ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2016.0163



# AN EFFICIENT DATA MINING METHOD TO FIND FREQUENT ITEM SETS IN LARGE DATABASE USING TR- FCTM

**Saravanan Suba[1] and T. Christopher[2]**
[1]*Department of Computer Science, Kamarajar Government Arts College, India*
E-mail: saravanansuba@rediffmail.com
[2]*Department of Computer Science, Government Arts College, Coimbatore, India*
E-mail: chris.hodcs@gmail.com

***Abstract***
***Mining association rules in large database is one of most popular data mining techniques for business decision makers. Discovering frequent item set is the core process in association rule mining. Numerous algorithms are available in the literature to find frequent patterns. Apriori and FP-tree are the most common methods for finding frequent items. Apriori finds significant frequent items using candidate generation with more number of data base scans. FP-tree uses two database scans to find significant frequent items without using candidate generation. This proposed TR-FCTM (Transaction Reduction- Frequency Count Table Method) discovers significant frequent items by generating full candidates once to form frequency count table with one database scan. Experimental results of TR-FCTM shows that this algorithm outperforms than Apriori and FP-tree.***

***Keywords:***
***Apriori, FP-Tree, TR-FCTM, Minimum Support***

## 1. INTRODUCTION

In recent years there has been an exponential progress in the generation and handling of electronic information as more and more tasks are computerized. Any organization or enterprise has started to realize that the information gathered over years is an important strategic benefit and it also recognizes that there are potential intelligences secreted in the large amount of data. So it requires techniques to extract the most valuable information from accumulated data [1].

The data mining provides such techniques to find useful hidden information. Data mining is a group of methods for effective automated discovery of previously unknown, valid, novel, valuable and understandable pattern in large databases [1], [2]. It has recently attracted considerable attention from database experts and researchers because of its applicability in several areas such as decision support, market strategy and financial forecasts [3].

The data mining techniques or tasks can be mostly classified as descriptive or predictive. Descriptive mining denotes to the method in which the essential characteristics or common properties of the data in the data base are depicted. The descriptive techniques involve tasks like clustering, association and sequential mining [4]. Predictive data mining tasks are those that perform inference on input data to arrive at hidden knowledge and make interesting and useful estimate [5]. The predictive mining techniques involve tasks like classification, regression and deviation [4]. Key research topics or challenges in data mining are performance, mining methodology, user interaction and data diversity. So the data mining algorithm and methodologies must be competent and scalable well to the size of data base and their execution times [5].

One of the most popular descriptive data mining techniques is association rule mining [4]. Since its introduction [7], association rule mining has become one of the core data mining tasks and has attracted remarkable interest among data mining researches and experts [8]. It is a good method for finding correlations (association rule) between variables in large database. For example,

$$\forall x \in \text{persons}, \text{buys}(x, \text{“bread”}) \rightarrow \text{buys}(x, \text{“butter”}) \quad (1)$$

where, $x$ is a variable and buy($x$, $y$) is a predicate that states that person $x$ purchase item $y$. This rule specifies that a high percentage of people who purchase bread also buy butter [9].

Most of the algorithms are created on traditional algorithm of association rule mining [7], [10]. Association rule mining can be formally defined as follows. Let $I = \{i_1, i_2 \ldots i_n\}$ is a set of items. Any subset of $I$ is called itemset. Let $D$ be a set of transactions. Each transaction $T$ is a set of items such that $T \subseteq I$. Each transaction has a unique identifier (TID). Let $P$, $Q$ be a set of items, Association rule has the form $P \rightarrow Q$, $P \wedge Q = \phi$, where $P$ is an antecedent and $Q$ is the consequent of the rule. It uses two statistical methods that control the activity of association rule mining are support and confidence [2]. Firstly, it determines frequent item set based on minimum support count. After that, minimum confidence is used to find association rules between frequent items. The support and confidence can be mathematically represented as follows.

$$\text{Support}(P \rightarrow Q) = \Sigma(P \cup Q)/N \quad (2)$$

$$\text{Confidence}(P \rightarrow Q) = \Sigma(P \cup Q)/\Sigma P \quad (3)$$

Many researches have been done in developing efficient method for finding frequent patterns [3],[6],[8],[9],[10] [11],[12],[13],[14] after introducing Apriori by Agrawal et al. [7]. Among those methods, the Apriori [7] and FPTree [14] are the most popular methods for finding frequent items. FP-tree uses different algorithm strategy and structure to find frequent items with less computing time than Apriori. This paper introduces the TR-FCTM method to find significant frequent items with less computing time than FP-tree.

