Efficient Tree Based Structure for Mining Frequent Pattern from Transactional Databases

Hitul Patel[1], Prof. Mehul Barot[2] , Prof. Vinitkumar Gupta[3]
[1]*Department of Computer Engineering, Hasmukh Goswami College of Engineering, Ahmedabad, Gujarat*
[2]*Department of Computer Engineering, LDRP Institute of technology Gandhinagar, Gandhinagar,Gujarat*
[3]*Department of Computer Engineering, Hasmukh Goswami College of Engineering, Ahmedabad, Gujarat*

ABSTRACT

Many different types of data structure and algorithm have been proposed to extract frequent pattern from a large given database. One of the fastest frequent pattern mining algorithm is the CATS algorithm, Which can efficiently represent whole data structure over single scan of the database. We have proposed an efficient tree based structure and algorithm that provide the performance improvement over CATS algorithm in terms of execution time and memory usage.

KEYWORDS: *Frequent Pattern Mining, Association Rule, Minimum Support, Transactional databases.*

I. INTRODUCTION

Frequent Pattern Mining plays a very important role in data mining technique. There are many various studies about the problem of frequent pattern mining and association rule mining in large transactional databases. Frequent pattern mining technique are divided into 2 categories : Apriori based algorithm and Tree structure based algorithm. The apriori based algorithm uses a generate and test strategy for finding frequent pattern by constructing candidate items and checking their frequency count and minimum support against transactional databases. The tree structure based algorithm follows a test only approach. There is no need to generate candidate items sets and test only the minimum support count or frequency count.

In particular, several tree structures have been devised to represent frequent pattern from transactional databases. Most of the tree based algorithm allow efficient incremental mining over single scan of the database as well as efficient insertion or deletion of transaction at any time. One of the fastest frequent pattern mining algorithm is CATS algorithm which can represent the whole data structure and allow frequent pattern mining over single scan of the database. The CATS algorithm enable frequent pattern mining without rebuilding the whole data structure. This paper describe improvement over the original CATS approach in terms of memory usage and execution time. The proposed tree structure allow insertion or deletion of transactions at any time like CATS tree algorithm but usually result in more efficient frequent pattern mining process.This paper is organized as follow : Section II describes some related work and the background knowledge of tree based algorithms. In Section III describe our proposed tree structure, its construction algorithm. Section IV shows comparison with CATS FELINE algorithm. In section V we show experimental result of our proposed algorithm. Section VI shows a conclusion.

II. Related Work

FP tree is one of the oldest tree structure based algorithm that do not use candidate item sets. In FP tree algorithm all items are arranged in descending order of their frequency. After constructing FP tree an iterative algorithm is used for mining the frequent patterns. Then the infrequent items are removed from prefix path of an existing node and the remaining items are regarded as the frequent patterns from transactional databases.
Another tree structure based algorithm is CAN tree algorithm and it construct a tree similar to the FP tree and it also does not require rescan of the original database once it is updated. CAN tree contains all the transactions in some canonical order, so the ordering of the nodes in CAN tree will be unaffected by any major changes like insertion, deletion or modification of the transaction.

CATS tree is a prefix tree. In CATS tree path from the root to the leaves represent set of transactions. After constructing CATS tree from database, it enables frequent pattern mining with different minimum supports without rebuild the whole tree structure.

CATS – FELINE algorithm is an extension of the CATS tree. CATS – FELINE algorithm build a conditional condensed CATS tree in which all infrequent patterns are removed and is different from a conditional FP tree. CATS – FELINE algorithm has to traverse both up and down the CATS tree.

III. Proposed Algorithm

Now we describe our proposed tree structure and it's algorithm. Our main focus is to construct a conditional condensed tree from CATS tree.

Like CATS – FELIENE Algorithm the overall mining process proceeds below :
Step 1 : First convert the CATS tree From database scan.

Step 2 : Now convert CATS tree into condensed tree with nodes having the frequency count less than the minimum support are removed.

Step 3 : Construct Conditional condensed CATS – tree for items with frequency count greater than the minimum support.

Step 4 : For each Conditional Condensed CATS tree from step 3, Item sets with minimum support are mined.
Our Proposed algorithm differ from CATS – FELINE algorithm. CATS-FELINE algorithm construct conditional condensed tree recursively for each frequent item sets. Our proposed algorithm generate a single condensed tree for each item using a pre – order traversal of the original CATS tree.

