ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 2, August 2013

A New Approach for Discovering Frequent Pattern from Transactional Database

Minu Pandya, Priyanka Trikha
Computer science, Rajasthan Technical University

Abstract: Frequent pattern mining is a heavily researched area in the field of data mining with wide range of applications. Finding a frequent pattern (or items) play essentials role in data mining. Efficient algorithm to discover frequent patterns is essential in data mining research. A number of research works have been published that presenting new algorithm or improvements on existing algorithm to solve data mining problem efficiently. In that Apriori algorithm is the first algorithm proposed in this field. By the time of change or improvement in Apriori algorithm, which compressed large database in to small tree data structure like FP tree, CAN tree and CP tree have been discovered. In CP tree, like FP tree it contains frequent and non frequent items at the mining time. So item required to extract frequent pattern is more. In this paper I propose a new novel tree structure - extension of CP tree that extract all frequent pattern from transactional database using CP-mine algorithm. So at the mining time it contains only frequent patterns. In my proposed algorithm it mine frequent item set from our proposed tree structure by using pruning tree and marked value technique. This tree structure constructs compact prefix tree structure with one database scan and it provide same mining performance as FP growth technique by efficient tree restructuring process. My proposed tree structure support interactice mining and incremental mining without rescan the original database.

***Keywords*-frequent pattern,association rule,data mining,data stream,CP-mine**

I. INTRODUCTION

Data mining is a powerful new technology with great potential to help companies focus on the most important information in the gathering data. It discovers some related information within data that queries and reports can't effectively reveal. Data mining performing interesting knowledge, regularities or high level information can be extracted from databases and viewed from different angel. This discovered knowledge can be applied to decision making control, information management and query processing [13]. A procedure called mining frequent item sets is a foundation step of association rules discovering. The problem of mining association rules- and the more general problem of finding frequent pattern- from large database has been subject of numerous studies [8]. In general first algorithm attempt to mine association rule is Apriori algorithm that depend on generate and test paradigm. They compute frequent pattern by generating candidate and checking their frequencies (i.e. support count) against the transactional database. To improve the efficiency of mining process, Han et. Al proposed an alternative framework [6], namely a tree based on the framework the algorithm they proposed in this framework to construct an extended new tree data structure , called Frequent pattern tree that capture the content of the transactional database rather than employing the generate-and-test strategy of Apriori algorithm, such tree based algorithm focus on frequent pattern growth-which is restricted test-only approach. And some tree based incremental mining algorithm were developed that is Cheung and Zaiane [11] proposed FELINE algorithm with CATS tree and where KOH and Shieh[12] proposed AFPIM algorithm. The former aim to make CATS tree (a variant of FPtree) compact and proposed for incremental mining, while FELINE algorithm well suited for interactive mining. So, it requires two database scan. CATS tree [8] is a single-pass solution but it still suffers from complex tree construction process. The above two limitation are well addressed in CanTree [3] ,that captures the complete information in a canonical order of terms from database into a prefix-tree structure in order to facilitate it for incremental and interactive mining using FP-growth mining technique . Since in the Can tree all items are arranged in canonical order so it usually yields poor compaction in tree size compared to FP-tree [3]. In CP-tree[1] , it construct compact tree data structure with one database scan and provides the same mining performance as the FP-growth technique by efficient tree restructuring process. CP-tree keeps all the items in the tree after restructuring phase. In this paper, we propose novel new tree data structure which is an extension of CP-Tree. Unlike CP tree, a new tree contains only frequent items in the tree at the mining time like FP-Tree but still extract frequent item set from database with only one database scan like CP-Tree. In our proposed tree structure use CP-mine algorithm for mining frequent pattern still it give better performance compared to FP-growth algorithm.. The main aim of CP-mine algorithm is to improve an efficiency of mining frequent pattern by using pruning and tree and marked value technique [14]. The organization our paper is as follow: In section II we put our idea of proposed data structure in detail. In section III showed several experiment on real and synthetic dataset with different support value for CP tree and our proposed tree structure.