The rest of the paper is organized as follows: Related works are described in section 2. The proposed algorithm is discussed in section 3. Experimental results and discussions are given in section 4. The conclusion and the ideas for future work are written in section 5.

## 2. RELATED WORKS

Mining of frequent itemsets is the main phase in association mining which determines frequent itemsets in transactions database. It is an essential in many tasks of data mining methods.

Many algorithms are suggested to discover frequent itemsets, but all of them can be categorized into two classes: candidate generation or pattern growth [15].

The AIS (Agrawal, Imielinski and Swami) algorithm introduced by Agrawal et al. [7] was the forerunner of all the algorithms used to find the frequent itemsets and confident association rules. It was renamed as Apriori by Agrawal et al. [10], [16]. Though a number of algorithms were placed forth following the introduction of Apriori algorithm, a majority of them dealt with the optimization of one or more steps of the Apriori bearing the related general structure. Apriori algorithm suffers from many numbers of database scans required to find the frequent items set and take more time if the database size is increased [17].

FP-tree was proposed by Han in 2000 which represents pattern growth approach and it uses a specific data structure FP-tree. It reduces number of data base scans required by constructing FP-tree structure but it suffers from the time required to construct a FP-Tree structure for large database. The increase in the size of the FP -tree with respect to the growth of database leads to difficulties in constructing, search and insert operation on large FP-tree.

Although computing power has increased terrifically over the years, competent algorithms with customized data structures are still necessary to get timely outcomes. This is specifically true for data mining as it is a computationally-intensive method. So this paper introduces the TR-FCTM method to find significant frequent items so that to reduce computing time than FP-tree.

## 3. PROPOSED METHODOLOGY

### 3.1 REDUCING THE DATABASE SCANS USING TRANSACTION MERGING

It is based on the observation that transaction databases frequently hold identical transactions. The technique consists of finding these transactions and to swap them with single transaction. This can be mathematically defined as, if a set of identical transactions ($Tx1$, $Tx2$,…, $Txm$) are in a database $D$ then it is replaced by a single new transaction as ($TM$, $m$) where $TM$ is the identical itemset and $m$ is the total number of particular identical itemset in the database [18]. This reduces the total transactions in database as less than or equal to $2^I$ -1 transactions where, $I$ represents total the number of different items in the shop. So this greatly reduces computing time of finding frequent itemset.

### 3.2 FREQUENCY COUNT TABLE (FCT)

Let it $X$ be any of an itemset in database $D$. It states that an itemset $X$ of transaction $T$ is a subset of $I$($X \subseteq I$) and a set of such transactions form the database $D$. So in database $D$ every transaction itemset $X$ will be an element of $2^I$-1, where $2^I$ is a power set of $I$. Power set of $I$ contain all the subsets of $I$ that may be in the form of transactions itemset in the transaction database $D$ except $\phi$. Hence our algorithm employ one table that's name is Frequency Count Table. It has two fields such as itemset and frequency count value. This table creates entries of frequency count of each itemset that are detected in transaction database. The frequency count of each itemset is the count of the existence of such itemset in transactional database $D$. This table is created and may be kept in memory till the frequent itemset are not found [19]. The format of frequency table count is given in Table.1.

Table.1. Structure of Frequency Count Table

| Sl. No. | Itemset(X) | Frequency Count(FC) |
|---|---|---|
| 1 | | |
| . | | |
| . | | |
| $2^I$-1 | | |

For example, Let $I$ = {$I1$, $I2$, $I3$} be the set of items and the different types of itemset that can be created from $I$ are {$I1$}, {$I2$}, {$I3$},…, {$I1$, $I2$, $I3$}. The frequency count table for above said items with initial value is given in Table.2.

