IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 5 (Mar. - Apr. 2013), PP 75-81
www.iosrjournals.org

Survey: Efficent tree based structure for mining frequent pattern from transactional databases

Hitul Patel[1], Vinit Kumar[2], Puspak Raval[3]
[1]Department of Computer Engineering, Hasmukh Goswami College of Engineering, Vehlal, Gujarat
[2]Department of Computer Engineering, Hasmukh Goswami College of Engineering, Vehlal, Gujarat
[3]Department of Computer Engineering, DAIICT, Gandhinagar, Gujarat

Abstract: *Different types of data structure and algorithm have been proposed to extract frequent pattern from a given databases. Several tree based structure have been devised to represent the data for efficient frequent pattern discovery. One of the fastest and efficient frequent pattern mining algorithm is CATS algorithm which represent the data and allow mining with a single scan of database. CATS tree can be used with incremental update of the database. Transaction can be added or removed without rebuilding of the whole data structure.*

Keywords *– Frequent Pattern Mining, Transactional Databases, Minimum Support, Itemsets.*

I. Introduction

One of the major function of association rules is to analyse large amount of databases. Association rules have been applied to many areas including classification, clustering, detection. The mining process can be broken down into frequent itemsets and the generation of association rules. Association rules is an iterative process. Frequent Pattern Mining plays a very dominant role in data mining. There are many various application such as text, transaction database and image processing to generate a large volume of data[3,9].

For frequent pattern mining the algorithm are divided into two categories 1) Apriori algorithm, 2) Tree structure algorithm. The apriori algorithm is an influential algorithm for mining frequent item sets for association rules. The apriori based algorithm uses a generate and test strategy approach to find frequent pattern by constructing candidate items and checking their counts and frequency from transactional databases. The tree structure algorithm uses a text only approach. There is no need to generate candidate items sets[7]. Several tree based structure have been devised to represent the data for efficient pattern discovery. Most of the tree based structure allow efficient mining with single scan over the database. One of the fastest and efficient frequent pattern mining algorithm is CATS algorithm[10]. The CATS algorithm enable frequent pattern mining with different support value without rebuild the whole structure. In addition CATS tree allow removal of transaction concurrently. Although a large database can be processed by CATS tree if out – of – date transaction are removed concurrently. CATS-FELINE uses a divide and conquer method to generate frequent pattern without generating candidate item sets. In order to ensure that all frequent pattern are captured by FELINE, FELINE has to traverse both up n down to include all frequent items. While building conditional condensed CATS tree, items are executed if the items are mined or if the items are infrequent.

II. Literature Survey

2.1 FP - Tree(Frequent Pattern Tree)

Frequent Pattern Tree is one of the oldest tree structure approaches that don't use candidate set generation adopted by apriori based algorithm. From a given set of transactional data it first construct a tree structure in which all items are arranged in descending order according to their frequency.After construction of FP tree the frequent pattern can be mined using an iterative algorithm which looks up the header table and select the item with minimum support[6]. Then the infrequent items are removed from existing node and remaining items are reordered as the frequent item-sets at the specified items.The items that does not meet the minimum threshold has been eliminated.

Consider the transactional database shown in Table 1 with 5 transactions.

Tran. ID	Items
T1	A,B,D,E
T2	A,C,D
T3	E,F,H,I
T4	A,B
T5	C,E,F

Table 1. Example of Transactional Database [7]

The frequent itemlist for the above database is given in Table 2.

Items	Count
A	3
B	2
C	2
D	2
E	3
F	2
H	1
I	1

Table 2. Frequent Itemlist for the Transactional Database in Table1[7]

The items that does not meet the minimum threshold has been eliminated. The frequent itemlist that support the minimum support threshold is given in Table 3.

Items	Count
A	3
E	3
B	2
C	2
D	2
F	2

Table 3. Frequent Itemlist for the Transactional Database that Support Minimum Threshold [7]

The transactional database according to the frequent item list is given in Table 4.

Tran. ID	Items
T1	A,E,B,D
T2	A,C,D
T3	E,F
T4	A,B
T5	E,C,F

Table 4. Sorted and Eliminated Transactions of the Database in Table 1 [7]

