International Journal For Technological Research In Engineering
Volume 2, Issue 2, October-2014 *ISSN (Online): 2347 - 4718*

REVIEW AND ANALYSIS OF VARIOUS EFFICIENT FREQUENT PATTERN ALGORITHMS

Ms. Mansi Trivedi
PG Scholar, Department of Computer Science
Kalol Institute of Technology and Research Centre, Kalol, Gujarat, India

Abstract: Frequent pattern mining is the widely researched field in data mining because of its importance in many real life applications. Many algorithms are used to mine frequent patterns which give different performance on different datasets. Apriori, Eclat and FP Growth are the initial basic algorithm used for frequent pattern mining. CATS Tree extends the idea of FP-Tree to improve storage compression and allow frequent pattern mining without generation of candidate item sets. It allows mining only through a single pass over the database. The efficiency of Apriori, FP-Growth, and CATS Tree for incremental mining is very poor. The implemented CATSIM Tree uses more memory compared to Apriori, FP-Growth and CATS Tree, but with advancement in technology, is not a major concern. In this work CATSIM Tree with modifications in CATS Tree is implemented to support incremental mining with better results.

Keywords: Frequent itemset: Pattern discovery: Tree structure.

I. INTRODUCTION

Data mining has long been an active area of research in databases. A database is a systematically arranged collection of data, so that it can be retrieved and manipulated easily at a later time. There are different kinds of database, like active database, cloud database, embedded database and transactional database etc, but in this paper the researcher deals with transactional database only. A transactional database is a database in which there is no auto commit. Most modern relational database are the transactional database [3]. A database layout tells how data is represented. There are two layouts which are in common use, horizontal layout and the vertical layout. It is a divide and conquers mechanism which reduces the size of database recursively by considering only the longest pattern. A frequent pattern is a pattern which occurs in comparatively more transactions. A frequent item set is an item set whose support is greater than some user-specified minimum support. The presented paper is organized in five sections: the first section contains the introduction; the second section presents a brief description of the three frequent pattern mining algorithms namely Apriori, Eclat, FP Growth, CATS, CATS-FELINE, FPM and CATS SIM tree. The third section gives the methodology used. The fourth section presents a comparative analysis of the algorithms used under varying conditions. Fifth section gives the conclusion and in the last references is listed.

II. FREQUENT PATTERN MINING ALGORITHMS

Now the researcher elaborates the various frequent itemset mining algorithms.

A. Apriori Algorithm

Apriori is the very first algorithm for mining frequent patterns. It was given by R Agarwal and R Srikant in 1994 [5].It works on horizontal layout based database. It is based on Boolean association rules which uses generate and test approach. It uses BFS (breadth first search). Apriori uses frequent k itemsets to find a bigger itemset of k+1 item. The calculation of frequency of an item is done by counting it's occurrence in all transactions [6]. All infrequent items are dropped. Apriori property: All subsets of a frequent itemsets which are non empty are also frequent. Apriori follows two steps approach: In the first step it joins two itemsets which contain k-1 common items in kth pass. The first pass starts from from the single item; the resulting set is called the candidate set Ck. In the second step the algorithm counts the occurrence of each candidate set and prune all infrequent itemsets. The algorithm ends when no further extension found.

B. Eclat Algorithm

Eclat is a vertical database layout algorithm used for mining frequent itemsets. It is based on depth first search algorithm. In the first step the data is represented in a bit matrix form. If the item is bought in a particular transaction the bit is set to one else to zero. After that a prefix tree needs to be constructed. To find the first item for the prefix tree the algorithm uses the intersection of the first row with all other rows, and to create the second child the intersection of the second row is taken with the rows following it [4]. In the similar way all other items are found and the prefix tree get constructed. Infrequent rows are discarded from further calculations. To mine frequent itemsets the depth first search algorithm is applied to prefix tree with backtracking. Frequent patterns are stored in a bit matrix structure. Eclat is memory efficient because it uses prefix tree. The algorithm has good scalability due to the compact representation.

