*International Journal of Computer Applications (0975 – 8887)*
*Volume 98– No.4, July 2014*

# A Performance based Transaction Reduction Algorithm for Discovering Frequent Patterns

V.Vijayalakshmi
Research Scholar
Manonmaniam Sundaranar University
Tirunelveli, T.N

A.Pethalakshmi
Head & Associate Professor of Comp. Science
M.V.M Government Arts College(W)
Dindigul, T.N

## ABSTRACT
Association rules are the main technique to determine the frequent item set in data mining. When a large number of item sets are processed by the database, it needs to be scanned multiple times. Consecutively, multiple scanning of the database increases the number of rules generation, which then consume more system resources. Existing approach TR-BAM scans the unnecessary transaction which takes more time to find frequent item set. This paper presents a modified transaction reduction technique named PBTRA which reduces the scanning times by cutting down the unnecessary transaction row. So, the corresponding item set is extracted directly without moving for entire database. Moreover, it exploits horizontal transaction of the matrix that automatically reduces the entire database scanning. Experimental results validate the performance of the proposed approach and expose that proposed method is more effective and efficient than previously proposed algorithm.

## Keywords
Frequent Item Set, Apriori, Support Count. TR-BAM, PBTRA

## 1. INTRODUCTION
The challenging task is to extracting useful information from the large collection of data in Dataware house and data base. Around the world lot of research is underway to discover the knowledge from the large collection of data in data warehouse. In this process many algorithms has been proposed to identify the associations between the data in the database, leads to mine the association rule among the data. Association rules are used for knowledge discovery and to take useful managerial decision in the organization based on the results of associations among data stepping toward to make a smarter system. In this regard, the first algorithm Apriori was proposed in the year 1994 by Agarwal and Srikanth to mine the frequent item set.[2] Time constraint and efficiency of algorithms leads to lot of research in the area of algorithm to build efficient algorithm which takes less time and few number of database scans to mine frequent item set and association rule.

Association rule is based mainly on discovering frequent item sets. Association rules are frequently used by retail stores to assist in marketing, advertising, inventory control, predicting faults in telecommunication network.

The remaining part of this paper is organized as follows: Section 2 contains existing Apriori and TRA approaches. In section 3, elaborate the proposed improved transaction reduction technique called PBTRA. Section 4 discussed about the performance analysis of proposed algorithm compared with Apriori and TRA algorithm. Section 5 contains conclusion.

## 2. EXISTING TECHNIQUES
## 2.1 Apriori Algorithm
Apriori employs an iterative approach known as a levelwise search, where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item,and collecting those items that satisfy minimum support. The resulting set is denoted L1. Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-temsets can be found. The finding of each Lk requires one full scan of the database. As the number of database scans are more the time complexity increases as the database increases [2]

## 2.2 TR-BAM/TRA Technique
TR-BAM discovers the frequent patterns in large databases by implementing a Bit Array Matrix. The whole database is scanned only once and the data is compressed in the form of a Bit Array Matrix. The frequent patterns are then mined directly from this Matrix. [16] But it takes more time to find frequent item set by scanning unnecessary transaction rows in the Matrix.

## 3. PROPOSED METHODOLOGY
In this method the frequent item sets are generated directly from the matrix which is generated from the transactional database. The major advantage of this approach is that, it reduces the scanning time by cutting down unnecessary rows in the matrix.

### 3.1 Steps involved in proposed algorithm are as follows
**PHASE 1:** Construct a matrix based on the presence and absence of items i.e. **"1" & "0"** indicate the presence and absence of items respectively.

**PHASE 2:** Preprocess the data by following two step procedure –
2.1 Count the number of 1's in column to check support count of an item.
2.2 Remove items which don't have minimum support count.

**PHASE 3:** Apply PBTRA technique on matrix.

**PHASE 4:** Generate Frequent item sets from BAM.

**PHASE 5:** End.

The proposed algorithm uses the following properties:

1. All the non empty subsets of a frequent itemset must also be frequent. So, there is no need to consider those frequent item sets which are having non frequent subsets.

2. Number of times transaction repeated in the database is represented by the count in the RC column and a new sum row stores the corresponding number of nonzero elements in the column on the bottom of the Bit Array Matrix.