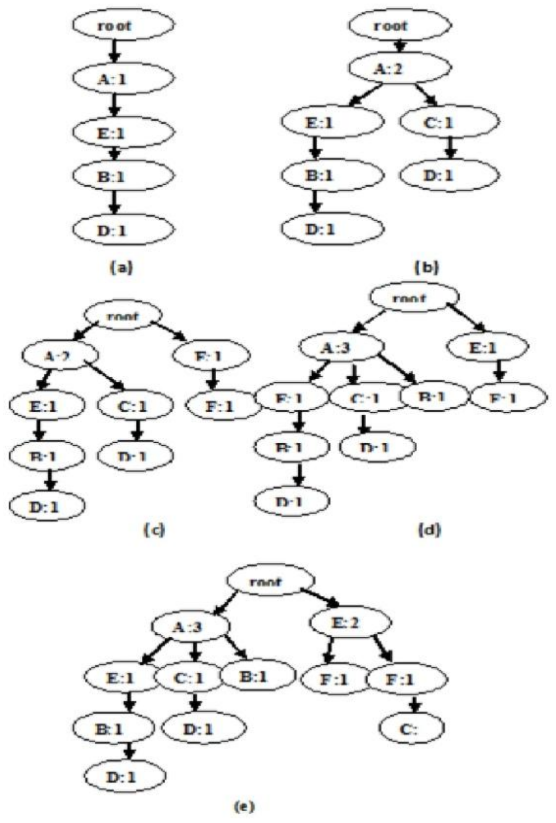
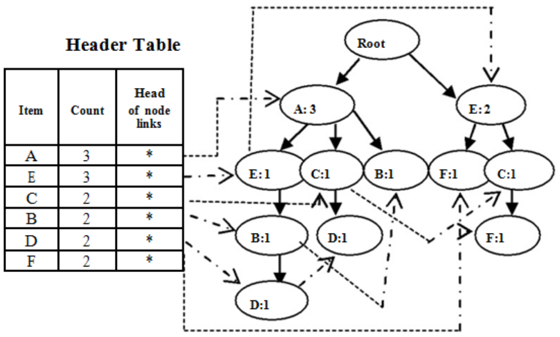


Figure 1. Steps in creating the FP – Tree [7]



Item	Count	Head of node links
A	3	*
E	3	*
C	2	*
B	2	*
D	2	*
F	2	*

Figure 2. FP-Tree with Header Table [7]

Consider the itemset D. Its prefix paths are {((A,E,B):1),((A,C):1)}. After removing the infrequent items,(A:2).So the frequent itemset for D is A.

2.1.1 *Advantages*

The main advantage of FP tree is that it does not generate any candidate items. It has only 2 passes over the data sets. FP tree is much faster then Apriori algorithm. FP tree uses a compact data structure. Another advantage is that it eliminate repeated database scan.

2.1.2 *Disadvantages*

The main disadvantage is that FP tree may not fit into memory. Another Disadvantage is that FP tree is expansive to build. FP tree suffer from temporal locality issues.

2.2 CAN Tree (Canonical Tree)

A CAN tree that arrange the nodes of a tree in some canonical tree. CAN tree generate a tree structure similar to the FP tree algorithm. Compare to FP tree it does not require a rescan of the original database once it is updated[3]. CAN tree contains all the transaction in canonical order hence the order of the node in CAN tree will not unaffected by any incremental updates like insertion, deletion and modification of the transactions.

Consider the following database,

		TID	Items
DB	Original database	T1 T2 T3	{a,c,d,g} {b,c,d,e} {b}
DB1	1st group of insertions	T5	{a,e,f}
DB2	2nd group of insertions	T6 T7	{b,c} {a}

Table 5. Example of Transactional Database [7]

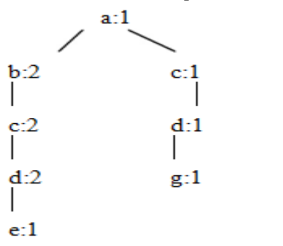


Figure 3. Initial CAN-tree[7]

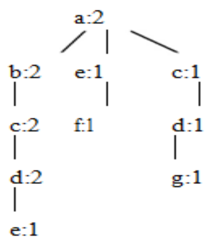


Figure 4. CAN-tree after 1st group of insertions[7].

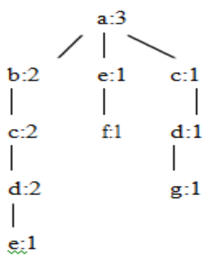


Figure 5. CAN-tree after 2nd group of insertions.[7]

2.2.1 Advantages
The main advantage is it support incremental support without any changes in the tree structure. There is no need to rescan the database when it is updated.

2.3 COFI Tree
COFI tree is much faster than the FP Growth tree and it require significantly less memory. The basic idea of COFI tree is to build a projections from the FP tree each corresponding to sub-transactions of items with a given frequent item. The COFI tree algorithm uses a top down approach where it's performance shows to be severely affected while mining databases that has potentially long candidate pattern that turns to be not frequent[7].

The basic idea behind COFI tree is simple and is based on maximal frequent patterns. A frequent item set X is said to be maximal if there is no frequent item set X' such that X X'. Maximal frequent pattern are a relatively small subset of all frequency item sets.

Let us assume that In Oracle we knows all the maximal frequent item sets in a transactional database so deriving all frequent item sets becomes trivial. The Oracle does not exist but propose a pseudo-oracle that discover this maximal frequent pattern using the COFI tree and we derive all item sets from them.

