a, b	2	4, 5	a, e	2
1, 6	f	1	4, 6	nil	0
1, 7	a, f	2	4, 7	a	1
1, 8	a, b, f	3	4, 8	a, e	2
1, 9	a, b	2	4, 9	a	1
2, 3	a, b, d	3	5, 6	nil	0
2, 4	a, e	2	5, 7	a	1
2, 5	a, b, d, e	4	5, 8	a, b, d, e	4
2, 6	c	1	5, 9	a, b	2
2, 7	a, c	2	6, 7	c, f	2
2, 8	a, b, c, d, e	5	6, 8	c, f	2
2, 9	a, b	2	6, 9	nil	0
3, 4	a	1	7, 8	a, c, f	3
3, 5	a, b, d	3	7, 9	a	1
3, 6	nil	0	8, 9	a, b	2

calculates the support and regularity of the itemset using the transaction pairs. In the above example the itemset (a, b, c, d, e) is the only itemset with length 5 appears in transaction pair (2, 8). There are no other itemset with length 5 or more. Hence we start our process with this itemset. Transaction pair (2, 8) indicates that itemset (a, b, c, d, e) appears in transaction 2 and 8. The support is calculated as 2 (as it appears only in two transactions). But the user given support threshold is 3. Hence this itemset is infrequent. The periods of this itemset are calculated using the transaction pair (2, 8) as (2 – 0) = 2, (8 – 2) = 6, (9 – 8) = 1. The regularity of this itemset is calculated as the maximum of these values (2, 6, 1) i.e., 6. Therefore the itemset (a, b, c, d, e) is neither frequent nor regular. We now continue with the next largest itemset (a, b, d, e) of length 4 from our example. In Table 2, this itemset appears in two transaction pairs (2, 5) and (5, 8). Hence we can easily draw a conclusion that this itemset appears in three transactions (2, 5, 8) by eliminating the duplication.

Support count of itemset (a, b, d, e) is 3, and the support threshold is 3. Hence this itemset is frequent. To find the regularity of this itemset, let the first transaction be null itemset i.e., $t_f = 0$ and the last transaction is $t_l = 9$. The itemset {a, b, d, e} appears in 2, 5 and 8 transactions. Periods are calculated by subtracting two consecutive transaction-ids i.e., $(2 – t_f) = 2$, $(5 – 2) = 3$, $(8 – 5) = 3$, $(t_l – 8) = 1$. Regularity is then calculated as 3, the maximum of (2, 3, 3, 1). The user given regularity threshold is also 3 and the condition is satisfied. Therefore itemset (a, b, d, e) is regular and also frequent of size 4 which is the maximum size of this example. Hence (a, b, d, e) is a maximal regular-frequent itemset. Since regular patterns satisfies downward closure property, all of the subsets of a, b, d, e are also regular-frequent. The subsets are {a, b, d, e, ab, ad, ae, bd, be, de, abd, abe, ade, bde} are also regular-frequent.

Let the second example transactional database be Table 3 from [16]. In this table also we are having 9 transactions. Let us consider support threshold δ is 2 and regularity threshold λ is 3. Table 4 shows all the common itemsets from transaction pairs along with their lengths. Here also we start with the largest itemset(s)

TABLE III. TRANSACTIONAL DATABASE2 TDB2

Tid	Itemset
1	a, d
2	a, b, c, e
3	a, b, e, f
4	a, b, c, e
5	a, b, e, f
6	b, c, d
7	c, e, d
8	d, e, f
9	d, c, b

TABLE IV. COMMON ITEMSETS FROM THE PAIR OF TRANSACTION-IDS WITH ITEMSET LENGTH

Pair of Tids	Common Itemset	Itemset Length	Pair of Tids	Common Itemset	Itemset Length
1, 2	a	1	3, 7	e	1
1, 3	a	1	3, 8	e, f	2
1, 4	a	1	3, 9	b	1
1, 5	a	1	4, 5	a, b, e	3
1, 6	d	1	4, 6	b, c	2
1, 7	d	1	4, 7	c, e	2
1, 8	d	1	4, 8	e	1
1, 9	d	1	4, 9	b, c	2
2, 3	a, b, e	3	5, 6	b	1
2, 4	a, b, c, e	4	5, 7	e	1
2, 5	a, b, e	3	5, 8	e, f	2
2, 6	b, c	2	5, 9	b	1
2, 7	c, e	2	6, 7	c, d	2
2, 8	e	1	6, 8	d	1
2, 9	b, c	2	6, 9	b, c, d	3
3, 4	a, b, e	3	7, 8	e	1
3, 5	a, b, e, f	4	7, 9	c, d	2
3, 6	b	1	8, 9	d	1