To understand the basic idea behind the algorithm, we will use the database shown in table (Assume the minimum support is 3).
Now Consider the transactional databases shown in the table 1

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

Table 1 : Transactional databases

Given the above database, the original CATS tree constructed from a database scan and its condensed one are shown in figure

1 **Figure 1(A) : CATS Tree**

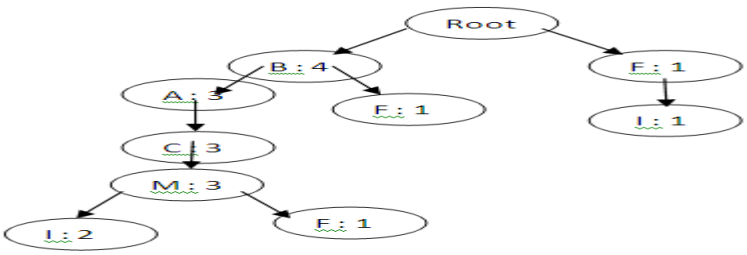
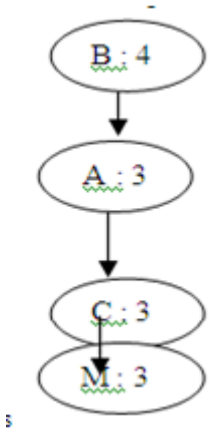


Figure 1(B) : Condensed Tree

Figure 1 : CATS Tree and it's Condensed Tree

This condensed tree, frequency count for each item and the required minimum support will be the input to our proposed algorithm. Given the above condensed tree our proposed algorithm starts building conditional condensed tree for each frequent pattern as follows:From the above figure 1 item B has the highest frequency. Our proposed algorithm construct conditional condensed tree for B first. By traversing the leftmost path of the



tree of figure 1 it will construct a partial tree as shown in figure 2.

Figure 2 : First step of creation conditional Condensed tree

After B-A-C-M has been added to the current conditional condensed tree , the node for (I : 2) will be encountered in the pre order traversal. In the case for node (I : 2), the node I is not frequent and there is no node for I in the current conditional condensed tree. Then a node is created for (I : 2) and will be inserted to the current tree as a child of the root.
The same process applies to node (F : 1) and the figure 3 shows the resulting conditional Condensed tree after the left subtree of the original Condensed tree hase been traversed.

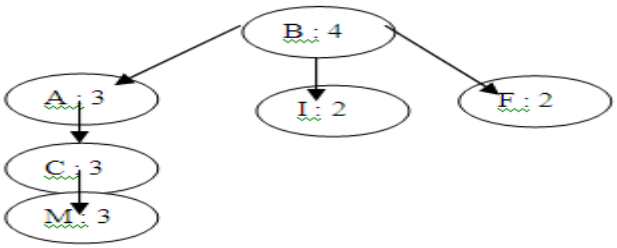


Figure 3:Second step of creation conditional Condensed tree

Now traverse the right sub tree of the condensed tree. It has one node for I and F so it should be placed as a child of the root. So the count of the node I and F should be incremented by 1 respectively. Figure 4 shows the Final conditional condensed tree for the above databases.

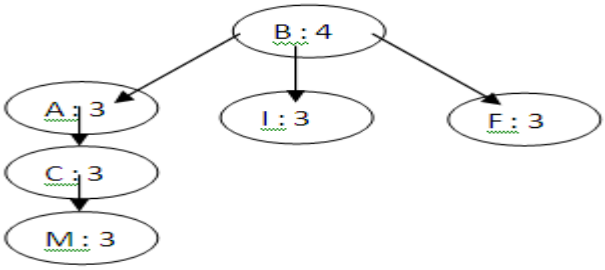


Figure 4 : Final step of creation conditional Condensed tree

IV. COMPARISON WITH CATS – FELINE ALGORITHM

Given the same database, conditional condensed tree constructed by CATS – FELINE and our proposed algorithm will be different as shown in the figure 5. The major difference is due to the way how the infrequent items are dealt with. From the figure CATS - FELINE algorithm keeps many separate node (I:2 and I:1) for infrequent items such as node I. Hence it needs more memory space for storing infrequent items. In our Proposed algorithm separate infrequent items are collapsed into single child nodes of the root.

