Impact Factor (SJIF): 3.632

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444
Volume 3, Issue 5, May-2016

"Frequent Pattern – Projected Sequential Pattern Mining Improve its Efficiency and Scalability"

Ankita M Patel (Post Graduate Student of Hasmukh Goswami college of Engineering, Ahmedabad)
Asst. Prof. Dhaval Patel (Assistant Professor of Computer Engineering Department, Ahmedabad)

Abstract

Data mining has become an important field and has been applied extensively across many areas. Mining frequent item sets in a transaction database is critical for mining association rules. This paper propose a novel frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree based mining method. Efficiency of mining is achieved by the apply parallel projected database and partition projected database in frequent pattern tree to reduce the database scan, execution time and less memory. So the large database is compressed into a condensed, smaller data structure. The FP Tree algorithm is faster than the Apriori algorithm and also faster than new proposed projected database frequent-pattern tree.

Keywords
Association Rule mining, Data Mining, Frequent Pattern Mining, Parallel Projected database, partition projected database.

1. INTRODUCTION

Association rule mining is to find association relationships among large data sets [1]. An association rule is of the form X => Y. And each rule has two measurements: support and confidence. The association rule mining problem is to find rules that satisfy user-specified minimum support and minimum confidence. It mainly includes two steps: first, find all frequent patterns; second, generate association rules through frequent patterns.

In 1994, Agrawal proposed the Apriori algorithm, but there are two drawbacks in it. First, it will scan database many times repeatedly for finding candidate item sets, second, it will be very low and inefficient when memory capacity is limited with large number of transaction. This problem solve by frequent pattern- tree (FP-Growth).The advantage of the FP-Growth algorithm is that it only scans database twice. It directly compresses the database into a frequent pattern tree instead of using a candidate set and finally generates association rules through FP-tree [2.3]. FP- Growth Tree is more efficient and better performance than apriori algorithm but the problem is that FP-Tree is also a large hierarchical data structure and cannot fit into the main memory and also it is not suitable for "Incremental-mining" nor used in "Interactive-mining" system. And the time complexity of FP Tree is very high because it takes large execution time to process the large number of database. So apply parallel projected database and partition projected database in frequent pattern tree. It requires only one time scanning. Parallel projected database FP tree take to less execution time than the partition projected database FP tree and the parallel projected database require more memory than the partition projected database. So the parallel projected database FP tree faster than the partition projected database FP tree .after comparison between the parallel projected database FP tree, partition Projected database FP tree and FP tree algorithm.

At the beginning it constructs a FP-tree like what FP-growth does, then to every item sorting according to descending order in the transaction dataset after construct the parallel projected database FP tree using depth first stagey. And partition projected database FP tree using divide conquer method.

The results of our experiments show that the partition projected database FP tree faster than FP tree and also faster than the parallel projected database FP tree. So the parallel projected database FP tree more efficient and scalable than the partition projected database FP tree.

This paper is organized as follows: Section 2.offers a literature review to identify the noticeable research already conducted on this paper central objective Section 3. Briefly reviews FP-tree structure. Section 4. Proposes a new algorithm. Section 5. Experimental Analysis 6. Concludes this paper.