from Table 4. In this example there are two length-4 itemsets from transaction pairs (2, 4) is (a, b, c, e) and (3, 5) is (a, b, e, f). Support of itemset (a, b, c, e) is 2 and support threshold is 2, so this itemset is frequent. Regularity of this itemset is 5 and regularity threshold is 3, so this itemset is not regular. Therefore itemset (a, b, c, e) is frequent but not regular. Next length-4 itemset is (a, b, e, f). Its support is 2 and regularity is 4. Here also this itemset is frequent but not regular. Now we continue with the second largest itemset(s) i.e., length-3 itemset(s). We find five length-3 itemsets in this example. They are (a, b, e) is in four transaction pairs ((2, 3), (2, 5), (3, 4), (4, 5)) and (b, c, d) is in one transaction pair (6, 9). We first calculate itemset (a, b, e) and next we calculate itemset (b, c, d) are regular-frequent or not. Itemset (a, b, e) is in four transactions i.e., (2, 3, 4, 5). Its support is 4 and support threshold is 2, its regularity is 4 and regularity threshold is 3. This itemset is frequent because the support is greater than the given support threshold and it is not regular because the regularity is not less than or equal to given regularity threshold. Even though it appears in more number of transactions but it is not equally distributed among transactions so it is not regular-frequent itemset. Now we see itemset (b, c, d) that is in two transactions (6, 9). Its support is 2 and regularity is 6.

Therefore this itemset is also not a regular-frequent itemset. We continue with the next largest itemset(s) i.e., length-2 itemset(s). Altogether ten length-2 itemsets are there in this example. Itemset (b, c) is in ((2, 6), (2, 9), (4, 6), (4, 9)) transaction pairs, itemset (c, e) is in transaction pairs ((2, 7), (4, 7)), itemset (e, f) is in ((3, 8), (5, 8)) and itemset (c, d) is in ((6, 7), (7, 9)). Now we see one by one whether these itemsets are regular-frequent or not. We calculate the support and regularity of itemset (b, c) which appears in 2, 4, 6 and 9 transactions. Its support is 4 because it is in four different transactions and is greater than the given support threshold, so it is frequent and its regularity is maximum of all periods of transactions (2, 2, 2, 3) i.e., 3 and is less than or equal to given regularity threshold, so it is regular. Therefore itemset (b, c) is a regular-frequent itemset. Next itemset (c, e) support is 3 and regularity is 3. Thus itemset (c, e) is also a regular-frequent itemset. Next itemset (e, f) is in (3, 8) and (5, 8). So it is in 3, 5 and 8 transactions. Its support is 3 and regularity is 3, so it is also a regular-frequent. Itemset (c, d) is in (6, 7) and (7, 9) so it is in three transactions 6, 7 and 9. Its support is 3 and regularity is 6 (failed). So (c, d) is not regular-frequent. Therefore itemsets (b, c), (c, e) and (e, f) are maximal regular-frequent itemsets.

V. EXPERIMENT RESULTS

In this section we are producing our results. Since there is no existing algorithm to discover maximal regular-frequent patterns, we only examine our *Marfi*-algorithm over synthetic and real datasets which are often used for frequent pattern mining experiments developed at IBM Almaden Quest research group from http://cvs.buu.ac.th/mining/Datasets/synthesis_data/ and UCI Machine Learning Repository (University of California–Irvine, CA). All our programs are written in Java NetBeans IDE 7.3. We did our experiments on 2.66 GHz CPU with 2GB of main memory running on Windows XP.