3. Support count for 1 item set is sum of nonzero elements of each column.

4. We introduced an attribute Size_Of_Transaction (SOT), containing number of nonzero elements in individual row of the Bit Array Matrix. So, the corresponding item set is extracted directly without moving transactions scanning. If we need 3 item set means, it is not necessary to check SOT of 2.

The process is started from a given transactional database as shown in Table 1.

**TABLE 1**

| *TID* | *ITEMS* |
|---|---|
| T1 | I1,I2,I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

The steps of proposed algorithm are as follows:

**1.** The transaction database D is transformed into the Bit Array Matrix as shown in the Table 2.

**2.** The proposed methodology exploits horizontal transaction of the data set that automatically reduces the entire database scanning. In horizontal transaction, the number of 1's in each transaction is counted horizontally and Table 2 represents the horizontal transaction for the given data set.

**Table 2 Horizontal Transaction for the given data set**

| | *I1* | *I2* | *I3* | *I4* | *I5* | *SOT* |
|---|---|---|---|---|---|---|
| I1,I2,I5 | 1 | 1 | 0 | 0 | 1 | **3** |
| I2,I4 | 0 | 1 | 0 | 1 | 0 | **2** |
| I2,I3 | 0 | 1 | 1 | 0 | 0 | **2** |
| I1,I2,I4 | 1 | 1 | 0 | 1 | 0 | **3** |
| I1,I3 | 1 | 0 | 1 | 0 | 0 | **2** |
| I2,I3 | 0 | 1 | 1 | 0 | 0 | **2** |
| I1,I3 | 1 | 0 | 1 | 0 | 0 | **2** |
| I1,I2,I3,I5 | 1 | 1 | 1 | 0 | 1 | **4** |
| I1,I2,I3 | 1 | 1 | 1 | 0 | 0 | **3** |
| **SUM** | **6** | **7** | **6** | **2** | **2** | |

**3.** In Bit Array Matrix, the summation of nonzero elements in each column is the supporting count of item $I_j$. Set the minimum support count as min_ sup=2, when item supporting count is less than min_sup, all item sets containing the $I_j$ are infrequent itemsets. Move all those transactions from $C_1$ to $L_1$ whose sum value is not less than min_support (min_sup=2). So, the set of frequent 1-itemset is: L1= {{I1}, {I2}, {I3}, {I4}, {I5}}

**4.** If a transaction is occurring more than once in the transactional database then updates the repetition column (RC) of the Bit Array Matrix.

**Table 3 BAM after Generation of One Itemsets**

| *I1* | *I2* | *I3* | *I4* | *I5* | *SOT* | *RC* |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | **3** | **1** |
| 0 | 1 | 0 | 1 | 0 | **2** | **1** |
| 0 | 1 | 1 | 0 | 0 | **2** | **2** |
| 1 | 1 | 0 | 1 | 0 | **3** | **1** |
| 1 | 0 | 1 | 0 | 0 | **2** | **2** |
| 1 | 1 | 1 | 0 | 1 | **4** | **1** |
| 1 | 1 | 1 | 0 | 0 | **3** | **1** |

**5.** For generation of $L_2$, scan every row of the above matrix and consider all the 2-itemsets combinations of the elements which have value 1 in the rows. Then count the support for each itemset and move only those itemsets from $C_2$ to $L_2$ whose support is not less than min_sup. {{I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I3, I5}} will be frequent 2 itemsets.

**6.** Next, for generation of 3-itemsets combinations, from above matrix consider the transactions which have the TS value greater than two. The various possible combinations are {I1, I2, I5}; {I1, I2, I4}; {I1, I2, I3}; {I2, I3, I5}; {I1, I3, I5}. Now, by the property of the Apriori i.e. all the non empty subsets of a frequent itemset must also be frequent, it is found that

itemsets {I1, I2, I4};{I2, I3, I5};{I1, I3, I5} contain subsets which are not frequent. Therefore these itemsets are not included in $C_3$. $L_3$ will contain {I1, I2, I3} and {I1, I2, I5}. If there is a frequent 3 itemsets, then it will contain $I_1$, I2, I3, I5.

**7.** Similarly, 4-itemsets possible combinations are considered. i.e. {I1, I2, I3, I5}. This itemset doesn't satisfy the Apriori property. Therefore $C_4$=NULL and $L_4$= NULL. Hence all the frequent itemsets are generated.