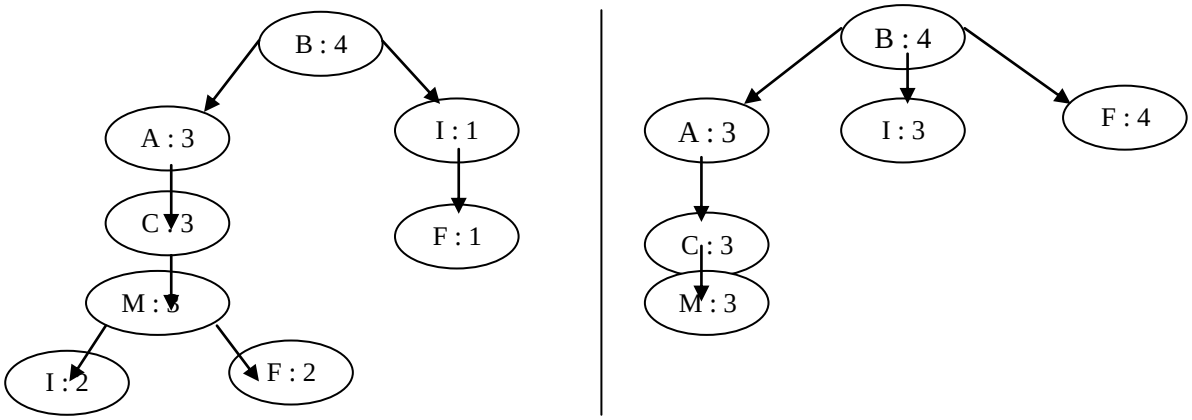


Figure 5 : Comparison with CATS – FELINE algorithm

V. Experiment Result

We have downloaded datasets from http://fimi.ua.be/data website (Frequent Pattern Itemeset Mining Implementations). Program are written in Microsoft Visual Studio 10 and run on the Windows 7 operating system on a 2.13 GHz machine with 4 GB of main memory. Table – 3 and Table – 4 shows total execution time required to extract all frequent patterns for Chess datasets and retailers dataset respectively by our proposed data structure and CATS – FELINE data structure for different minimum support.

Minimum Support	CATS – FELINE ALGORITHM	Our Proposed Algorithm
0.90	29	15
0.70	22.5	10
0.50	18.2	12
0.30	14.2	8
0.25	8.2	2.3

Table 2 : Execution time for Accident database for different Minimum Suppport

Minimum Support	CATS – FELINE ALGORITHM	Our Proposed Algorithm
0.90	5200	3700
0.70	5400	4200
0.50	6500	4900
0.30	7200	5200
0.25	7800	5400

.

Table : Execution time for Retailer database for different Minimum Suppport

VI. CONCLUSION

In this Paper, We have described an efficient tree based structure for representing a conditional condensed tree for mining frequent patterns from transactional databases. Performance improvement of our proposed algorithm over CATS – FELINE algorithm by conducting various minimum support values and

different sizes of databases.Our Proposed Algorithm reduces the overall processing time to extract frequent pattern from transactional databases compare to CATS – FELINE algorithm. It also support iterative mining like CAN tree and FP tree means it can extract the frequent patterns without rescan the original database

Acknowledgement

I forward my sincere thank to Prof. Mehul P Barot for there valuable suggestion during my desertion. His suggestion are always there whenever I needed it.

REFERENCES

[1] Muthaimenul Adnan and Reda Alhajj, "A Bounded and Adapative Memory-Based Approach to Mine Frequent Patterns From Very Large Database" IEEE Transactions on systems Management and Cybernetics- Vol.41,No, 1,February 2011.

[2] W.cheung and O. R. Zaiane, "Incremental mining of frequent patterns without generation or support constraints," in proc. IEEE Int.Conf. Database Eng. Appl., Los Alamitos, CA, 2003, pp. 111-116

[3] C. K.-S. Leung, Q.I. Khan, and T. Hoque, "Cantree: A tree structure for efficient incremental mining of frequent patterns," in Proc. IEEE Int.Conf.Data Mining, Los Alamitos, CA, 2005,PP. 274-281.

[4] C. K. –S. Leung. Interactive constrained frequent- pattern mining system.In Proc.IDEAS 2004, PP. 432-444.

[5] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", Proc. Of the 20th Very Large Databases International Conference, (1994), pp. 487-499, Santiago, Chile.

[6] J. Han, J. Pei, Y. Yin and R. Mao, "Mining frequent patterns without candidate generation: a frequent pattern tree approach", Data Mining and Knowledge Discovery,(2004)

[7] B. Rácz, "nonordfp: An FP-growth variation without rebuilding the FP-tree," in Proc. FIMI, 2004

[8] C.K.-S. Leung. Interactive constrained frequent-pattern mining system.In Proc.IDEAS 2004, pp. 49–58.

[9] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee, "CP-Tree Structure for Single-Pass Frequent Pattern Mining." In: Springer-Verlag Berlin Heidelberg 2008,pp 1-6

[10] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", Proc. of the 20th Very Large Data Bases International Conference, **(1994)**, pp. 487-499, Santiago, Chile.

[11]W. Cheung, and O. Zaiane, "Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint", Proc. of 7th International Database Engineering and Applications Symposium, **(2003)**, pp. 111–116, Los Alamitos, CA.

[12] Agrawal, Rakesh; Srikant, Ramakrishnan: "Fast Algorithms for Mining Association Rules." Proc. 20th Int Conf. Very Large Data Bases, VLDB, 1994.