2. LITERATURE SURVEY

Frequent item set mining is important role in association rules mining. The apriori algorithm and the FP-growth algorithm are the most famous algorithms [8, 9, 10]. FP-tree in short A novel compact data structure called frequent-pattern tree is constructed, which is extended prefix-tree structure storing crucial, quantitative information about frequent patterns. To ensure that the tree structure is compact and informative, only frequent length items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of node sharing than less frequently occurring ones. A tree is compact, and it is sometimes orders of magnitude smaller than the original database. Subsequent frequent-pattern mining will only need to work on the FP-tree instead of the whole data set. Second, an FP-tree-based pattern-fragment growth mining method is developed, which starts from a frequent length-1 pattern examines only its A performance study has been conducted to compare the performance of FP-growth with two representative frequent-pattern mining methods and Tree Projection. FP-growth is about an order of magnitude faster than Apriori, especially when the data set is dense (containing many patterns) and/or when the frequent patterns are long; also, FP-growth outperforms the Tree Projection [11] algorithm. Moreover, our FP-tree-based mining method has been implemented in the DB Miner system and tested in large transaction databases in industrial applications. Akshita Bhandari, Ashutosh Gupta, Debasis Das [12] in their paper for Elsevier proposed Improvised Apriori algorithm. Its Aim to reduce the time consuming for candidate item set generation. And also by reducing the number of transactions to be scanned. Whenever the k of k-item set increases, the gap between our improved Apriori and the original Apriori increases from view of time consumed, and whenever the value of minimum support increases, the gap between our improved Apriori and the original Apriori decreases from view of time consumed. This approach is far more efficient than the original apriori algorithm.O.Jamsheela, Raju.G[13] in their paper for IEEE Proposed Binary Search Tree Using Adaptive Method. Its aim to be improved by minimizing the searching time of items while sorting out the transactions.Sajedul Hoque, Sujit Kumar Mondal, Tassnim Manami Zaman, Dr. Paresh Chandra Barman & Dr. Md. AI·Amin Bhuiyan[14] in their paper for IEEE implication of association rules among the quantitative and categorical attributes of a database employing classical logic and Frequent Pattern (FP) . Growth algorithm which solved using fuzzy logic. Wu, Defu Zhang, Qihua Lan, Jiemin Zheng[15] in their paper for IEEE proposed Apriori-Growth algorithm. This algorithm only scans the data set twice and builds FP-tree once while it still needs to generate candidate item sets. Wei Zhang, Hongzhi Liao & Na Zhao[16] IN THEIR PAPER FOR IEEE proposed FP-growth algorithm .Its aim to resolve two neck-bottle problems of traditional apriori algorithm and has more efficiency than original one. Qihua Lan, Defu Zhang, Bo Wu[17] in their paper for IEEE proposed APFT which works little faster than FP-Growth when the support threshold is small. Paresh Tanna, Y ogesh Ghodasara [18] in their paper for IEEE proposed matrix Aprior algorithm. Its aim to the imperative update problem. It handles both additions and deletions in increments and avoids a full database scan when the database is updated.

3. FP TREE

Han et al. developed an efficient algorithm, Growth, bases on FP-tree. It requires two scans of the database [19]. In first scan of the database FP- Growth first computes a list of frequent items sorted by frequency in descending order. In second scan of the database, the database is compressed into a FP-tree. After that FP-Growth starts to mine the FP-tree for each item whose support is larger than ξ by recursively building its conditional FP-tree.

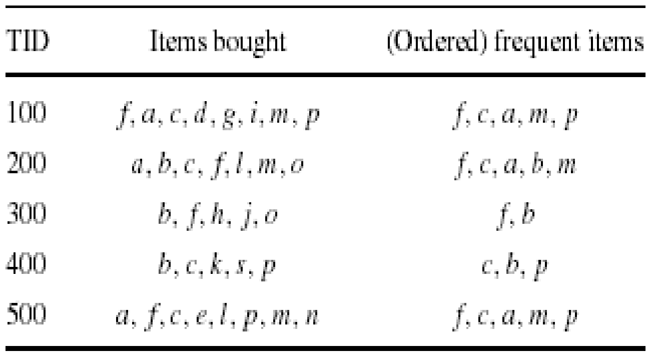
The FP-tree is a compressed representation of the transactions, and it also allows quick access to all transactions that share a given item. Once the tree has been constructed, the subsequent pattern mining can be performed [20].

3.1 FP Tree Algorithm

Algorithm 1. FP-tree construction [7]
Input: A transaction database TDB and a minimum
Support threshold ξ.
Output: Its frequent pattern tree, FP-Tree
1. Scan the transaction database DB once. Collect the
Set of frequent items F and their supports. Sort F in support descending order as L.
2. Create the root of an FP-tree, T, and label it as
“Null”, for each transaction in TDB do the following.
Select and sort the frequent items in transaction

```
According to the order of L. Let the sorted frequent
item list in transaction be [p|P], where p is the first
Element and P is the remaining list. Call insert-tree
([p|P, T].
Function insert-tree ([p|P], T)
 If T has a child N such that N.item-name = p.itemname
Then increment N’s count by 1;
Else do
Create a new node N;
 N’s count = 1;
 N’s parent link be linked to T;
 N’s node-link be linked to the nodes with the same
Item-name via the node-link structure;
If P is nonempty, Call insert-tree (P, N);
```

An example of an FP-tree is shown in Figure 1.This FP-tree is constructed from the TDB shown in Table 1. With minsup = 3.



TID	Items bought	(Ordered) frequent items
100	*f, a, c, d, g, i, m, p*	*f, c, a, m, p*
200	*a, b, c, f, l, m, o*	*f, c, a, b, m*
300	*b, f, h, j, o*	*f, b*
400	*b, c, k, s, p*	*c, b, p*
500	*a, f, c, e, l, p, m, n*	*f, c, a, m, p*

Table 1 : Example of FP tree

First, a scan of DB derives a list of frequent items, (f: 4), (c: 4), (a: 3), (b: 3), (m: 3), (p: 3) in which items are ordered in frequency-descending order. This ordering is important since each path of a tree will follow this order.