Consider   the following database,

D	E				
A	B	D	F	E	H
A	G	D	E	C	B
A	G	D	F	E	I
A	G	E	B	F	C
A	D	H	E	F	
A	G	D	B	L	C
A	B	C	F		
A	D	B	C	G	
A	F	B	C	E	
A	B	C	H		

Table 6.Example of Transactional database [7]

A	B	C	D	E	F	G	H
10	8	7	7	7	6	5	3

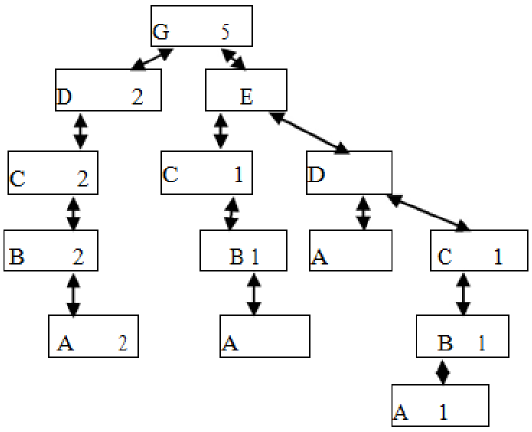


Figure 6. Creation of COFI Tree [7]

2.3.1 Advantages
This method is advantageous because the size of the generated candidate item list is minimized. This algorithm built and efficient mined one at a time making memory significantly small.

2.3.1 Disadvantages

It is disadvantageous because maximum effort is required for minimization.

2.4 CATS Tree

CATS tree is an extension of FP tree. CATS tree[7] is a prefix tree and it contains all element of the FP tree including the header and the item links. To improve storage compression and allow frequent pattern mining without generating candidate itemsets we use CATS-tree algorithm. CATS tree algorithm enable frequent pattern mining with different support without re-building the tree structure. This algorithm allow single pass over the database. It also allow efficient insertion and deletion of transaction at any time. In other word transaction can be added to or removed from the tree at any time. CATS tree contains all items in every transaction. In CATS tree ordrer of the items within paths from the roots to leaves are ordered by local support.The main properties of CATS tree is it's Compactness. Compactness measures how many transactions are sharing a node. It is possible to create a CATS tree with maximum compression.

		TID	items
DB	Original database	T1 T2 T3	{a,c,d,g} {b,c,d,e} {b}
DB1	1st group of insertions	T5	{b,e,f}
DB2	2nd group of insertions	T6 T7	{b,c} {b}

Table 7. Example of Transactional Database [7]

The steps in the construction of CATS tree is illustrated in Figure 7.

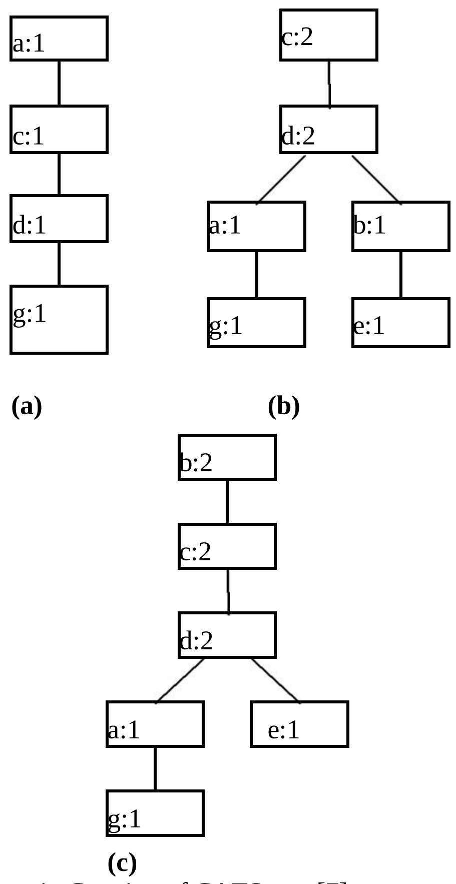


Figure 7. Steps in Creation of CATS tree [7]

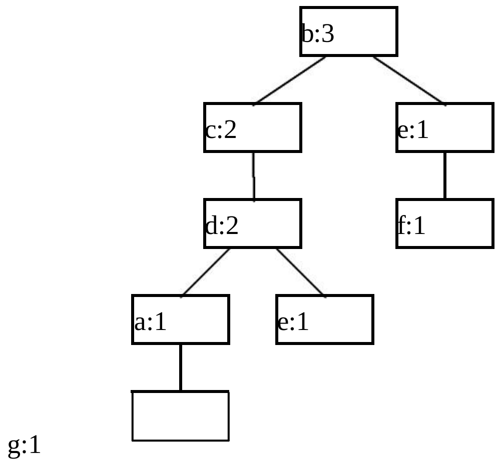


Figure 8. CATS tree after 1st group of insertions [7]

