Association Rule Mining through Matrix Manipulation using Transaction Patternbase

Shahid Kamal*[,1,2], Roliana Ibrahim[1] and Zia-ud-Din[2]

[1]*Faculty of Computer Science and Information System, Universiti Teknologi, Malaysia, 81310 Skudai, Johor Bahru, Malaysia*

[2]*ICIT, Gomal University Dera Ismail Khan, Pakistan*

Abstract: In data mining studies, mining of frequent patterns in transaction databases has been a popular area of research. Many approaches are being used to solve the problem of discovering association rules among items in large databases. We also consider the same problem. We present a new approach for solving this problem that is fundamentally different from the known techniques. In this study, we propose a transactional patternbase where transactions with same pattern are added as their frequency is increased. Thus subsequent scanning requires only scanning this compact dataset which increases efficiency of the respective methods. We have implemented this technique by using two-dimensional matrix instead of using FP-Growth method, as used by most of the algorithms. Empirical evaluation shows that this technique outperforms the database approach, implemented with FP-Growth, in many situations and performs exceptionally well when the repetition of transaction patterns is higher. We have implemented it using Visual Basic which has substantially reduced coding and computational cost. Success of this method will open new directions.

Keywords: Association Rules, Frequent Patterns, Patternbase, Transaction Base, Matrix, Algorithm.

1. INTRODUCTION

Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [1]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories [2].

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence.

Different algorithms are used to generate association rules. In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. Further, the association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several strategies have been proposed to reduce the number of association rules, such as generating only "interesting" rules, generating only "no redundant" rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength [2].

Hegland [8] reviews the most well known algorithm for producing association rules - Apriori and discuss variants for distributed data, inclusion of constraints and data taxonomies.

The AIS algorithm was the first algorithm proposed for mining association rule [1]. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item.

FP-Tree [7], frequent pattern mining, is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones.

The following is a formal statement of the problem [1]: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Associated with each

*Address corresponding to this author at the ICIT, Gomal University Dera Ismail, Khan Pakistan; E-mail: skamaltipu@gmail.com transaction is a unique identifier, called its TID. We say that a transaction T contains X, a set of some items in I, if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \varnothing$;. The rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with confidence c if c% of transactions in $\mathcal{D}$ that contain X also contain Y. The rule $X \Rightarrow Y$ has support s in the transaction set $\mathcal{D}$, if s% of transactions in $\mathcal{D}$ contain $X \cup Y$.

Given a set of transactions $\mathcal{D}$, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (called minsup) and minimum confidence (called minconf) respectively. Our discussion is neutral with respect to the representation of $\mathcal{D}$. For example, $\mathcal{D}$ could be a data file, a relational table, or the result of a relational expression.

In this paper we have tried to reduce the computational cost by:

- Reducing the number of passes over the database. For this purpose we used Transaction Patternbase instead of transaction database for generating association rules.
- Introducing Two-Dimensional matrix for generating required association rules instead of FP-Tree.

2. THE PROPOSED METHOD

The computational cost of association rules mining can be reduced by reducing the number of passes over the database. In order to achieve this objective, we develop the transactional patternbase which is substantially smaller than the transactional database without loss of any information.

In order to obtain required transactional patternbase we sort every transaction of $\mathcal{D}$ to prune the transactional base. This pruned transactional base is then converted to Transactional Patternbase, presented in section 2.1. Construction of Patternbase is described in section 2.2. Section 2.3 describes the construction of base matrix. In section 2.4, we describe how required association rules are generated.

2.1. Transactional Patternbase

Information from transactional databases is essential for generating association rules. If we can construct transactional patternbase from the transaction database in the first scan, it may reduce frequent scanning of database due its compact size as compared to transactional database which contain redundant transactional pattern.