II. RELATED WORK

In this section key concept and terms widely used in frequent pattern mining explained. In this section we discuss some tree data structure like FP Tree,Can Tree and CP tree and also discuss some issue related to that. In first known proposed method for extra frequent pattern is Apriori algorithm proposed by Agrawal et al [7]. There have been developed enormous modified versions to improve it]. Since the main drawback for Apriori-based algorithms was

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 2, August 2013

involving multiple database scans and generation of a large number of candidates. Han et al [6] proposed new algorithm, FP-Growth to extract frequent pattern. In this approach the number of database scans was reduced from K to just two times without candidate and test generation by using this approach the number of database scans was reduced from K to just two times without candidate generation. So far there have been conducted enormous studies on improving this algorithm efficiency. In [11] a tree structure called CATS is proposed to mine frequent patterns in an incremental manner. The proposed tree structure improves FP-tree on data compression and allows extracting frequent patterns without the need to generate a set of candidate items. According to this method, the first transaction in database is added to the tree roots. And for subsequent transactions, the item within the transaction are compared with the items in the tree for identify shared items. If there is any item in common between tree nodes and the transaction then transaction is merged with the node that has the highest frequency level. Then, the remainder of the transaction is added to the merged nodes. This process is recursively repeated until all common items are discovered In [12] authors proposed the AFPIM algorithm support incremental mining. Like Fp tree it keeps only frequent items. In this algorithm, a threshold called PreMinsup is considered whose values are set less than the Minsup. Since, items are ordered based on the number of events, the insertion, deletion or modification of transactions may affect the frequency and order of the items. More specifically, items in the tree are adjusted when the order of the items changes. The AFPIM algorithm swaps such items by applying bubble sort algorithm that involves huge calculation. In [3] authors put forward a new tree structure called Can-Tree. Can-tree algorithm is used for incremental mining and needs only one database scan. In Can tree, all items arranged in some canonical order which can be determined by the user. Therefore, any changes in frequency, which is caused by incremental updates (such as insert, delete, or modify transaction(s), will not affect the order of the item in the Can-tree. Therefore, new transactions are inserted into the tree without swapping any tree nodes. In [2] a new tree structure called CP-tree is put forward. CP-tree is a new prefix tree data structure which can be used for interactive and incremental mining. In this, all transactions are inserted into the tree in accordance with a predefined item order . This item order by a list, called I-list. After inserting some of transactions, if the item of the order I-list differs from the current frequency-descending item order to a predefined degree so, the CP tree restructured using branch sorting method. Then the item order is updated with current list current list.

III. OVERVIEW OF PROPOSED DATA STRUCTURE

In this section, we introduce a new tree data structure which is an extension of CP-Tree. We have seen that CP-Tree contains all the items (frequent and also non-frequent) in the tree at the mining time. We introduce a new tree which contains only frequent items at the mining time like FP-Tree but still extract frequent item sets from database with only one database scan like CP-Tree with CP-mine algorithm. In FP growth algorithm when mining a long frequent item set on large database, the algorithm is significantly outperforms the Apriori algorithm. When we keep only small portion of candidate item sets becomes frequent item sets then generating FP tree which is very costly .In CP-mine it extract frequent item set from our proposed tree structure by using pruning tree and marked value technique.

A. Proposed tree construction

Like a CP tree, our new proposed structure tree structure contain two phase. (i) Insertion Phase and (ii) Restructure Phase. In first phase it scan all transaction(s) , inserted in to tree according to current item order of I-list and update the frequency count of respective items in I-list. Next phase is rearrange I-list according to frequency descending order of items and restructure a tree nodes according to new I-list , But our proposed data structure like FP tree ,it contain only frequent items in restructure phase . In this restructure phase it sorted items uses Branch Sorting Method (BSM). It is an array-based technique that performs the branch-by-branch restructuring process from the root of T. In BSM it sorts each path in the branch according to the new sort order by removing path from the tree it sorting in to temporary array and again inserting it in to the tree. However, while processing a path, if it is found path which is already in sorted order then that path is skipped and move to the next path. And finally restructuring mechanism is completed when all the branches are processed which produces the final Tsort. Here theses two phase are dynamically executed in alternate fashion, starting with insertion phase the first part of DB and end with the restructuring phase at the end of DB.

TID	Transaction
1	c,a,e
2	b,d,f
3	a,b,d
4	b,d,e

(a) Transcational Database

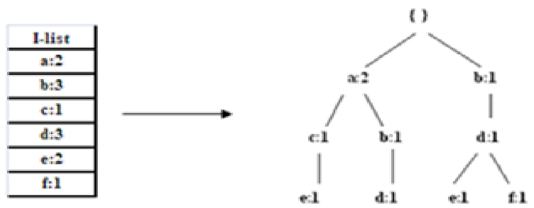
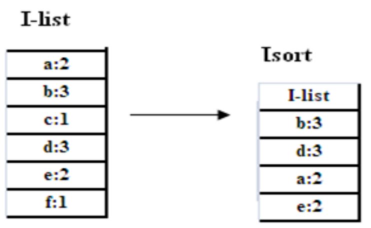


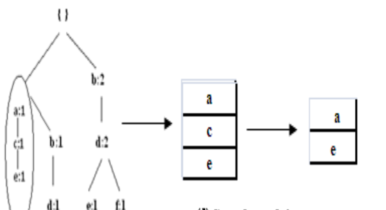
Fig. 1 I-List and Tree after Insertion Phase for Database

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 2, August 2013

As shown in fig 1. (a) First items inserted in to tree which are not in frequency descending order and construct tree. In restructure phase I-list sorted according to frequency descending order and generate new I-list that contain only frequent items like FP tree which shown in fig (b).