C. FP Growth Algorithm

Frequent pattern growth also labelled as FP growth is a tree based algorithm to mine frequent patterns in database the idea was given by (han et. al. 2000) [10]. It is applicable to projected type database. It uses divide and conquer method [7]. In it no candidate frequent itemset is needed

rather frequent patterns are mined from FP tree. In the first step a list of frequent itemset is generated and sorted in their decreasing support order. This list is represented by a structure called node. Each node in the FP tree, other than the root node, will contain the item name, support count, and a pointer to link to a node in the tree that has the same item name [6]. These nodes are used to create the FP tree. Common prefixes can be shared during FP tree construction. The paths from root to leaf nodes are arranged in non increasing order of their support. Once the FP tree is constructed then frequent patterns are extracted from the FP tree starting from the leaf nodes. Each prefix path subtree is processed recursively to mine frequent itemsets. FP Growth takes least memory because of projected layout and is storage efficient. A variant of FP tree is conditional FP tree that would be built if we consider transactions containing a particular itemset and then removing that itemset from all transactions. Another variant is parallel FP growth (PFP) that is proposed to parallelize the FP tree on distributed machines [8]. FP Growth is improved using prefix-tree-structure, Grahne and Zhu [9].

D. CATS Tree

In the present study, we have developed a novel data structure, CATS Tree, an extension of FP-Tree [3]. Researchers have proposed to use tree structure in data mining [3]. However, they are not suitable for interactive frequent pattern mining. CATS Tree is a prefix tree and it contains all elements of FP-Tree including the header, the item links etc. Paths from the root to the leaves in CATS Tree represent sets of transactions. We use the database in Table 1 to illustrate the construction of a CATS Tree. Initially, the CATS Tree is empty. Transaction 1 is added as it is. Transaction 2 is added, common items, F, A, C, is extracted from Transaction 2 and is merged with the existing tree. Although item D is not contained in Transaction 2, common items could be found underneath node D. Item M is found to be common. However, Transaction 2 cannot be merged directly at node M because it would violate the structure of CATS tree that the frequency of a parent node must be greater than the sum of its children's frequencies. Node M of CATS Tree is swapped in front of node D and it is merged with the transaction. After that, there is no more common item. The remaining portion of Transaction 2 is added to node M.

Table 1: Transaction Data [13].

TID	Original Transaction
1	A, F, C, D, G, I, B, P
2	A, B, C, F, L, M, O
3	B, A, H, J, O
4	B, C, K, S, P
5	A, F, C, E, L, P, M, N

Transaction 3 is added. Item F of Transaction 3 is merged. Since the frequency of node A is the same as that of node F, The search for other possible merge nodes continues along the branch. It passes through node A, C, and M and finally, reaches node B. Even though Transaction 3 also contains an item B, but the frequency of node B is smaller than that of node M, the remaining of the transaction is inserted as a new branch at node F. When Transaction 4 is added, there is no common item. Transaction 4 is added as it is. In Figure 2, Transaction 5 is added; F, A, C, and M are merged. The search for common items continues along the path. Item P is common in both the tree path and Transaction 5. This triggers swapping of node P to the front of node D. After item P is merged, there is no more common item. The remainders of Transaction 5 are inserted as a new branch at node P. Finally we get answer in figure 1.

All CATS Trees have the following properties:

1) The compactness of CATS Tree measures how many transactions are sharing a node. Compactness decreases as it is getting away from the root. This is the result of branches being arranged in descending order.

2) No item of the same kind could appear on the lower right hand side of another item. If there were items of the same kind on the right hand side, they should have been merged with the node on the left to increase compression. Any items on the lower right hand side can be switched to the same level as the item, split nodes as required if switching nodes violates the structure of CATS Tree [14].

E. CATS-FELINE and FPM

In the mining process with a CATS-tree, the CATS-FELINE algorithm builds a conditional condensed CATS-tree for each frequent item p by gathering all transactions that contain p. A conditional condensed CATS-tree is one in which all infrequent items are removed and is different from a conditional FP-tree. In order to ensure that all frequent patterns are captured by CATS-FELINE, it has to traverse both up and down the CATS-tree.

CATS-FELINE the overall mining process proceeds in three phases:

Step 1: Convert the CATS tree generated from a database scan into a condensed tree with nodes having the frequency count less than the minimum support removed.

Step 2: Construct conditional condensed CATS-trees (also known as alpha-trees) for items in the header table with frequency counts greater than the minimum support.

Step 3: For each alpha-tree generated in step 2, item sets with at least minimum support are mined [13].

FPM algorithm differs from CATS-FELINE in step 2. Instead of recursively constructing alpha trees for each frequent item set, FPM generates a single conditional condensed tree for each item using a pre-order traversal of the original CATS-tree. To illustrate the basic idea behind the algorithm, we will use the database shown in Table 1 as an example and the original CATS-tree constructed from a database scan and its condensed one will look like the following (assuming minimum support of 3) [3]: table 1. This condensed tree, a header table containing all the frequency counts for each item, and the required minimum support will be the actual input to our algorithm called FPM (Frequent Patterns Merge). Given the above condensed tree, FPM starts building an alpha tree for each

frequent item as follows: Since C is an item with the highest frequency in the header table, FPM constructs an alpha tree for c first. By traversing the leftmost path of the tree of Figure 1 in pre order, it will construct a partial tree Figure 2 consisting of a single path C-F-A-M. Note that the order of nodes and the frequency count of some node have been slightly changed from Figure 1 to Figure 2. The node for item c has been moved to the root. Because it is the current alpha item under consideration and the frequency of node F has been changed to 3 from 4 because the branch F-B in Figure 1 does not contain item C and thus the frequency of F has been decremented by 1.

