FP-Tree Based Algorithms Analysis: FP-Growth, COFI-Tree and CT-PRO

Bharat Gupta
Student, Department of Computer Science
Thapar University
Patiala, India
bharatgupta35@gmail.com

Dr. Deepak Garg
IEEE Senior Member, ACM Senior Member, Department of Computer Science
Thapar University
Patiala, India
dgarg@gmail.com

***Abstract* — Mining frequent itemsets from the large transactional database is a very critical and important task. Many algorithms have been proposed from past many years, But FP-tree like algorithms are considered as very effective algorithms for efficiently mine frequent item sets. These algorithms considered as efficient because of their compact structure and also for less generation of candidates itemsets compare to Apriori and Apriori like algorithms. Therefore this paper aims to presents a basic Concepts of some of the algorithms (FP-Growth, COFI-Tree, CT-PRO) based upon the FP- Tree like structure for mining the frequent item sets along with their capabilities and comparisons.**

Keywords – Data mining, FP-Tree based Algorithm, Frequent Itemsets.

1. INTRODUCTION

Frequent item set mining is one of the most important and common topic of research for association rule mining in data mining research area. As an association rule mining is defined as the relation between various itemsets. Association rule mining takes part in pattern discovery techniques in knowledge discovery and data mining (KDD). As performance of association rule mining is depends upon the frequent itemsets mining, thus is necessary to mine frequent item set efficiently.

The process of extracting association rule mining consists of two parts firstly, mine all frequent itemsets pattern each of these pattern should satisfy the minimum support threshold. Once these entire frequent patterns are mined, then only second phase of mining i.e. association rules are produced from these frequent itemsets. These association rules must satisfy the minimum support and minimum confidence. This minimum support and confidence should be defined by the user. In this way the generation of association rule is largely depends upon the generation of the frequent items in the first phase.[1,3,4]

A large number of algorithms were proposed by many researchers for generation of frequent itemsets , firstly Apriori and Apriori like algorithms are proposed [1,3,15] but due to their large number of candidate generation , more database scan, slow processing and for some cases when support threshold is low then frequent patterns generation become problematic because of high memory dependency, huge search space and large i/o required in these type of algorithms thus new algorithms have been studied like FP-Growth and their variations to reduce the memory requirements, to decrease I/O dependencies, and also to reduce the pruning strategies for efficiently generation of frequent itemsets. In this paper restrict only to the working, properties and comparative performance of FP-Growth and its variations.

The further organisation of this paper is as follows. In Section 2, we briefly define the problem statement for finding the frequent itemsets from transactional database. Section 4 defines the classic FP-Growth algorithm. Sections 4 define the existing techniques based upon the FP- tree data structure. Section 5 defines their comparisons. Section 6 concludes the paper.

2. PROBLEM STATEMENT

The problem of mining association rules over market basket analysis was introduced in [2] .i.e. finding associations between the items that are present in the transaction from the database. The database may be from any retail shop, medical or from any other applications [5]. As defined in [3] the problem is stated as follows: Let $I = i_1, i_2, \ldots\ldots i_m$ be a set of literals, called items and m is considered the dimensionality of the problem. Let

D be a set of transactions, where each transaction T is a set of items such that T ∈ I. A unique identifier TID is given to each transaction. A transaction T is said to contain X, a set of items in I. $X \in T$ An *association rule* is an implication of the form "$X \longrightarrow Y$ ", where $\in I$, $Y \in I$, and $X \cap Y = \emptyset$. An itemset X is said to be *large* or *frequent* if its *support* s is greater or equal than a given minimum support threshold σ. An itemset X satisfies a constraint C if and only if $C(X)$ is *true*. The rule $X \longrightarrow Y$ has a *support* s in the transaction set D if s% of the transactions in D contain $X \cup Y$. In other words, the support of the rule is the probability that X *and* Y hold together among all the possible presented cases. It is said that the rule $X \longrightarrow Y$ holds in the transaction set D with *confidence* c if c% of transactions in D that contain X also contain Y . In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent$X$. The problem of discovering all association rules from a set of transactions D consists of generating the rules that have a *support* and *confidence* greater than a given threshold. These rules are called Strong Rules. This association-mining task can be broken into two steps:

1. Finding the frequent k-itemset from the large database.
2. Generate the association rule from these frequent item sets.

In this paper, we focus exclusively on the first step: generating frequent itemsets.

3. FP-GROWTH ALGORITHM

Among from the many algorithms suggested like Apriori algorithms. It is based upon the anti monotone property. Due to their two main problems i.e. repeated database scan and high computational cost, there is need of compact data structure for mining frequent item sets, which moderates the multi scan problem and improve the candidate item set generation. Tree projection is an efficient algorithm based upon the lexicographic tree in which each node represents a frequent pattern [2].

FP-Growth algorithm [2] is an efficient algorithm for producing the frequent itemsets without generation of candidate item sets. It based upon the divide and conquers strategy. It needs a 2 database scan for finding all frequent item sets. This approach compresses the database of frequent itemsets into frequent pattern tree recursively in the same order of magnitude as the numbers of frequent patterns, then in next step divide the compressed database into set of conditional databases.

3.1 Construction of Frequent Pattern Tree

First scan the database and manage the items appearing in the transaction. Then all the items whose support is less than the minimum support which is user defined are considered as infrequent are deleted from consideration. All other remaining items are considered as frequent items and arrange in the sorted order of their frequency. This list is known as header table when store in table. All the respective support of the items is stored using pointers in the frequent pattern tree. Then construct the frequent pattern tree which is also known as compact tree. The sorted items according to frequency in header table are used to build the FP-tree. This needs a complete database scan. when the item insert in the tree checks if it exist earlier in tree as in same order then increment the counter of support by one which is mentioned along with each item in the tree separated by comma, otherwise add new node with 1 as a support counter[2,3]. A link is maintained using pointers which same item and its entry in header table. In header table, pointer points to the first occurrence of each item.