Table.2. Frequency Count Table with initial value

| Sl. No. | Itemset(X) | Frequency Count(FC) |
|---|---|---|
| 1 | {$I1$} | 0 |
| 2 | {$I2$} | 0 |
| 3 | {$I3$} | 0 |
| 4 | {$I1$, $I2$} | 0 |
| 5 | {$I1$, $I3$} | 0 |
| 6 | {$I2$, $I3$} | 0 |
| 7 | {$I1$, $I2$, $I3$} | 0 |

### 3.3 PROPOSED ALGORITHM

This proposed work employs transaction merging and FCT to find the significant frequent items. First it merges the similar transactions in the database and stores the merged transactions in the main memory. Later it reads the merged transactions one by one from main memory and update the FCT correspondingly.

To find the frequent itemset for any threshold value it scans the FCT not the database. FCT has entries of frequency count of all itemset but not the total support count of that itemset. The frequency count of each itemset is the count of the direct existence of such itemset in transactional database $D$. The total count of particular itemset $X$ is calculated by comparing whether it a subset of all its bigger itemset in the FCT. If total frequency count of particular itemset is greater than or equal to ST (Support Threshold) then the itemset is included in the FI (Frequent Itemsets).

1. ***Algorithm:*** *The TR-FCTM*
2. ***Input:*** *A database D and the support Threshold ST*
3. ***Output:*** *frequent itemsets FI*
4. *Begin*
5. *Construct TRT;*
6. *Scan the raw database D record by record until it reaches the end of record*
7. *{*

*Update the TRT ;*

8. *Count the number of different items involved in the data base;*
9. *}*
10. *Construct the FCT;*
11. *Scan TRT and update corresponding entry in FCT;*
12. *FI = { ϕ };*
13. *For ( i=1; i<2^I; i++)*

*//for each itemset $X_i$ in FCT, Repeat the steps*

14. *{*
15. *$TC_{Xi}$ = 0;*
16. *For (j=i+1; j< 2^I; j++)*
17. *{*
18. *If Xi ⊆ Xj;*
19. *{*
20. *$TC_{Xi}$ = $TC_{Xi}$ +FCT.FC (j);*
21. *}*
22. *}*
23. *if ( $TC_{xi}$ >= ST)*
24. *{*
25. *FI = FI U Xi;*
26. *}*
27. *}*
28. *End.*

## 3.4 ILLUSTRATION OF PROPOSED MOTHED

Suppose this problem has the transaction set D with 10 transactions , minimum support threshold values as 2 and item set *I* = {A,B,C}. The transaction set is shown in Table.3.

Table.3. Transaction Database

| Tid | Itemset |
|---|---|
| T1 | A, B, C |
| T2 | A, B |
| T3 | B, C |
| T4 | A |
| T5 | A |
| T6 | A, B |
| T7 | B, C |
| T8 | B, C |
| T9 | A, B |
| T10 | A |

The algorithm scans the transactions one by one and merges the identical transactions and store those merged transactions into a table called TRT as shown in Table.4.

Table.4. Transaction Reduction Table (TRT)

| Sl. No. | Itemset | Count |
|---|---|---|
| 1 | A, B, C | 1 |
| 2 | A, B | 3 |
| 3 | B, C | 3 |
| 4 | A | 3 |

Next step is to construct the FCT with initial values as shown in Table.5.

Table.5. Initial Frequency Count Table

| Sl. No. | Itemset | Frequency Count |
|---|---|---|
| 1 | A | 0 |
| 2 | B | 0 |
| 3 | C | 0 |
| 4 | A, B | 0 |
| 5 | A, C | 0 |
| 6 | B, C | 0 |
| 7 | A, B, C | 0 |

The FCT is updated for each transaction scan as shown in Table.6.

Table.6. Updated Frequency Count Table

| Sl. No. | Itemset | Frequency Count |
|---|---|---|
| 1 | A | 3 |
| 2 | B | 0 |
| 3 | C | 0 |
| 4 | A, B | 3 |
| 5 | A, C | 0 |
| 6 | B, C | 3 |
| 7 | A, B, C | 1 |

Finally, the total frequency counts of all combinations of itemsets are generated as stated in the proposed algorithm. The value TFCT is shown in Table.7.