Second, the root of a tree is created and labeled with “null”. The FP-tree is constructed as follows

1. The scan of the first transaction leads to the construction of the first branch of the tree: (f: 1), (c: 1), (a: 1), (m: 1), (p: 1) Notice that the frequent items in the transaction are listed according to the order in the list of frequent items.

2. For the second transaction, since its (ordered) frequent item list f, c, a, b, m shares a common prefix f, c, a with the existing path f, c, a, m, p, the count of each node along the prefix is incremented by 1, and one new node (b:1) is created and linked as a child of (a:2) and another new node (m:1) is created and linked as the child of (b:1).

3. For the third transaction, since its frequent item list f, b shares only the node f with the f -prefix sub tree, f ’s count is incremented by 1, and a new node (b:1) is created and linked as a child of (f :3).

4. The scan of the fourth transaction leads to the construction of the second branch of the tree, (c: 1), (b: 1), (p: 1).

5. For the last transaction, since its frequent item list f, c, a, m, p is identical to the first one, the path is shared with the count of each node along the path incremented by 1.

In Figure 1. Every node is represented by (item - name: count). Links to the next same item-name node are represented by dotted Arrows.

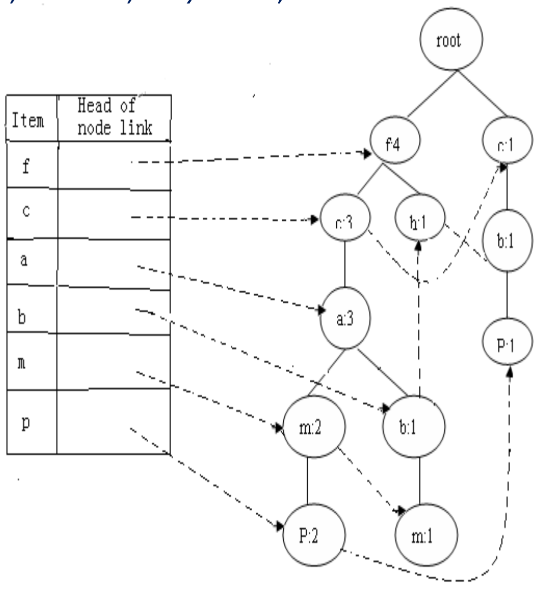


Figure1: Constructing of FP-tree

4. PROPOSED ALGORITHM

In this paper new algorithm is proposed which combined FP Tree algorithm and projected Database algorithm.

1. Parallel Projected Database FP Tree
2. Partition Projected Database FP Tree

4.1 Construct Parallel Projected Database FP Tree

Scan the number of transaction database to be equal number of projected database .In parallel projected database more than one program will execute at a time and all the projected datasets are stored in the same memory location and easily retrieved from data. The Parallel Projected databases are scan at the end of the mining and these projected databases can be mined in parallel but it takes more memory.

Algorithm 2: Parallel Projected Database FP-tree construction

```
Input:  A transaction database and a minimum support threshold ξ.
Output: PFP-tree
Method:
 (1) First scan the transaction database (D) once.
 (2) Count the number of frequent item (I) and their Support (n).
 (3) Sort the list of  frequent items (I) to the descending order in Transaction Database (D).
 (4) Compute the F_list which contain the frequent item in the descending order.
 (5) Apply parallel projection in the FP tree.
       ➢  Using the Depth First Strategy the projected trees are determined the each frequent item.
       ➢ Each projected database(PDB) store the set of frequent item & their support[F:n]
       ➢ Compare the each projected database to the transaction database
       ➢ If  available parent node  = yes
           {
           Save the parent node in the current projected database
           }
           Otherwise
           {
           Empty
 }
```

4.2 Construct Partition Projected Database FP Tree

Scan the number of transaction in to the projected database. In partition projected database only one projected database scan after scanning process the entire database. The partition is logically done by the projection scheme and it defines the set of projected segments. Each segment is processed separately with its own local memory.

Algorithm 3: Partition Projected Database FP-tree construction

Input: A transaction database and a minimum support threshold ξ.