2.1.1. Transactional Pattern

A transactional database TDB is a set of transactions. A transaction T = (tid, X) is a tuple where tid is a transaction-id and X is an itemset. The itemset is called transactional pattern which has tid in the transactional database. The itemsets Y and Z (such that $Y \subseteq X \subseteq Z$) are not transactional pattern if they do not have their own tid in TDB. A pattern may belong to multiple transactions with different tid's. Example is given in the Table **1**. We selected 20 tuples in the sample transactional Database from [1].

Table 1. Sample Transactional Database

Tid	Items
1	1 5 9 14 19 23 27 32 37
2	1 5 9 14 19 23 27 32 38
3	2 8 12 14 19 24 27 35 38
4	3 5 9 14 19 23 27 36 38
5	1 8 9 14 19 23 27 32 38
6	2 8 12 14 19 24 27 35 38
7	1 5 9 14 19 23 27 32 37
8	1 5 9 14 19 23 27 32 38
9	2 8 12 14 19 24 27 35 38
10	1 5 9 14 19 23 27 34 37
11	1 5 9 14 19 23 27 32 38
12	1 5 9 14 19 23 27 32 38
13	2 8 12 14 19 24 27 35 38
14	1 5 9 14 19 23 27 34 37
15	1 8 9 14 19 23 27 32 38
16	2 8 12 14 19 24 27 35 38
17	1 5 9 14 19 23 27 32 37
18	1 5 9 14 19 23 27 32 38
19	2 8 12 14 19 24 27 35 38
20	1 5 9 14 19 23 27 34 37

We find the transactional patterns from the transactional database during the first scan of the database, while generating the frequent items list. Transactional patterns found from the example dataset given in Table **1** are displayed in Table **2**.

Table 2: Sample Transactional Patterns

Transactional Patterns
1 5 9 14 19 23 27 32 37
1 5 9 14 19 23 27 32 38
2 8 12 14 19 24 27 35 38
3 5 9 14 19 23 27 36 38
1 8 9 14 19 23 27 32 38
2 8 12 14 19 24 27 35 38
1 5 9 14 19 23 27 34 37

2.1.2. Observation of Transactional Patternbase

We have designed a compact data structure called transactional patternbase which is based on the following observations.

1. Transactional patterns are repeating in the transactional database.
2. If multiple transactions share the transactional patterns it is possible to just count the frequency of each transactional pattern in transactional database.
3. A subset or superset of transactional pattern may or may not be a transactional pattern if it has its own transactional id in transactional database.
4. If we store the frequency information of transactional pattern in the transactional patternbase it may be possible to avoid the massive transactional database scanning during the FP Tree construction.

2.1.3. Definition of Transactional Patternbase

A transactional database TDB is the set of transactions consisting of set of items I. A transaction T = (tid,X) is a tuple where tid is a transaction-id A TPB is a set of patterns P. A Pattern P = (pid, X, f) is a tuple where pid is a pattern id, X is a transactional Pattern and f is the frequency of the transactional pattern, where f is the frequency of the respective pattern in the transactional database.

f = Frequency(X, TDB) := count{ tid | (tid, X) $\in$ TDB , X $\subseteq$ I }

It is observed that the number of the transactions in the transactional database is equal to the sum of frequencies of the transactional patternbase. All transactional patterns in the transactional patternbase will be unique.

In order to construct Transaction Patternbase, best suited for our base matrix, we pruned the Table **1**. This pruning is performed by sorting items in ascending order with respect to descending order of their frequencies.

During first scan the frequencies of items are listed in Table **3**.

The table after pruning the input Table **1**, is presented in Table **4**. With 20% support items 3, 36 and 34 are discarded. In this table items with same frequencies are sorted in ascending order in each transaction with respect to descending order of frequencies.