Table.7. Total Frequency Calculation

| Sl. No. | Itemset | Total Frequency Count |
|---|---|---|
| 1 | A | 7 |
| 2 | B | 7 |
| 3 | C | 4 |
| 4 | A, B | 4 |
| 5 | A, C | 1 |
| 6 | B, C | 4 |
| 7 | A, B, C | 1 |

It is observed from Table.7 that the frequent itemsets for given set of transactions are,

FI = {{A}, {B}, {C}, {A, B}, {B, C}}

## 4. EXPERIMENTAL ANALYSIS

To study the performance of this proposed algorithm, it has been done several experiments. The intel core™ i5-2450m CPU @2.5GHZ, 4.0GB RAM, 64 bit windows 7 operating system and NetBeans IDE 8.0.2 were used to conduct the experiment. The synthetic Data set of 100,1000, 2000 and 4000 with 5 items were created to compare this proposed TR-FCTM with Apriori and FP-tree in terms of time required to find significant frequent itemsets from given datasets.

The first experiment compares the execution time consumed of Apriori, FP-tree and proposed algorithm by applying the above said four groups of datasets. The Table.8 shows execution time to find significant frequent items generated by Apriori, FP-tree and TR-FCTM for four groups of data sets with 20% minimum support threshold.

Table.8. Execution Time Comparision for Different Data Sets

| No. of Transactions | Execution Time in ms | | |
|---|---|---|---|
| | Apriori | FP-Tree | TR-FCTM |
| 100 | 18 | 14 | 12 |
| 1000 | 56 | 35 | 19 |
| 2000 | 100 | 45 | 25 |
| 4000 | 150 | 65 | 31 |

It is observed that the execution time is decreased linearly from Apriori to FP-tree and FP-tree to TR-FCTM and the difference increases more and more as the number of transactions increases.

The Fig.1 demonstrates the performance of Apriori, FP-tree and TR-FCTM according to the execution time of each algorithm for given four groups of transactions. It is easily observed that the TR-FCTM outperforms than Apriori and FP-tree.

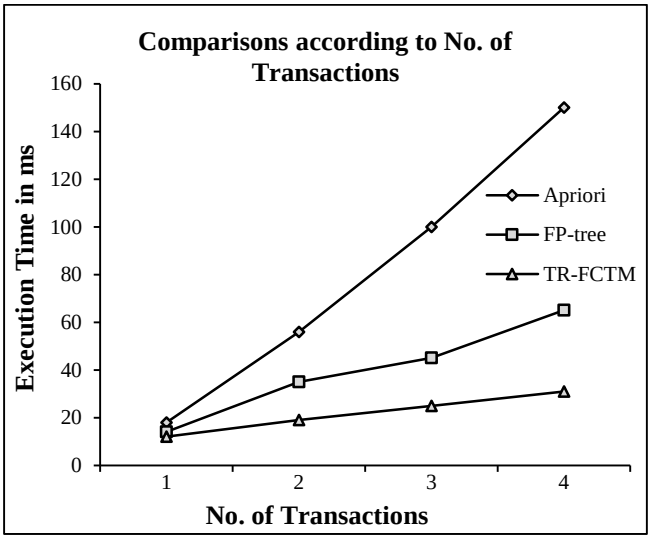


Fig.1. Time Comparison for Different Group of Transaction sets

The second experiment compares the execution time consumed of Apriori, FP-tree and proposed algorithm by applying 2000 transactions through 3 different minimum support thresholds. The Table.9 shows execution time to find significant frequent items generated by Apriori, FP-tree and TR-FCTM.

Table.9. Execution time comparision for different thresholds

| Minimum Support Threshold | Execution Time in ms | | |
|---|---|---|---|
| | Apriori | FP-tree | TR-FCTM |
| 10 | 110 | 46 | 27 |
| 25 | 100 | 45 | 25 |
| 50 | 66 | 41 | 24 |

The Fig.2 demonstrates the performance of Apriori, FP-tree and TR-FCTM according to the execution time for 3 different minimum support thresholds with 2000 transactions. It is easily seen that the TR-FCTM outperforms than Apriori and FP-tree.