Output: PFP-tree

Method:

```
(1) First scan the transaction database (D) once.
(2) Count the number of frequent item (I) and their Support (n).
(3) Sort the list of frequent items (I) to the descending order in Transaction Database (D).
(4) Compute the F_list which contain the frequent item in the descending order.
(5) Apply Partition Projection in the FP tree.
      ➢ Using the Divide and Conquer method
      ➢ Read the First Transaction database after create the projected database to the particular item.
      ➢ Each projected database (PDB) counts frequent of each item (flocal(i)) using its local data.
      ➢ Compare the particular projected database to the transaction database
      ➢ If available parent node = yes
          {
          Save the parent node in the current projected database
          } else

          {
          After create the projected database to the parent node
          Finally merge the all items then generate a global count.
          Remove the items with support count less than the minimum support
          }
          Otherwise
          {
          Empty
 }
```

4.3 Formula of projected Database FP Tree

The amount of time required by the parallel algorithm (Tp) to compute the frequencies of the candidate patterns

$$Tp = f(D, I, M).$$

D=number of sequences in the database
I = number of distinct items
M= the number of nodes

M depends both on I and on the value of the minimum Support σ.
In particular, as I increases and/or σ decreases, the number of frequent patterns will also increase and M will increase.
The time complexity of the Projected FP Tree [21]

$$T_F = k_F \times \left(\frac{m}{p} + d\right) + a$$

TF = execution time of the projected database FP tree

kF = the number of iterations
p = number of projected
m = number of nodes
d = computation time of data transfer
a = given data set: super market analysis

5. EXPERIMENTAL ANALYSIS

5.1 Experiment-1

Aim: To compares the time consumed of Frequent Pattern tree algorithm, Frequent Pattern tree apply partition projected database and Frequent Pattern tree apply parallel projected database by using different number of record.

Real life dataset: market basket analysis.

Explanation: This experiment was performed for real life datasets. This experiment is shown in Table 2 for different number of record. This experiment was performed different number of record according to the different execution time. Number of record was taken 120 to 150.

Number of record	Execution Time (In millisecond) FP-Tree with conditional	Execution Time (In millisecond) Partition projected Database FP- Tree algorithm	Execution Time (In millisecond) Parallel projected Database FP- Tree algorithm
120	166	120	85
130	179	136	100
140	243	165	150
150	290	240	220

Table 2: Execution time with different number of record for real life datasets

The result analysis in figure2 is the time consuming in parallel projected database and partition projected database in each group of record is less than Frequent pattern tree and the difference increases more and more as the number of record increases. Here partition projected database less than the FP tree and also less than the parallel projected database.

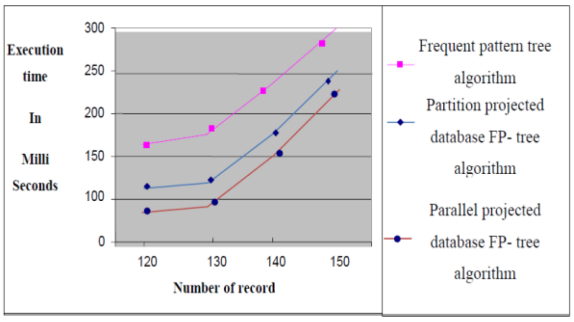


Figure 2: Time consuming comparison for different Number of record on real life dataset

5.2 Experiment-2

Aim: To compares the time consumed of Frequent Pattern tree algorithm, Frequent Pattern tree apply partition projected database and Frequent Pattern tree apply parallel projected database by using different value of minimum support.

Real life dataset: market basket analysis.

Explanation: This experiment was performed for real life datasets. This experiment is shown in Table 3 for different min sup values. This experiment was performed different number of min sup values according to the different execution time. Minimum support was taken 2 to 5.

Minimum support	Execution time (In millisecond) **FP-Tree with conditional**	Execution time (In millisecond) **Partition projected Database FP- Tree algorithm**	Execution time (In millisecond) **Parallel projected Database FP- Tree algorithm**
2	360	349	276
3	340	290	250
4	250	220	160
5	210	150	130

Table 3: Execution time with different Min sup for real life datasets

The result analysis in figure3 the time consuming in parallel projected database and partition projected database in each value of minimum support is less than Frequent pattern tree and the difference increases more and more as the value of minimum support decreases.

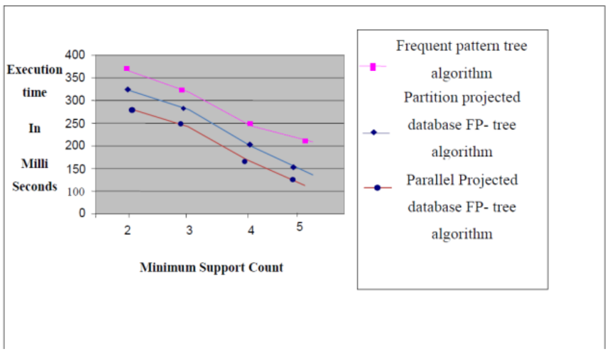


Figure3: Time consuming comparison for different minimum support value on real life dataset