Table 3: Item Frequencies without Threshold

Item	Frequency
1	13
5	12
9	14
14	20
19	20
23	14
27	20
32	10
37	6
38	14
2	6
8	8
12	6
24	6
35	6
3	1
36	1
34	3

Table 4: Sorted Table after Excluding Less Frequent Items

Tid	Items
1	14 19 27 9 23 1 5 32 37
2	14 19 27 9 23 38 1 5 32
3	14 19 27 38 8 2 12 24 35
4	14 19 27 9 23 38 5
5	14 19 27 9 23 38 1 32 8
6	14 19 27 38 8 2 12 24 35
7	14 19 27 9 23 1 5 32 37
8	14 19 27 9 23 38 1 5 32
9	14 19 27 38 8 2 12 24 35
10	14 19 27 9 23 1 5 37
11	14 19 27 9 23 38 1 5 32
12	14 19 27 9 23 38 1 5 32
13	14 19 27 38 8 2 12 24 35
14	14 19 27 9 23 1 5 37
15	14 19 27 9 23 38 1 32 8
16	14 19 27 38 8 2 12 24 35
17	14 19 27 9 23 1 5 32 37
18	14 19 27 9 23 38 1 5 32
19	14 19 27 38 8 2 12 24 35
20	14 19 27 9 23 1 5 37

Table **4** is used for the construction of transactional. patternbase we have constructed the transactional patternbase from the pruned transactional database by excluding repetition of redundant pattern but including frequencies of those repeated transactions. This Transactional Patternbase is shown in the Table **5**.

Table 5: Transactional Patternbase

Pid	Transactional Patterns	f
1	14 19 27 9 23 1 5 32 37	3
2	14 19 27 9 23 38 1 5 32	5
3	14 19 27 38 8 2 12 24 35	6
4	14 19 27 9 23 38 5	1
5	14 19 27 9 23 38 1 32 8	2
6	14 19 27 9 23 1 5 37	3

2.2. Construction of Transactional Patternbase

Information from transaction databases is necessary for mining frequent patterns. If we can calculate the frequencies of pruned transactions from pruned database and use them instead of individual transactions in Matrix population, it will reduce substantial amount of calculation time. Motivated by this thinking, we develop a compact data structure, called transactional patternbase.

We build transactional patternbase from pruned transactional database to reduce the cost of scanning the massive transactional database. We use this data structure, which is smaller in size from the transactional database, but we have the same impact as transactional database.

2.2.1. Data Structure Being Used for Transactional Patternbase

We transform the transactional database to a compact data structure called transactional patternbase. Transactional patternbase consists of pid, itemset X and frequency f of the patterns. Where pid is the unique identifier each pattern X and frequency is the number of occurrences of the itemset X in the transactional database, where from we construct transactional patternbase.

2.2.2. Algorithm for the Construction of Transactional Patternbase

Algorithm for the construction of the transactional patternbase that takes transactional database and minimum-support as input and generate transactional patternbase and frequent items list is given in algorithm as follow.

INPUT: Transactional Database, Min-Sup.

OUTPUT: Transactional Patternbase, Frequent Items List

METHOD: Transactional Patternbase is constructed as follow.

1) Set the Transactional Patternbase empty.

2) Scan the transactional database until last transaction reached

   a) For each item that is occurring in this transaction.

      i. If the item does not exist in the frequent items list, insert this item in it and set its count equal to 1.

      ii. If the item exists in the frequent items list, increase its count.

   b) Check nth transactional Pattern of the transaction database in the transactional Patternbase. If it does not exist, then insert it into Transactional Patternbase and set its frequency to one.

   c) If transactional Pattern does exists in the transactional Patternbase then do not insert it

Table 6:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PID	**14**	**19**	**27**	**9**	**23**	**38**	**1**	**5**	**32**	**8**	**2**	**12**	**24**	**35**	**37**
1	3	3	3	3	3		3	3	3						3
2	5	5	5	5	5	5	5	5	5						
3	6	6	6			6				6	6	6	6	6	
4	1	1	1	1	1	1		1							
5	2	2	2	2	2	2	2		2	2					
6	3	3	3	3	3		3	3							3
Net	20	20	20	14	14	14	13	12	10	8	6	6	6	6	6

into transactional Patternbase but only increase its frequency by 1 in the transactional Patternbase.

d) Sort frequent items list in support descending order.