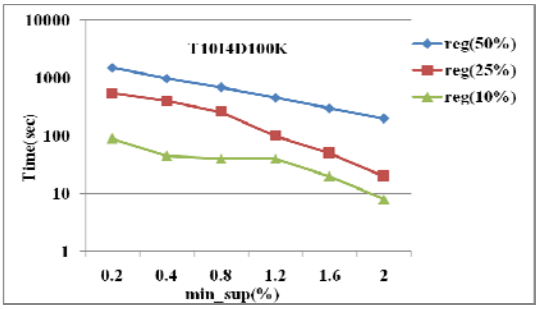

Figure 1. Building and Execution time over T10I4D100K

TABLE V. RESULTS OVER 200 TRANSACTIONS ON T10I4D100K

S. No.	min_sup > =	max_reg < =	Maximal regular-frequent Itemsets
1	1%	50%	{ 183, 533, 541, 594, 642, 72, 96 } – len-7 itemset
2	1.5%	50%	{ 274, 33, 573, 740, 883 } { 392, 461, 569, 801, 862 } two, length-5 itemsets
3	2%	50%	{ 392, 461, 569, 801, 862 } one, length-5 itemset
4	1%	25%	{ 183, 569 } one, length-2 itemset
5	2%	25%	{ 183, 569 } one, length-2 itemset

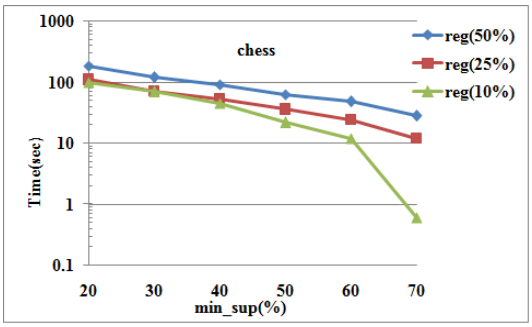

Figure 2. Building and Execution Time over Chess dataset

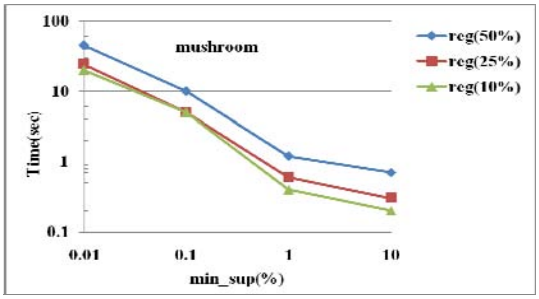

Figure 3. Building and Execution time over Mushroom dataset

We report the results on *T10I4D100K* synthetic dataset which contains 100K transactions, 870 items and 10.10 as the average transaction length. Figure1 shows the building up transaction-pair tables time and execution time on different *Sup*() and *Reg*() values. While doing experiments on *T10I4D100K*, we got long patterns when support measure is very less and regularity is maximum i.e., 50%. As support value increases and regularity value decreases, the length of the output patterns also reducing. In some cases we are getting more number of two itemsets or more number of one itemsets whenever the support is very high and regularity is low. Figure 1 shows the execution time over different min_sup and different max_reg values. The table showing above are the results over 200 transactions on T10I4D100K dataset. Chess data set contains 28056 transactions, 34 items and 7 is the maximum length as well as average length. As the average length decreases, the building transaction pairs and retrieving common itemsets from transaction pairs from the dataset also decreases. Figure 2 shows the building and execution time over different min_sup and max-reg values. Mushroom dataset contains 8124 transactions, 120 items and 23 as the average length. Figure 3 shows the building and execution time over different min_sup and max-reg values on mushroom dataset.

VI. CONCLUSION

In this paper, we propose a new algorithm MaRFI, mines the complete set of maximum regular frequent patterns at once in transactional databases using common items from transaction pairs that requires only one database scan. It is efficient over other algorithms that mine only for maximum frequent itemsets because it is very simple and easy to identify the common itesmets from transaction pairs and to calculate support and regularity threshold values. Our algorithm uses the advantage of transaction pairs instead of itemsets that needs simple calculations. Our experimental results show the out performance of our MaRFI-algorithm.