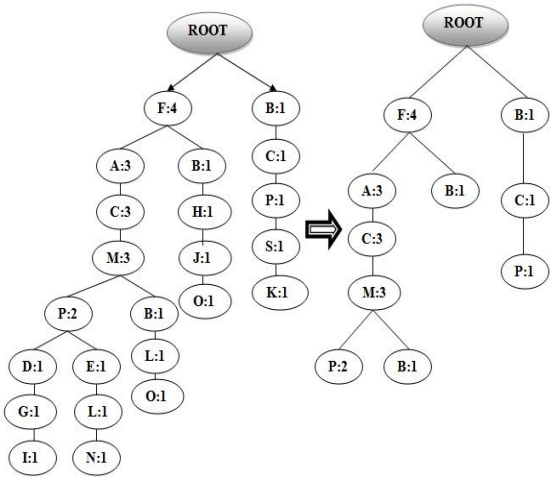


Figure 1: CATS-Tree and its condensed one [13]

After C-F-A-M has been added to the current alpha-tree, the node for „P: 2" will be encountered in the preorder traversal. In this case, P is not frequent and there is no node for P in the current alpha tree. Then, a node is created for „P: 2" and will be inserted to the current alpha tree as a child of the root. This is the major difference between the CATS-FELINE and FPM. FPM attempts to reduce the number of nodes in the alpha tree by condensing even infrequent items. The same process applies to node 'B: 1" and Figure 3 shows the resulting alpha tree after the left subtree of the original condensed tree has been traversed.

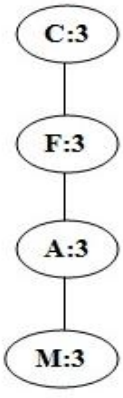


Figure 2: Initial Round of Constructing Alpha Tree for c [13]

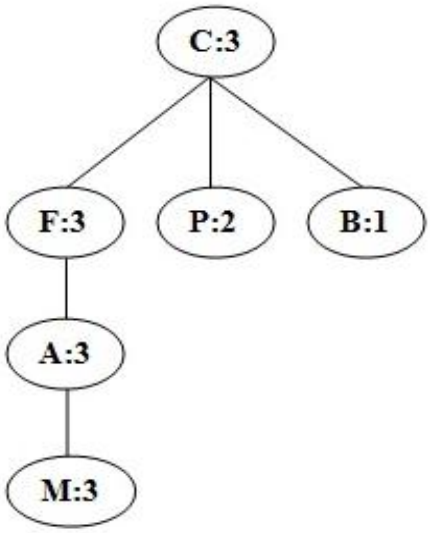


Figure 3: next round of constructing alpha tree for c [13]

Now, the right sub tree of the input tree is to be traversed. It has one node for C and thus the root count of the current alpha tree should be incremented by 1, making it 4. And also the counts of node B and P should also be incremented by 1 respectively. Figure 4 shows the final alpha tree constructed for item C:

F. CATSIM Tree

In the CATS Tree, all the items are stored as they are appearing in the sequence of particular transaction. The sequence may be changed if any of the lower leaf appears more time than the upper leaf. The procedure is continuous up to the last transaction of the database. So, in CATS Tree we cannot predict which item will remain on top of the tree up to