6 CONCLUSION

In this paper, parallel projected database FP tree and partition projected database FP tree is proposed by reducing the time consumed in Number of record in the item set. When the k of k-item set increases so the gap between partition projected database FP tree less than FP tree and also less than parallel projected database FP tree from view of time consumed and whenever the value of minimum support increases so the gap between partition projected database FP tree less than FP tree and also less than parallel projected database FP tree from view of time consumed. When the k of k-item set fix and the value of minimum support increases so the gap between partition projected database FP tree less than FP tree and also less than parallel projected database FP tree from view of time consumed. The parallel projected database is more efficient and scalable than the partition projected database.

REFERENCES

(1) R.Agrawal, T.Imielinski and A.Swami, “Mining association rules between sets of items in large databases,” in Proceedings of the Association for Computing Machinery, ACM-SIGMOD,May- 1993 , pp.207-216.
(2) L. Zhou , X. Wang “Research of the FP Growth algorithm based on Cloud Environment” Journal of Software, March 2014,volume 9 N0. 3.
(3) J. Han, J. Pei and Y. Yin “Mining Frequent Patterns without Candidate Generation (PDF)” Proc. 2000, ACM-SIGMOD Int May 2000.
(4) R. Agrawal and R. Srikant “Fast algorithms for mining association rules” VLDB, 1994 pp 487-499.
(5) L. Haoyuan, Y. Wang, Z. Dong, Z. Ming, C. Edward “PFP Parallel FP Growth for query Recommendation”.
(6) J. Han, M. Kamber “Data Mining-Concepts and Techniques” Sam Francisco 2009, Morgan Kanufmann Publishers.

(7) J.S. Park, M.S. Chen and P.S. Yu "An effective hashbased algorithm for mining association rules" In SIGMOD1995, pp 175-186.

(8) Q. Lan, D. Zhang, B. Wu "A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree" In Department of Computer Science, Xiamen University, Xiamen 361005, China 2009.

(9) J. Han, J. Pei, and Y. Yin "Mining Frequent Patterns without Candidate Generation (PDF)" (Slides), Proc. 2000 ACM-SIGMOD Int. May 2000.

(10) R. Agrawal and R. Srikant "Fast algorithms for mining association rules"In VLDBY94, pp. 487-499.

(11) R.Agarwal, C.Aggarwal, V Prasad "A tree projection algorithm for generation off request item sets" In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining) 2000.

(12) Akshita Bhandari, Ashutosh Gupta, Debasis Das **"**Improvised apriori algorithm using frequent pattern tree for real time applications in data mining" in Elsevier2014

(13) O.Jamsheela, Raju.G **"**An Adaptive Method for Mining Frequent Item sets Efficiently An Improved Header Tree Method**"** In IEEE2015

(14) S. Hoque, S.K. Mondal, T. M. Zaman, Dr. P.C. Barman & Dr. M. A.Bhuiyan **"**Implication of Association Rules Employing FP·Growth Algorithm for Knowledge Discovery" Dept. of Computer Science & Engineering, Northern University Bangladesh, Dhaka, Bangladesh in IEEE 2011.

(15) W.D. Zhang, Q.Lan, J. Zheng**"** "An Efficient Frequent Patterns Mining Algorithm based on Apriori Algorithm and the FP-tree Structure" Department of Computer Science, Xiamen University, Xiamen 361005, China Longstop Group Post-doctoral Research Center, Xiamen, 361005, China in IEEE 2008 .

(16) W. Zhang, H. Liao & N. Zhao "Research on the FP Growth Algorithm about Association Rule Mining" in IEEE2008.

(17) Q. Lan, D.Zhang, B.Wu"A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree"Department of Computer Science, Xiamen University, Xiamen 361005, China in IEEE2009.

(18) P. Tanna, Y. Ghodasara" Frequent Pattern Mining Based On Imperative Tabularized Apriori Algorithm (ITAA)"IN IEEE2015

(19) J. Han, M. Kamber "Data Mining-Concepts and Techniques" Sam Francisco 2009, Morgan Kanufmann Publishers.

(20) J. Han, J. Pei, R. Mao" Mining Frequent Patterns without Candidate Generation A Frequent- Pattern Tree Approach" Data Mining and Knowledge Discovery, April 2001, Kluwer Academic Publishers, Manufactured in the Netherlands.

(21) N**.** Dang, V. Bay, L. Bac "Efficient strategies for parallel mining class association rules" University of Information Technology, Vietnam National University, Ho Chi Minh, Viet Nam.