REFERENCES

[1] Agarwal, R., Imielinski, T., Swamy, A.N.: Mining Association Rules between sets of Items in Large Databases, ACM, SIGMOD Conference of Management of Data, pp. 207-216 (1993).

[2] Agarwal, R., Srikanth, R. Fast algorithms for mining association rules, In Proc. 1994 International Conference on very large databases (VLDBA'94), Santiago, Chile, pp. 487-499, Sept. 1994.

[3] Han, J., Pei, J., Yin, Y. Mining frequent patterns without candidate generation, In Proc. ACM, SIGMOD, International Conference on Management of Data, pp. 1-12, 2000.

[4] Zaki, M.J., Hsiao, C.-J. Efficient Algorithms for Mining Closed Itemsets and their Lattice Structure. IEEE Trans. on Know. And Data Engg. 17(4), 2005, pp. 462-478

[5] Minh, Q.T., Oyanagi, S., Yamazaki, K.: Mining the K-most Interesting Frequent Patterns Sequentially. LNCS,vol. 4224, Springer, Heidelberg, 2006, pp. 620-628

[6] Hu, T., Sung, S. Y., Xiong, H., Fu, Q.: Discovery of Maximum Length Frequent Itemsets. Information Sciences 178, 2008, pp. 69-87

[7] Tanbeer S.K., Ahmed, C.F., Jeong, B.S., Lee, Y-K. Mining Regular Patterns in Transactional Databases, IEICE, Trans. On information and systems, E91-D, 11, pp.2568-2577, 2008

[8] Bayardo, R.J., Efficiently mining long patterns from databases, In proceeding of the ACM SIGMOD international conference on management of data, pp. 85-93, 1998

[9] Agarwal, R. C., Aggarwal, C. C., & Prasad, V. V. V. " Depth first generation of long patterns," In Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining , 2000,pp. 108–118

[10] Burdick, D., Calimlim, M., & Gehrke, J, "Mafia: A maximal frequent itemset algorithm for transactional databases" In Proceedings of 17th international conference on data engineering ,2001,pp. 443–452

[11] Gouda, K., & Zaki, M. J, "Efficiently mining maximal frequent itemsets", In Proceedings of 1st IEEE international conference on data mining, 2001, pp. 163–170

[12] Gouda, K., & Zaki, M. J, "GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets", Data Mining and Knowledge Discovery, Springer Science, 11, 2005, pp. 1-20

[13] Lin, D., & Kedem, Z. M. "Pincer-Search: an efficient algorithm for discovering the maximum frequent set." IEEE Transactions on Knowledge and Data Engineering, 2002, pp.14 (3), 553–566.

[14] Zhou, Q. H., Wesley, C., & Lu, B. J. "SmartMiner: A depth 1st algorithm guided by tail information for mining maximal frequent itemsets" In Proceedings of IEEE international conference on data mining, 2002, pp. 570–577.

[15] G. Vijay Kumar, Dr. V. Valli Kumari, "Sliding Window Technique to Mine Regular Frequent Patterns in Data Streams using Vertical Format", 2012 IEEE International Conference on Computational Intelligence and Computing Research, 2012, pp. 590 - 593.

[16] G. Vijay Kumar, M. Sreedevi, NVS Pavan Kumar, "Mining Regular Patterns in Transactional Databases using Vertical Format", International Journal of Advanced Research in Computer Science, Volume (2), Issue (5) 2011, pp. 581 - 583.

[17] Song, Y. Q., Zhu, Y. Q., & Sun, Z. H., "An algorithm and its updating algorithm based on FP-tree for mining maximum frequent itemsets", Journal of Software, 14(9), 2003 pp-1586–1592.

[18] Sunil Joshi, Dr. R. C. Jain: "A Dynamic Approach for Frequent Pattern Mining Using Transposition of Database" IEEE International Conference on Communication software and Networks (ICCSN 2010) 2010.

[19] Sunil Joshi, R. S Jadon, R. C. Jain, A Frame Work for Frequent Pattern Mining using Dynamic Function, International Journal of Computer Science Issues, 8(3), 2011, pp. 141-147.