---

**Algorithm 1 – PBTRA for FIM**

---

Min_sup. : Minimum support count

**Step 1:** Begin
**Step 2:** Read BAM
**Step 3:** Generate the set of frequent 1 itemset

Add RC & SOT columns // **.** Number of times transaction repeated in the database is represented by repetition count and Size_Of_Transaction (SOT), containing number of nonzero elements in individual row of the Bit Array Matrix.
k:=2;
while ($L_{k-1} \neq ø$) do
begin

calculate the sup_count for each k itemset
for each k itemset
if TS is greater than or equal to k.
compute support count for k itemsets
$(Ii,Ij\ldots\ldots\ldots,I_k)$
if sup_count >= min_sup then
$L_k$ := All candidates in $C_k$ with min_sup
end if
end if
end for
k := k + 1;
end
end
Answer := $U_k\ L_k$

**Step4:** End.

## 4. EXPERIMENTAL RESULT

In order to appraise the performance of the PBTR-BAM algorithm, we conducted an experiment using the Apriori algorithm, TR-BAM and the PBTRA algorithm.

### 4.1 Experiment 1

For this purpose, we select dataset from [15] ( 958X9 Database,) apply algorithms on same number of transaction and compare the execution time with support count 5,10,15,20,25 shown in Figure-1.

All experiments are performed on Intel core i3, 3.07GHz processor and 2GB of RAM, the algorithms were implemented in Java and tested on a Windows XP platform

**. Table 4 The Time Reducing Rate of TRA on the original Apriori according to the value of minimum support; The average of reducing time rate in the TRA is 43.46%.**

| MIN_SUP | APRIORI(S) | TRA(S) | TRR(%) |
|---|---|---|---|
| 5 | 0.359 | 0.156 | 56.54 |
| 10 | 0.266 | 0.141 | 46.99 |
| 15 | 0.265 | 0.141 | 46.79 |
| 20 | 0.188 | 0.125 | 33.51 |
| 25 | 0.188 | 0.125 | 33.51 |

**Table 5 The Time Reducing Rate of PBTRA on the original Apriori according to the value of minimum support, The average of reducing time rate in the PBTRA is 69.65%.**

| MIN_SUP | APRIORI(S) | PBTRA(S) | TRR(%) |
|---|---|---|---|
| 5 | 0.359 | 0.093 | 74.09 |
| 10 | 0.266 | 0.078 | 70.67 |
| 15 | 0.265 | 0.078 | 70.56 |
| 20 | 0.188 | 0.063 | 66.48 |
| 25 | 0.188 | 0.063 | 66.48 |

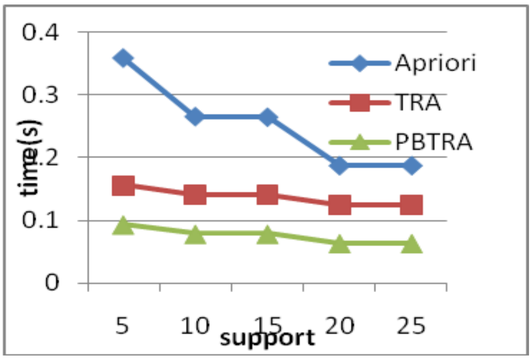


**Figure 1 – Time consuming comparison for different values of minimum support**

As we observe in figure 1, that the time consuming in proposed approach in each value of minimum support is less than it in the original Apriori, and the difference increases more and more as the value of minimum support decreases.

### 4.2 Experiment 2

The second experiment compares the time consumed of original Apriori, and our proposed algorithm by applying the three groups of transactions in the implementation. The result is shown in Figure 2.

- T1 : 958 Transactions
- T2 : 999 Transactions
- T3 : 1200 Transactions

**Table 6 The Time Reducing Rate of TRA on the original Apriori according to the number of transactions, The average of reducing time rate in the TRA is 50.54%.**

| MIN_SUP | APRIORI(S) | TRA(S) | TRR(%) |
|---|---|---|---|
| 958 | 0.343 | 0.172 | 49.85 |
| 999 | 0.360 | 0.188 | 47.77 |
| 1200 | 0.390 | 0.156 | 60 |