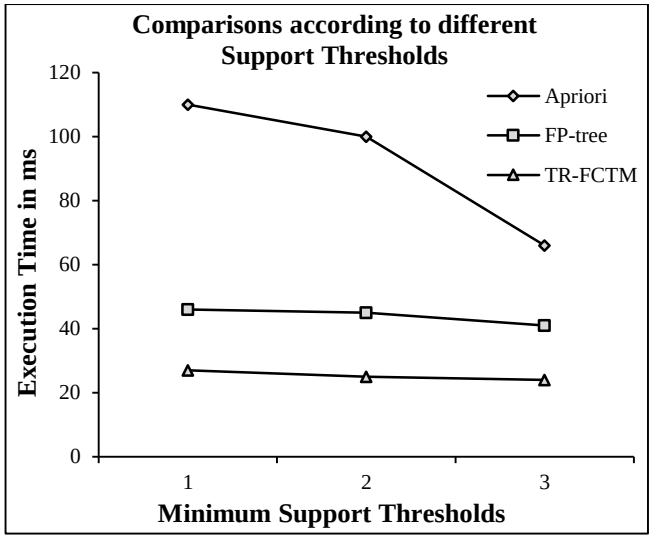


Fig.2. Time Comparison of Different Threshold levels

### 4.1 THE ANALYSIS AND EVALUATION OF THE TR-FCTM

It is observed that the time consuming in TR-FCTM is less than the Apriori and FP-tree with reference to the Fig.1 and Fig.2 [15].

Table.10. Time Reduction Rate with Different Datasets (TR-FCTM & Apriori)

| No. of Transactions | Execution Time in ms | | % of Time Reduction |
|---|---|---|---|
| | Apriori | TR-FCTM | |
| 100 | 18 | 12 | 33.33% |
| 1000 | 56 | 19 | 66.07% |
| 2000 | 100 | 25 | 75.00% |
| 4000 | 150 | 31 | 79.33% |

The Table.10 shows that the time reduction rate of using TR-FCTM against Apriori according to vary the number of transactions. It proves that the time reduction rate increases as number of transactions increases. The average time reduction rate for using TR-FCTM against Apriori of this case is 63.43%.

Table.11. Time Reduction Rate with Different Datasets (TR-FCTM & FP-tree)

| No. of Transactions | Execution Time in ms | | % of Time Reduction |
|---|---|---|---|
| | FP-Tree | TR-FCTM | |
| 100 | 14 | 12 | 14.28% |
| 1000 | 35 | 19 | 45.71% |
| 2000 | 45 | 25 | 44.44% |
| 4000 | 65 | 31 | 52.30% |

The Table.11 indicates that the time reduction rate of using TR-FCTM against FP-tree by varying the number of transactions. Even though, the time reduction rate fluctuates as number of transactions increases, it agrees that the use of TR-FCTM will reduce the time. The average time reduction rate for using TR-FCTM against FP-tree of this case is 39.18%.

Table.12. Time Reduction Rate of Using TR-FCTM Against Apriori with different threshold levels

| Minimum Support Threshold | Execution Time in ms | | % of Time Reduction |
|---|---|---|---|
| | Apriori | TR-FCTM | |
| 10 | 110 | 27 | 75.45% |
| 25 | 100 | 25 | 75.00% |
| 50 | 66 | 24 | 63.63% |

The Table.12 shows that the time reduction rate of using TR-FCTM against Apriori by varying support thresholds for 2000 transactions. Even though, the time reduction rate varies according to value of support threshold, it confirms that the use of TR-FCTM will reduce the time. The average time reduction rate for using TR-FCTM against Apriori for this case is 71.36%.

Table.13. Time Reduction Rate of Using TR-FCTM Against FP-tree with different threshold levels

| Minimum Support Threshold | Execution time in ms | | % of time reduction |
|---|---|---|---|
| | FP-tree | TR-FCTM | |
| 10 | 46 | 27 | 41.30% |
| 25 | 45 | 25 | 44.44% |
| 50 | 41 | 24 | 41.46% |

The Table.13 shows that the time reduction rate of using TR-FCTM against FP-tree by varying support thresholds for 2000 transactions. Even though the time reduction rate is approximately equal according to various value of support threshold, it accepts that the use of TR-FCTM will reduce the time. The average time reduction rate for using TR-FCTM against Apriori for this case is 42.4%.