2.3. Construction of Base Matrix

The base matrix is MxN matrix where M represents pruned transactions and N represents Number of items. Frequency of each transaction is populated into this matrix against each item. The sum of each column will represent the frequency of that particular item.

2.4. Generation of Association Rules

In order to generate association rules, we have used Listbox control in Visual Basic, instead of storing each rule in a separate array. These rules have bee generated as string containing transactions concatenated with comma. The generated rule's frequency is compared against confidence level, if its confidence level is greater than minimum support, it is added to the output text box, from where it can be stored in a text file.

2.4.1. Algorithm for generating Association Rules

Algorithm for generating Association Rules of the desired confidence level is subdivided into two modules. The first module *Generate Rule* uses a list box where Association rules are generated for next item in base matrix, whereas second module *Transaction Frequency* is used to calculate minimum frequency of the rule in each transaction. The sum of all frequencies is frequency of the rule.

a. Generate Rule

INPUT: TotItems **(**Number of items), Min-Sup (Minimum Support), TotTrans (Total Transactions)

OUTPUT: Set of association rules with desired confidence level

METHOD: Association Rules are generated as follows

```
1.    Clear the List Box
2.    Repeat through step 11 varying I from 1 by 1
      until (I > TotI tems)
3.    Determine the List Count
      LC=ListBox.ListCount-1
4.    Add Item Number in List Box
      List Box. AddItem (I)
5.    Repeat through step 11 varying J from 0 by 1
      until (J > LC)
6.    Generate New Rule String
      New Rule=Str(I) + “,” + List Box. List(J)
7.    Add NewRule to List Box
8.    Initialize Frequency for New Rule
      RF=0
9.    Calculate Frequency by repeating through step
      10 varying K from 1 by 1 until (T > TotTrans)
10.   Update Rule’s Frequency
      RF=RF + Transaction Frequency (New Rule, T)
11.   Check for confidence level
      If (Rule Freq>Min Sup)
      Add Rule alongwith frequency to Output List
```

b. Transaction Frequency

INPUT: BTab (Table **6**), New Rule (Rule String Passed as Parameter), T (Transaction Number)

OUTPUT: Calculated frequency of the Transaction

METHOD: Minimum frequency is calculated as follows

```
1.    Split Rule string into Array
      Item[ ]=Split(New Rule,”,”)
2.    Initialize Frequency
      TFreq=BTab[T,Item[0])
3.    Calculate Minimum Frequency by repeating
      through step 4 varying N from 1 by 1 until
      (N>UBound(Item))
4.    Check for Minimum Frequency
      If (TFreq>BTab(T,Item[N])
      Tfreq= BTab(T,Item[N])
5.    Return(TFreq)
```

2.4.1. Example for Generation of Association Rules

As we are using List box to populate Association Rules, generated for next items, thus a sample of the generated rules is presented in Table **7**. This List box will contain all the rules generated for specified items. The maximum number of rules will be 2^n-1, where n is number of items.

Rules satisfying minimum support will be populated in an output text box. A sample is presented in Table **8**.

For our input database in Table **1**, the total number of rules, generated, is 1135. A summary of these rules is presented in Table **9**.

Table 7: A Sample of the Generated Rules

```
1
2
2,1
3
3,1
3,2
3,2,1
4
4,1
4,2
4,2,1
4,3
4,3,1
4,3,2
4,3,2,1
:
```

Table 8: A Sample of Rules Satisfying Min Support

```
[1] {14}=20
[2] {19}=20
[2,1] {19 14}=20
[3] {27}=20
[3,1] {27 14}=20
[3,2] {27 19}=20
[3,2,1] {27 19 14}=20
[4] {9}=14
[4,1] {9 14}=14
[4,2] {9 19}=14
[4,2,1] {9 19 14}=14
[4,3] {9 27}=14
[4,3,1] {9 27 14}=14
[4,3,2] {9 27 19}=14
[4,3,2,1] {9 27 19 14}=14
:
```

Table 9: No of Rules for EACH ITEM

S. No	Item	No of Rules Generated
1	14	1
2	19	2
3	27	4
4	9	8
5	23	16
6	38	32
7	1	64
8	5	128
9	32	256
10	8	16
11	2	32
12	12	64
13	24	128
14	35	256
15	37	128
Total		1135

3. RESULTS AND DISCUSSION

In this section we present the computational performance comparison of this approach with the FP Tree approach using the transactional database.