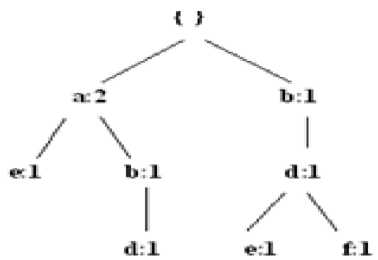


(b)Restructure list to Construct I_sort

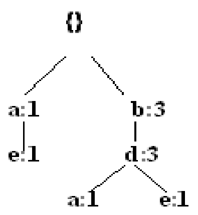


(d) Sort the path in teporary array

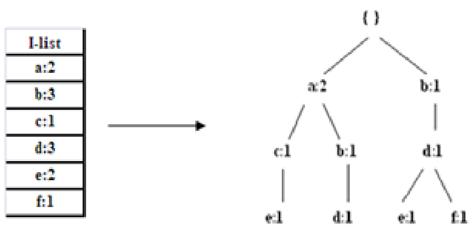
(c) Remove unsorted path



(e)insert sorted path in to tree



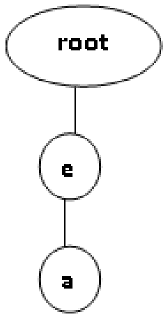
(f)Final structure of our proposed tree



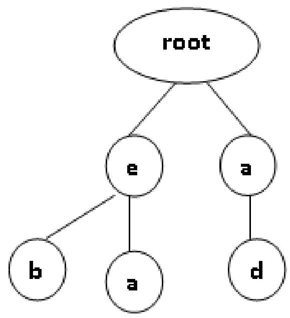
(g) Tree structure of CP-tree of given Database DB

Fig.2 Total Processing Of Restructure Phase of My Proposed Data Structure

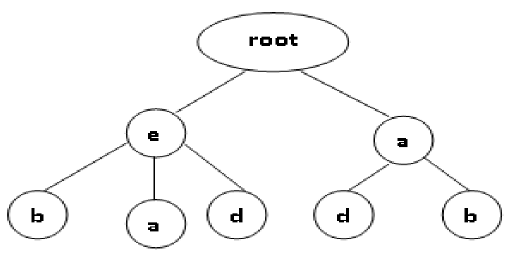
In tree, restructuring phase start with the first branch that is leftmost branch. From the first path {a:1?c:1?e:1} of the branch which is un sorted path is removed from the tree. And sorted it in temporary array to the order {a:1?e:1} that satisfying I sort order then again inserted in to tree, branch which is already in sorted is skipped and move to next branch as shown fig (c),(d),(d1).After the construction of the tree, it mine frequent pattern using CP-mine algorithm. This algorithm is variation of FP growth algorithm. This proposed algorithm also discovers entire frequent item set from the treelike FP tree. But in CP-mine does need to recursively construct new tree during mining frequent item set. CP-mine algorithm determine frequent item sets from only one tree that is our proposed tree structure, where FP tree has to construct many (sub)trees. We use marked Value and pruning tree to trim irrelevance sub tree and infrequent item sets. Here we show proposed mining algorithm. In proposed mining algorithm mine frequent item set starting from (n-1)th item set and assume that item set as new root. We then examine every path under this root. This root recursively determined for frequent item sets and this step repeated until first item in header table. Here as shown fig 2. We put only all non-frequent items in pruning item starting from bottom repeat until it reach the first item .as shown in fig.3 marked value is performed in order to trim some sub tree. Let us consider item set bd is frequent (because is support is 3), so the algorithm nee d to recursively performed. Here as shown in fig.4 marked value of item d which is child of b is et to 1 and considering a prefix d in the tree. So,it only traverse node which is set to 1 , instead of traversing all sub trees.