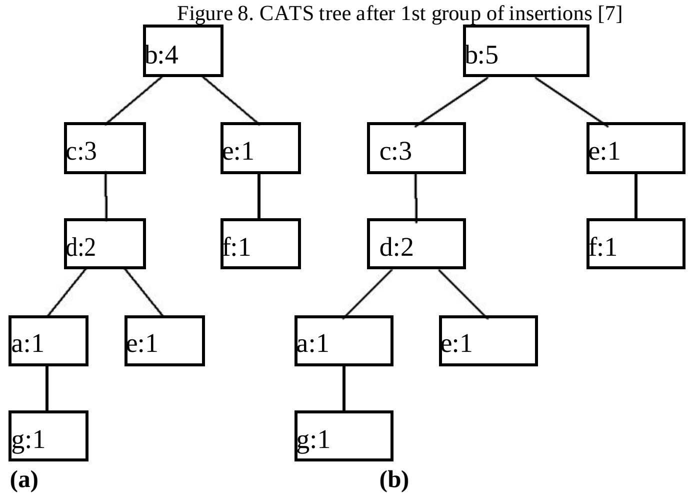


Figure 9. CATS tree after 2^{nd} group of insertions[7]

After constructing the CATS tree, the frequent patterns can be found by following the frequency of an item by considering its both upward and downward paths.

2.4.1 Advantages

The main advantage is that the database is scan only once and the tree are reordered according to their local frequency in the paths. The disadvantage is that a lot of computation are required to build the tree structure. CATS tree algorithm allow addition and deletion of transaction in a single transaction. CATS tree algorithm is suitable for frequent pattern mining where modification and frequent pattern are common.

2.4.2 Disadvantages

The main disadvantage is that it require more computation to build a tree.

2.5 CATS – FELINE

CATS – FELINE algorithm employs divide and conquer method to generate frequent pattern without generating candidate itemsets. This algorithm partititon the data based on what pattern transacrions have.
CATS – FELINE algorithm builds a conditional condensed CATS-tree for each frequent item p by gathering all transactions that contain p. In conditional condensed CATS- tree all infrequent items are removed and is different from FP tree. All Frequent pattern are captured by CATS-FELINE, it has to traverse both up and down the CATS tree.

2.5.1 Algorithm for a Conditional Condensed Tree

Like CATS-FELINE the mining process proceeds in three steps:
Step 1: First convert the CATS tree generated from scanning a database into condensed tree. All the nodes having the minimum support are removed from the tree.

Step 2: Now construct condtional condensed CATS-tree for items in the header table having greater than the minimum support.
Step 3: For conditional condensed CATS-tree generated from step 2, items with atleast minimum support are frequently mined.

III. Conclusion and Prospect

In this paper it is summarised that various data structure that can be used to extract frequent pattern from a large database. We have discussed about the structure like CAN tree, FP tree, CATS tree with their advantages and disadvantages. Among the discussion on the above structure CATS tree can be considered to optimal because it scans the database only once and transaction can be added or removed from the tree at any time. As well as CATS tree may be suitable for any incremental update in the database. This makes CATS tree algorithm suitable for real time transactional frequent pattern mining where frequent pattern are common.

It can be conclude that some research direction for mining frequent pattern from a transactional databases by existing work in future. As the future scope, the number of database scan, execution time, memory usage may still be decrease by investigating other kind of algorithm.

Acknowledgment

I would be thankful to my guide assistant professor Vinitkumar Gupta here for fervent help when I have some troubles in paper writing. I will also thank my class mates in laboratory for their concern and support both in study and life.

References

[1] 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.
[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] 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 Minig, Los Alamitos, CA, 2005,pp. 274-281
[4] 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
[5] B. Rácz, “nonordfp: An FP-growth variation without rebuilding the FP-tree,” in Proc. FIMI, 2004
[6] J. Pei, J. Han, and L.V.S. Lakshmanan.“Mining frequent itemsets with convertible constraints”, In Proc. ICDE 2001, pp. 433–442
[7] K. Sasireka, G.Kiruthinga, K.Raja, “A survey about Various Data Structures For Mining Frequent Patterns from Large Database,” International Journal of Research and Reviews in Information Sciences(IJRRIS) Vol. 1, No. 3, September 2011, ISSN: 2046-6439.
[8] C.K.-S. Leung. Interactive constrained frequent-pattern mining system.In Proc.IDEAS 2004, pp. 49–58.
[9] 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.
[10] 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.
[11] Agrawal, Rakesh; Srikant, Ramakrishnan: “Fast Algorithms for Mining Association Rules.” Proc. 20th Int Conf. Very Large Data Bases, VLDB, 1994.