3.1. Environments of the Comparison

All the experiments are performed on a 1.8 GHz Pentium PC machine with 512 megabytes main memory, running on Microsoft Windows XP. All the programs for this technique are written in Visual Basic 6.0 Service Pack 6 whereas programs for FP Tree are written in JDK 1.5.0_02.

Code for FP-Tree processing has been downloaded from LUCS-KDD Software Library, Liverpool University, Computer Science Knowledge discovery in Datas [10].

Not only our synthetic dataset presented in Table **1** was tested, but real datasets, downloaded from [10] were also tested. The following datasets were downloaded from [10] and they were renamed as:

Actual Name	Renamed
chessKRvK.D58.N28056.C18.num	First.num
nursery.D32.N12960.C5.num	Second.num
pima.D38.N768.C2.num	Pima.num

3.2. Execution Time

1. Our Synthetic Dataset (Table 1)

No. of Transactions	=	20
Minimum Support	=	20%
Confidence	=	80%

a) Matrix Method

Construction of patternbase	=	0.01 sec
Construction of matrix	=	0.03 sec
Generation of Association Rules	=	0.34 sec
Total Time	=	0.38 sec

b) FP-Tree

Construction of FP-Tree	=	0.04 sec
Generation of Association Rules	=	0.55 sec
Total Time	=	0.59 sec

2. First.num

No. of Transactions	=	28056
Minimum Support	=	20%
Confidence	=	80%

a) Matrix Method

Construction of patternbase	=	28.02 sec
Construction of matrix	=	0.25 sec
Generation of Association Rules	=	622.34 sec
Total Time	=	650.61 sec

b) FP-Tree

Construction of FP-Tree	=	53.02 sec
Generation of Association Rules	=	771.34 sec
Total Time	=	824.36 sec

3. Second.num

No. of Transactions	=	12960
Minimum Support	=	20%
Confidence	=	80%

c) Matrix Method

Construction of patternbase	=	4.38 sec
Construction of matrix	=	0.21 sec
Generation of Association Rules	=	192.34 sec
Total Time	=	196.99 sec

d) FP-Tree

Construction of FP-Tree	=	6.02 sec
Generation of Association Rules	=	248.41 sec
Total Time	=	254.43 sec

4. Pima.num

No. of Transactions	=	768
Minimum Support	=	20%
Confidence	=	80%

e) Matrix Method

Construction of patternbase	=	0.631 sec
Construction of matrix	=	0.11 sec
Generation of Association Rules	=	14.45 sec
Total Time	=	15.17 sec

f) FP-Tree

Construction of FP-Tree	=	0.84 sec
Generation of Association Rules	=	21.31 sec
Total Time	=	22.15 sec

As seen from the result shown in above, the time taken by our proposed method is smaller in all cases.

The compactness of the transactional patternbase will leads to the less number of tree nodes updates and less time consumption for tree construction. We perform the experiments using the above said datasets and found the following results.

3.3. Storage Requirements

The proposed method consumes more memory than the FP-Tree method because general rules are created first and then only those rules are selected which meet confidence level. Therefore the storage requirements are directly proportional to the number of items, as number of general rules, generated will be equal to 2^n, where n represents number of items.