## 5. CONCLUSIONS AND FUTURE WORKS

Frequent pattern mining algorithms are very important to find interesting patterns in large data base. This TR-FCTM is designed using transaction merging, finding direct frequency count and total frequency count for an itemsets. The experimental results show that the proposed methodology consumes less time for transaction scanning and candidate generation while it is compared with Apriori and FP-tree. So the TR-FCTM outperforms than Apriori and FP-tree for horizontal scalability of transactions for synthetically generated datasets. The vertical scalability and real time data set should be applied in future to improve its efficiency further.

## REFERENCES


[1] G.K. Gupta, "*Introduction to Data Mining with Case Studies*", Prentice-Hall of India Pvt. Limited, 2006.

[2] Saravanan Suba and Christopher T, "A Study on Milestones of Association Rule Mining Algorithms in Large Databases", *International Journal of Computer Applications*, Vol. 47, No. 3, pp. 12-19, 2012.

[3] Ya-Han Hu and Yen-Liang Chen, "Mining Association Rules with Multiple Minimum Supports: A New Mining Algorithm and A Support Tuning Mechanism", *Decision Support Systems*, Vol. 42, No. 1, pp. 1-24, 2006.

[4] S. Shankar and T. Purusothaman, "Utility Sentient Frequent Itemset Mining and Association Rule Mining: A Literature Survey and Comparative Study", *International Journal of Soft Computing Applications*, No. 4, pp. 81-95, 2009.

[5] N.P. Gopalan and B. Sivaselvan, "*Data Mining Techniques and Trends*", PHI Learning, 2009.

[6] Ashoka Savasere, Edward Omiecinski and Shamkant B. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", *Proceedings of the 21st International Conference on Very Large Data Bases*, pp. 432-444, 1995.

[7] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 207-216, 1993.

[8] M.J. Zaki and C.J. Hsiao, "CHARM: An Efficient Algorithm for Closed Association Rule Mining", Technical Report 99-10, Department of Computer Science, Rensselaer Polytechnic Institute, 1999.

[9] Yin-Ling Cheung and A.W.-C. Fu, "Mining Frequent Itemsets without Support Threshold: with and without Item Constraints", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 9, pp. 1052-1069, 2004.

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

[11] Jong Soo Park, Ming-Syan Chen and Philip S. Yu, "Using a Hash- Based Method with Transaction Trimming and Database Scan Reduction for Mining Association Rules", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 5, pp. 813-825, 1997.

[12] H. Toivonen, "Sampling Large Databases for Association Rules", *Proceedings of the 22th International Conference on Very Large Data Bases*, pp. 134-145, 1996.

[13] Jong Soo Park, Ming-Syan Chen and Philip S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules", *Proceedings of the 1995 ACM SIGMOD*

*International Conference on Management of Data*, pp. 175-186, 1995.

[14] Jiawei Han, Jian Pei and Yiwen Yin, "Mining Frequent Patterns without Candidate Generation", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1-12, 2000.

[15] Mohammed Al-Maolegi1, Bassam Arkok, "An Improved Apriori Algorithm for Association Rules", *International Journal on Natural Language Computing*, Vol. 3, No. 1, pp. 21-29, 2014.

[16] R. Srikant and R. Agrawal, "Mining Generalized Association Rules", *Future Generation Computer Systems*, Vol. 13, No. 2-3, pp. 161-180, 1997.

[17] S. Sunil Kumar, S. Shyam Karanth, K.C. Akshay, Ananth Prabhu and M. Bharathraj Kumar, "Improved Apriori Algorithm based on Bottom Up Approach using Probability and Matrix", *International Journal of Computer Science Issues*, Vol. 9, No. 2, pp. 232-246, 2012.

[18] Z. Souleymane, P.F. Viger, J.C.-W. Lin, C.-W. Wu and V.S. Tseng, "EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining", *Proceedings of 14th Mexican International Conference on Artificial Intelligence*, pp. 530-546, 2015.

[19] R. Ahirwal, N.K. Kori and Y.K. Jain, "Improved Data Mining Approach to Find Frequent Itemset using Support Count Table", *International Journal of Emerging Trends & Technology in Computer Science*, Vol. 1, No. 2, pp. 195-201, 2012.