(a)



(b)



(c)

Fig. 3 Pruning Tree

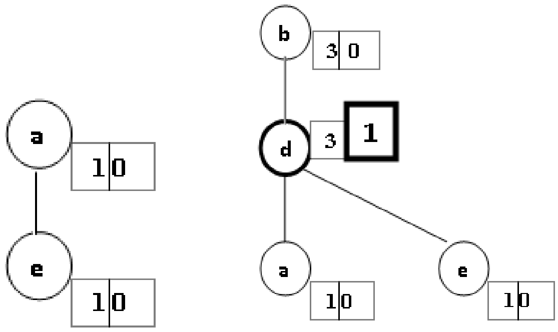


Fig. 4 the Marked Value indicate the Sub trees in Propose tree

B. Proposed data structure support

Incremental Mining:

The main goal of the incremental mining is to provide environment so that all frequent patterns can discovered when DB constantly updated, such that when new trnscat6ion(s) are added and\or old transactions are removed from, the pattern can be mined without restarting the mining process from the scratch. Here we save the tree after inserting all items in I-list so when adding new transaction(s) or remove old transaction(s) form the save we easily performed according to that we restructure tree in restructure phase without need to rescan the database.

Interactive Mining

The main goal of interactive mining that once tree constructed it can change support value without rebuilding tree or rescan database. Here in our proposed tree structure it save the tree after inserting all transactions. So if user easily changed support value and according to that it restructure tree without rebuilt tree or rescan the database, like FP tree .We have support of all items on our I-list , so we can keep only frequent items according to different minimum support without rescan the database.

IV. EXPERIMENT RESULTS

In this section, the performance of our proposed structure evaluated by comparing with it CP tree structure. All experiments are performed on a 3.0 GHz Pentium VI pc machine with 1 GB memory and 40 GB hard disk running on a MS window XP professional. Both tree structures are implemented in java (jdk6).Here we perform experiments on two kind of dataset, synthetic and real world dataset. Here we take both database from Item set Mining Dataset Repository.

Table 1.Total time require for execute Mushroom Dataset

Support value	CP-Tree	My proposed Tree
0.2	23000	2406
0.25	23782	2015
0.3	23047	1719
0.35	23360	1594

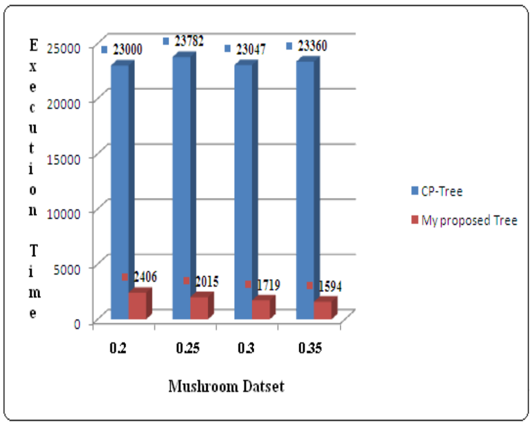


Fig.5 Performance study of CP-tree and Proposed tree Mushroom Dataset

In first shown total time required to extract frequent pattern for Mushroom dataset from real dataset for CP tree and proposed data structure with respective different support value Here as shown in table 1 total time require to extract frequent items in CP tree and proposed tree date structure. In fig.5 shows performance study for both, as shown in table 2,the time require to execute CP tree is almost four times higher than our proposed data structure.

Table 2.Total Time Require For Execute T10I4D100K Dataset

Support value	CP-Tree	My proposed Tree
0.015	386188	188094
0.02	391407	206172
0.025	381000	218969
0.03	380579	199125

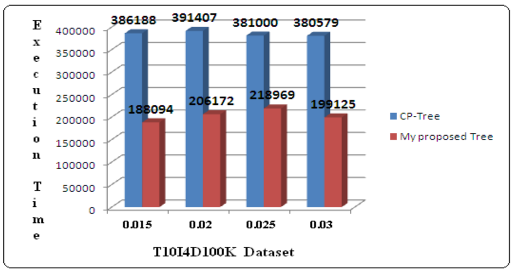


Fig.6 Performance study of CP-tree and Proposed tree for T10I4D100K Dataset

Similarly we have shown in fig.6 performance study for T10I4D100k synthetic dataset. In that we take different support value for CP tree and proposed data structure and calculate total execution time, which represent in Table 2.The time require to extract frequent pattern in proposed data structure less than CP tree. in fig.7, it shows time require to construct both proposed tree and CP tree for different real database**.** In that total construction time includes both insertions phase and restructure phase of both

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 2, August 2013

tree. in fig.8 shows total execution time require extracting frequent pattern CP tree and proposed data structure.