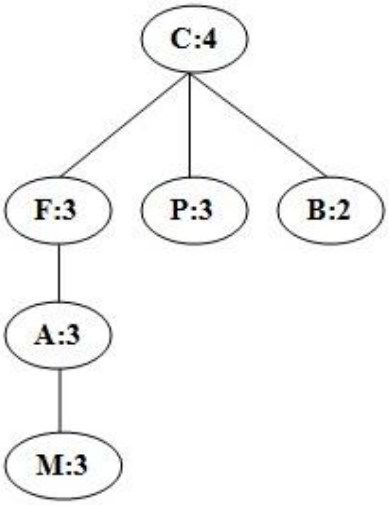
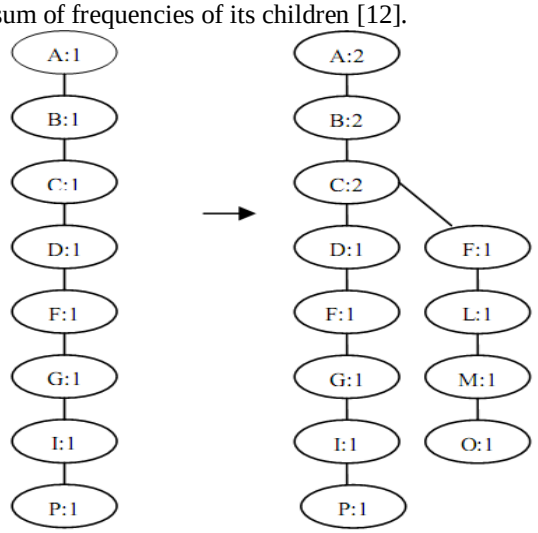


Figure 4: Final Round of Constructing Alpha Tree for c [13]

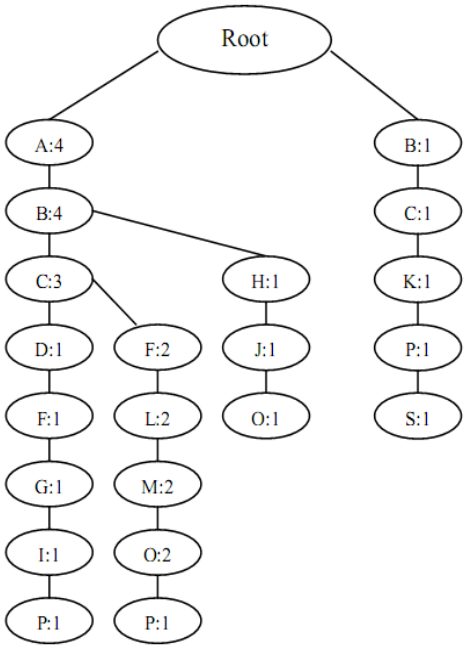
The last transaction. While in the CATSIM Tree all the items are stored either in the alphabetical or any other order as related with items. CATSIM Tree satisfies the following

properties.

1. Items ordering is unaffected by the changes in frequency caused by incremental updates.
2. Node frequency in the CATSIM tree is at least as high as the sum of frequencies of its children [12].



(a) TID - 1 (b) TID - 2



(c) Final CATSIM Tree

Figure 5: CATSIM Tree construction [12]

III. COMPARATIVE ANALYSIS

Figure 6, shows the comparison for the six algorithms for incremental database. The Apriori algorithm requires the maximum execution time. The FP-Growth requires less time compare to Apriori. CATS FELINE is better than CATS and FPM is also better than CATS FELINE. So for incremental size of the database, CATSIM Tree is better than any of the existing algorithms.

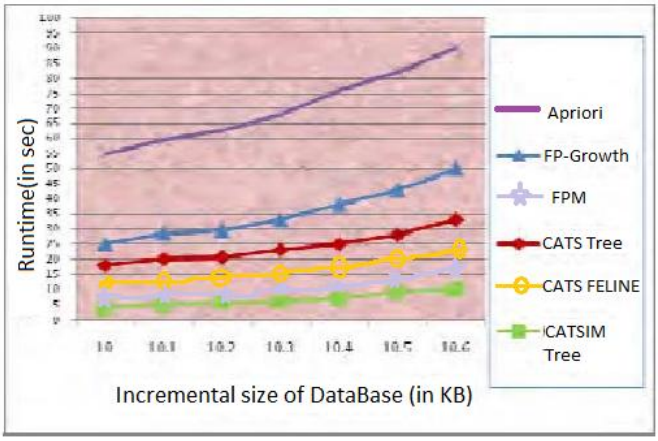


Figure 6: Incremental size of Database vs. Runtime [12]

Experiment for the incremental size of database and memory usage is shown in figure 7. The Apriori, FP-Growth, and CATS are working on the principle of regeneration of the tree, so these three algorithms use the same memory that had been used previously to construct the tree. While in the case of CATSIM Tree, it requires more memory in the normal static database conditions, so also in the incremental size of the database it requires more memory.