3.3. Scalability Study

We also perform the experiments for the efficiency of the FP Tree construction for the recurrence of the

Table 10: Summarizes Execution Times for Both of these Methods

Table	No. Of Transactions	Matrix Method Time(Sec)	FP-Tree Time (Sec)
Synthetic	20	0.38	0.59
First.num	28056	650.61	824.36
Second.num	12960	196.99	254.43
Pima.num	768	15.17	22.15

transactional pattern in the transactional database. We observed that higher frequency of transactional pattern in the transactional database results in higher efficiency as compared to FP Tree method. We have simply copied the same transaction 2,3, and 4 times in these tables. As FP-Tree scans tree for each transaction to update the nodes, therefore the execution time requirement are very high in FP-Tree method. In our proposed method same patternbase will be constructed with different frequencies and association rules generation will take exactly same time in all cases, therefore there is very small increase in execution time.

Figures **1** to **4** shows comparison of both methods for all four files with different supports.

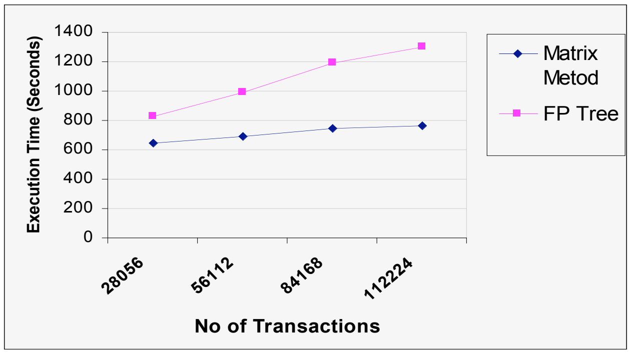

Figure 1: Comparison of FP Tree and Matrix Method with more repetitions in First.num.

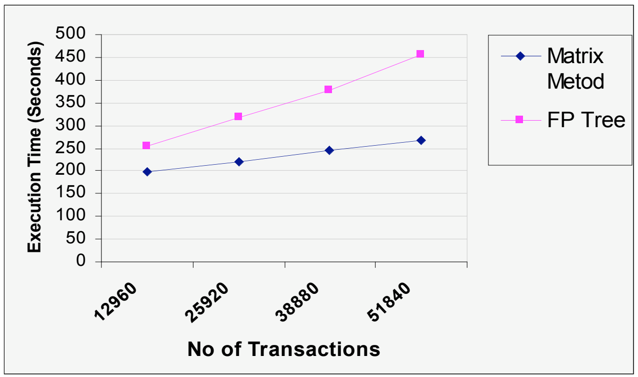

Figure 2: Comparison of FP Tree and Matrix Method with more repetitions in Second.num.

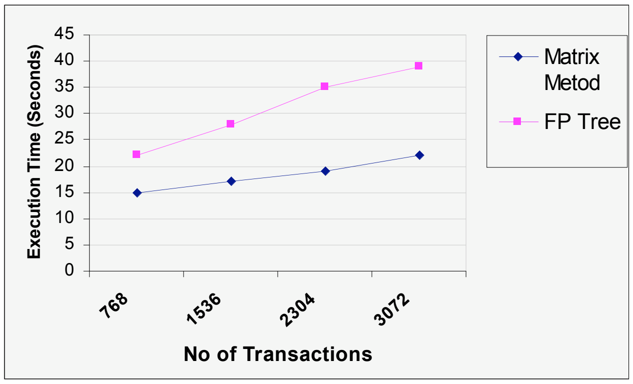

Figure 3: Comparison of FP Tree and Matrix Method with more repetitions in Pima.num.

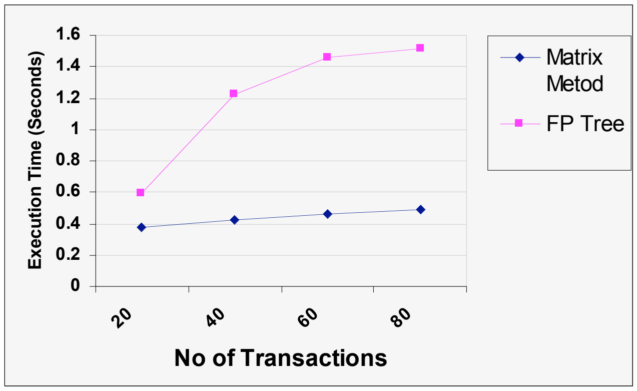

Figure 4: Comparison of FP Tree and Matrix Method with more repetition in transactional patterns in the example dataset given in Table **1**.