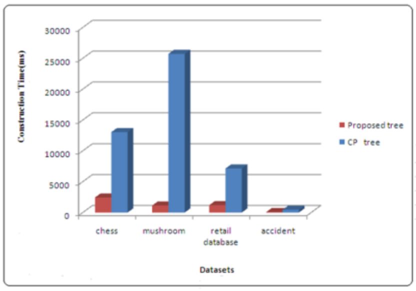


Fig. 7 Construction time V/s Database

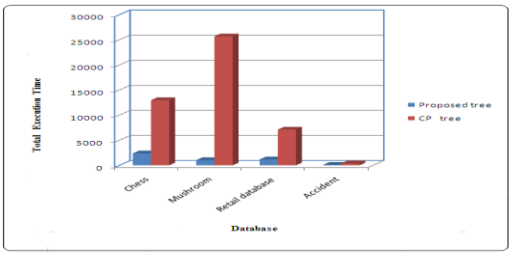


Fig.8 Execution Time V/s Database

CONCLUSION

a) The advantage of new tree data structure is that it occupies less memory as compared to CP tree because it stores only frequent items.

b) Furthermore, it also supports Interactive mining like Can-Tree and CP-Tree, means if user specified minimum support is changed then also it can extract frequent patterns without the need to rescan the database.

c) In addition that, it also supports Incremental mining like Can Tree and CP-Tree, means later if transaction(s) are added or old transaction(s) are deleted then also it can extract frequent patterns without the need to rescan the original database

d) This data structure not only mines frequent pattern using CP-mine algorithm but also gives better performance compared FP-growth algorithm.

e) There are many algorithms to extract frequent patterns once tree is constructed. Here we extract frequent patterns using cp-mine algorithm from our proposed tree data structure. CT-Pro [10] and other algorithm can also be used once CFP-tree is constructed to optimize the result

REFERENCES

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

[2] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong , Young-Koo Lee, "Efficient single-pass frequent pattern mining using a prefix-tree" In: Information Sciences-179 2008, pp 559–583.

[3] Leung, C.K., Khan, Q.I., Li, Z., Hoque, T., "CanTree: A Canonical-Order Tree for Incremental Frequent-Pattern Mining." In:Knowledge and Information Systems- 11(3) 2007, pp 287-311.

[4] Carson Kai-Sang Leung, Quamrul I. Khan, Zhan Li · Tariqul Hoque "CanTree: a canonical-order tree for incremental frequent-pattern mining." In: Springer-Verlag London Limited 2006.

[5] Jiawei Han, Jian Pei, Yiwen Yin, Runying Mao "Mining Frequent Patterns Without Candidate Generation: A Frequent-Pattern Tree Approach" In: Kluwer Academic Publishers 2004,pp 53-87.

[6] Han, Jiawei; Pei, Jian; Yin, Yiwen "Mining Frequent Patterns without Candidate Generation." In: ACM SIGMOD Intl. Conference on Management of Data, ACM Press 1999,pp 1-12.

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

[8] Koh J-L, Shieh S-F "An efficient approach for maintaining association rules based on adjusting FP-tree structures." In: Springer-Verlag, Berlin Heidelberg New York, 2004 pp 417–424

[9] Cheung, W., Zaïane, O.R., "Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint." In: Seventh International Database Engineering and Applications Symposium (IDEAS) 2003

[10] Bharat Gupta, Dr. Deepak Garg, "FP-Tree Based Algorithms Analysis: FPGrowth, COFI-Tree and CT-PRO" In: International Journal on Computer Science and Engineering (IJCSE), ISSN : 0975-3397 Vol. 3 No. 7 July 2011, pp1-9

[11] W. Cheung and O.R. Za¨ıane. Incremental mining of frequent patterns without candidate generation or support constraint. In Proc. IDEAS2003, pp. 111–116.

[12] J.-L. Koh and S.-F. Shieh. An efficient approach for maintaining association rules based on adjusting FP-tree structures. In Proc. DASFAA2004, pp. 417–424.

[13] R. Dass, A Mahanti, "An Efficient Algorithm for Real-Time Frequent Pattern Mining for Real- Time Business Intelligence analytics", Proceedings of the 39th Hawaii International Conference on System Science-2006.

[14] Nuansri Denwattana and Yutthana Treewai ,"Discovering of Frequent Itemsets with CP-mine Algorithm " from Department of Computer Science,Faculty of Science, Burapha University, Muang, Chonburi 20131 Thailand.

AUTHOR'S PROFILE

First Author: Minu Pandya is pursing MTech in Computer Science in SBTC College from Rajasthan Technical University in Jaipur.

Second Author: Mrs. Priyanka Trikha is completed her Mtech in Mody College of Engineering and Technology, Lachhmangarh . She is currently serving Assistant Professor at SBTC, Faculty of Computer Science.