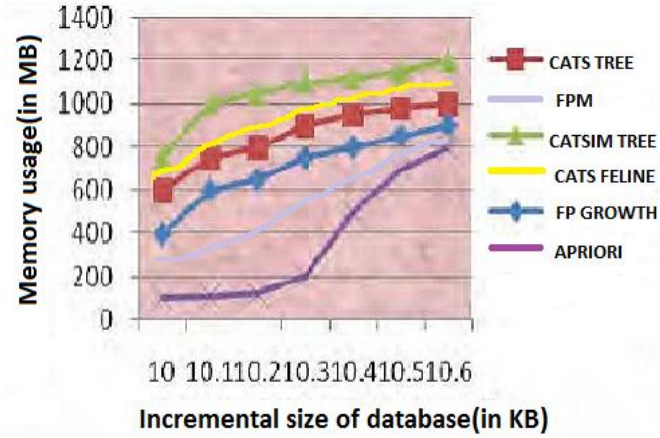


Figure 7: Incremental size of Database Vs Memory Usage [12]

IV. CONCLUSION

Frequent pattern mining is the most important step in association rules which finally helps us in many applications. In this paper the researcher surveyed the pattern mining algorithms. Apriori algorithm uses join and prune method, and major weakness algorithm is producing large number of candidate itemsets and large number of database scans which is equal to maximum length of frequent itemset [5]. A true reason of Apriori failure is it lacks efficient processing method on database and expensive [7]. FP Growth is the best among the three algorithms and is thus most scalable. Eclat performs poorer than FP Growth and the Apriori performs the worst. CATS Tree algorithms allows single pass frequent pattern mining. But it require more memory so to overcome this problem CATS-FELINE is introduce and then FPM. But the tree size can be exponential

for the case of dense data, so there is a need in the improvement in the tree structure which reduces the tree size and make it scalable to handle large database which is highly incremental in nature So, attempt may be made to use concept of CATSIM for heterogeneous databases. In the future improvements must be taken care to enhance the performance of Algorithms for better layout to store the data.

REFERENCES

[1] C.K.-S. Leung,Q. Khan and T. oque, "Cantree: A tree structure for efficient incremental mining of frequent patterns", Proc. 5th IEEE International Conference on Data Mining, (2005), pp.274–281, Los Alamitos, CA.
[2] W. Cheung and O. R. Zaiane, "Incremental mining of frequent patterns without candidate generation or support constraint," in Proc. IEEE Int.Conf. Database Eng. Appl., Los Alamitos, CA, 2003, pp. 111–116.
[3] S. Patel and S. Garg, "Basic Frame work of CATSIM Tree for efficient frequent pattern mining" (In Communication with "Infocomp journal of Computing ", Bazil).
[4] C.Borgelt. "Efficient Implementations of Apriori and Eclat". In Proc. 1st IEEE ICDM Workshop on FrequentItem Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany 2003.
[5] Goswami D.N et. al. "An Algorithm for Frequent Pattern Mining Based On Apriori" (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, pp. 942-947.
[6] Rahul Mishra et. al. "Comparative Analysis of Apriori Algorithm and Frequent Pattern Algorithm for Frequent Pattern Mining in Web Log Data." IJCSIT Vol. 3 (4) , 2012, Pp. 4662 – 4665.
[7] SathishKumar et al. "Efficient Tree Based Distributed Data Mining Algorithms for mining Frequent Patterns" International Journal of Computer Applications (0975– 8887) Volume 10 No.1, November 2010.
[8] Antonie, M.-L. and Zaïane, O. R., "Text Document Categorization by Term Association", IEEE ICDM 2002, pp 19-26, Maebashi City, Japan, December 9-12, 2002.
[9] Cheung, W., "Frequent Pattern Mining without Candidate generation or Support Constraint." Master's Thesis, University of Alberta, 2002.
[10] Muthaimenul Adnan and Reda Alhajj, "A Bounded and Adaptive Memory-Based Approach to Mine Frequent Patterns From Very Large Databases" IEEE Transactions on Systems Management and Cybernetics Vol.41,No. 1,February 2011.
[11] M.El-Hajj and O.R. Zaiane, "COFI approach for mining frequent itemsets revisited," In Proc. ACM SIGMOD Worksjop Res. Issues Data Mining knowl. Discovery, New York, 2004, pp. 70-75.
[12] Ketan Modi and B.L.Pal , "Frequent Pattern Mining using CATSIM Tree", International Journal on Computer Science and Engineering (IJCSE), Vol. 4 No. 09 Sep 2012.
[13] Byung Joon Park, "Efficient Tree-based Discovery of Frequent Itemsets", International Journal of Multimedia and Ubiquitous Engineering Vol. 7, No. 2, April, 2012.
[14] Hitul Patel, Vinit Kumar, Puspak Raval, "Survey: Efficent tree based structure for mining frequent pattern from transactional databases", IOSR Journal of Computer Engineering (IOSR-JCE),Volume 9, Issue 5 (Mar. - Apr. 2013), PP 75-81.