4. CONCLUSIONS

We have proposed a novel technique, it uses the algorithm that transforms the transactional database to a compressed form called transactional patternbase and also generate frequent items list in support descending order.

There are several advantages of the proposed method

1. Size of transactional patternbase is smaller than the transactional database. This patternbase can also be used for construction of FP-Tree which

will ultimately reduce the cost of FP Tree construction. While using the transaction patternbase, number of FP Tree nodes updates decreases, which also leads to the efficiency of the construction FP Tree.

2. The matrix, used to generate Association rules, is very small in size as compared to FP Tree. The rules are generated more quickly.

3. The size of matrix is not directly proportional to the no of transactions. If frequency of transactions is high, the size of matrix will be even smaller.

We have implemented this technique and studied its performance and found that this technique outperform as compared to FP Tree construction using transactional database.

We also observed that with increase in the transactional repetition in transactional database, the efficiency of the method will increase substantially.

It is important to note that the proposed method can also suffer the problem of incompetence, particular to the situation when there is no repetition of the transactional patterns in the transactional database.

Success of transactional patternbase opens new directions, it is interesting to re-examine and explore the existing algorithms that uses transactional databases and scan the database multiple times. In all these situations, transactional patternbase should be constructed during the first scan, and use that transactional patternbase in all remaining scans. This will also improve the efficiency of the algorithms. We are trying to construct FP-Tree from Transaction Patternbase instead of Transaction database.

REFERENCES

[1] Agrawal R. Imielinski T, Swami AN, Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, of SIGMOD Record, ACM Press 1993 volume 22(2): pp. 207-216. http://dx.doi.org/10.1145/170035.170072

[2] Zaki MJ, Mining Non-redundant Association Rules, Data Mining and Knowledge Discovery, 2004; 9: pp. 223-248. http://dx.doi.org/10.1023/B:DAMI.0000040429.96086.c7

[3] Agrawal R, Srikant R. Fast algorithms for mining association rules. Proceedings 20th International Conference on Very Large Data Bases, Morgan Kaufmann 1994; pp. 487-499.

[4] Brin S, Motwani R, Ullman JD, Tsur S, Dynamic itemset counting and implication rules for market basket data. Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, of SIGMOD Record, ACM Press, 1997; volume 26(2): pp. 255-264. http://dx.doi.org/10.1145/253260.253325

[5] Pei J, Han J, Mao R, CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets, ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000, Dallas, TX, 2000; pp. 21-30.

[6] Han J, Pei J, Yin Y, Mining frequent patterns without candidate generation. Proceeding of 2000 ACM SIGMOD Int. Conf. Management of Data (SIGMOD'00), Dallas, TX, May 2000; pp. 1-12.

[7] Han J, Pei J, Yin Y, Mao R, Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery 2003.

[8] Markus Hegland, Lecture Notes in Computer Science 2003; volume 2600: pp. 226-234. http://dx.doi.org/10.1007/3-540-36434-X_7

[9] Zaki MJ, Hsiao C-J, CHARM: An efficient algorithm for closed itemset mining. In R. Grossman. Proceedings of the Second SIAM International Conference on Data Mining 2002.

[10] THE LUCS-KDD SOFTWARE LIBRARY (LIVERPOOL UNIVERSITY COMPUTER SCIENCE KNOWLEDGE DISCOVERY IN DATAS) http://www.csc.liv.ac.uk/~frans/KDD/Software/FPgrowth/

Received on 17-02-2012 Accepted on 23-03-2011 Published on 07-04-2012

DOI: http://dx.doi.org/10.6000/1927-5129.2012.08.30