**Table 7 The Time Reducing Rate of PBTRA on the original Apriori according to the number of transactions, The average of reducing time rate in the PBTRA is 76.75%**

| MIN_SUP | APRIORI(S) | PBTRA(S) | TRR(%) |
|---|---|---|---|
| 958 | 0.343 | 0.094 | 72.59 |
| 999 | 0.360 | 0.109 | 69.72 |
| 1200 | 0.390 | 0.047 | 87.94 |

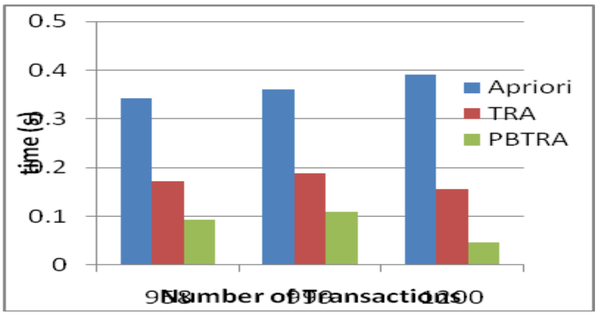


**Figure 2 - Time consuming comparison for different groups of transactions**

As we observe in figure 2, that the time consuming in proposed approach in each group of transactions is less than it in the original Apriori, and the difference increases more and more as the number of transactions increases.

## 5. CONCLUSION

We proposed a new approach PBTRA algorithm for discovering frequent pattern among Boolean databases of transactions. We compared the new approach with Apriori, TR-BAM algorithms and illustrated the experimental result in Figure-1 and Figure -2, shows that the proposed technique performs better in order to time efficiency.

## 6. REFERENCES

[1]Agrawal, R., Imielinski, T., and Swami, A. N. Mining Association Rules Between Sets of Items in Large Databases. Proceedings of the ACM SIGMOD, International Conference on Management of Data, pp.207- 216,

[2] Agrawal. R., and Srikant. R., Fast Algorithms for Mining Association Rules, Proceedings of 20th International Conference of Very Large Data Bases. pp.487-499,1994.

[3]M. S. Chen, J. Han, and P. S. Yu. Data mining: An overview from a database Perspective. IEEE Trans. Knowledge and Data Engineering, 8:866-883, 1996.

[4]U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996.

[5]Agarwal, R. Agarwal, C. and Prasad V., A tree projection algorithm for generation of frequent item sets. In J. Parallel and Distributed Computing, 2000

[6] L. Cheng and B. B. Wang, "An Improved Apriori Algorithm for Mining Association Rules, " Comput. Eng., Shanghai, vol. 28(7), pp. 104-105, 2002.

[7] Sheng Chai; Jia Yang; Yang Cheng;," The Research of Improved Apriori Algorithm for Mining Association Rules," Service System and Service Management, 2007 International Conference on, vol., no.,pp.1-4, 9-11 June 2007

[8] Li Xiaohong,Shang Jin.An improvement of the new Apriori algorithm [J].Computer science, 2007,34 (4) :196-198. 2007

[9]Wanjun Yu, Xiachun Wang and et.al, (2008), "The Research of Improved Apriori Algorithm for Mining Association Rules", pp. 513-516

[10] PEI Guying. A Fast Algorithm for Mining of Association Rules Based on Boolean Matrix. *Automation & Instrumentation.* 2009; 5: 16-18.

[11]LV Taoxia, LIU Peiyu. Algorithm for Generating Strong Association Rules Based on Matrix. *Application Research of Computers.* 2011; 28(4): 1301- 1303

[12] ZHANG Zhongping, LI Yan, YANG Jing. Frequent Itemsets Mining Algorithm Based on Matrix. *Computer Engineering.* 2009; 35(1): 84-85.

[13]S.Prakash, R.M.S.Parvathi., An Enhanced Scaling Apriori for Association Rule Mining Efficiency. European Journal of Scientific Research, ISSN 1450-216X Vol.39 No.2 (2010), pp.257-264

[14]Wang Lifeng. An Efficient Association Rule Algorithm Based on Boolean Matrix. *International Review on Computers and Software*. 2012; 7(2): 695-700.

[15] Database URL: http://www2.cs. Uregina .ca/~ dbd/ cs831/ notes/ itemsets/ datasets.php.

[16]V.Vijayalakshmi, A.Pethalakshmi., "Mining of Frequent Itemsets With an Enhanced Apriori Algorithm", IJCA, Vol 81-No.4, November 2013