It uses the tree data structure which stores all frequent elements in a compact form as comparative with Apriori which uses array type structure. Let sample database in table 1 and corresponding the support count in table 2 is:

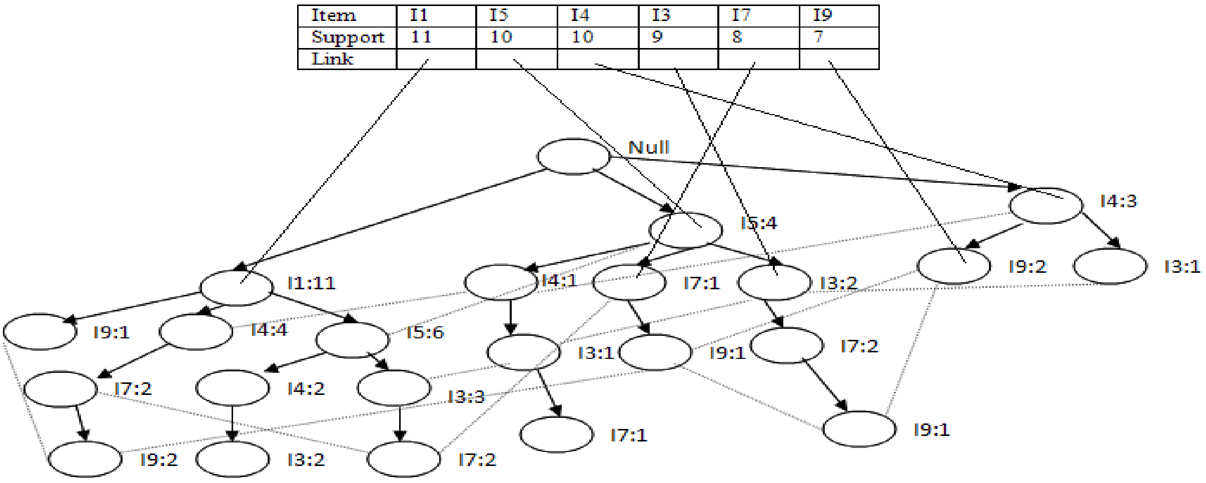
TABLE 1. SAMPLE DATABASE

Tid	Items
T1	I1, I2, I3, I4, I5
T2	I5, I4, I6, I7, I3
T3	I4, I3, I7, I1, I8
T4	I4, I7, I9, I1, I10
T5	I1, I5, I10, I11, I12
T6	I1, I4, I13, I14, I2
T7	I1, I4, I6, I15, I2
T8	I16, I7, I9, I17, I5
T9	I1, I9, I8, I10, I11
T10	I4, I9, I12, I2, I14
T11	I1, I3, I5, I6, I15
T12	I3, I7, I5, I17, I16
T13	I8, I3, I4, I2, I11
T14	I4, I9, I13, I12, I18
T15	I5, I3, I7, I9, I15
T16	I18, I7, I5, I1, I3
T17	I1, I17, I7, I9, I4
T18	I4, I3, I16, I5, I1

TABLE 2. SUPPORT COUNT FOR EACH ITEM

Item	Support	Item	Support
I1	11	I10	3
I2	4	I11	3
I3	9	I12	3
I4	10	I13	2
I5	10	I14	2
I6	3	I15	3
I7	8	I16	3
I8	3	I17	3
I9	7	I18	3

Suppose minimum support is 5. Thus delete all infrequent items whose support is less them 5.After all the remaining transactions arranged in descending order of their frequency. Create a FP- tree. For Each Transaction create a node of an items whose support is greater than minimum support, as same node encounter just increment the support count by 1.



Item	I1	I5	I4	I3	I7	I9
Support	11	10	10	9	8	7
Link						

Figure 1. FP-Tree Constructed For Sample Database

In this way after constructing the FP-Tree one can easily mine the frequent itemsets by constructing the conditional pattern base.

4. FP-GROWTH VARIATIONS

The above approach is efficient then Apriori algorithm but as the database become large it makes the processing slow, due to large database the FP-tree construction is very large and sometimes does not fit into the main memory. To overcome these limitations there are many optimized techniques listed in [10] on this basic FP-Growth algorithm [8, 9, 11,12, 15] . In this paper investigate the details of some of the variations of FP-growth namely COFI-tree mining [8], CT-PRO Algorithm [12] and FPgrowth [2] (as discussed above). Our goal is to take the overview details of each algorithm and discuss the main optimization ideas of each algorithm.

4.1 COFI-Tree Algorithm

COFI tree generation is depends upon the FP-tree however the only difference is that in COFI tree the links in FP-tree is bidirectional that allow bottom up scanning as well [7,8]. The relatively small tree for each frequent item in the header table of FP-tree is built known as COFI trees [8]. Then after pruning mine the each small tree independently which minimise the candidacy generation and no need to build he conditional sub-trees recursively. At any time only one COFI tree is present in the main memory thus in this way it overcome the limitations of classic FP-tree which can not fit into main memory and has memory problem.

COFI tree is based upon the new anti-monotone property called global frequent/local non frequent property [8]. It states that all the nonempty subsets of frequent patterns with respect to the item X of an X-COFI tree must also be frequent with respect to item X. In this approach trying to find the entire frequent item set with respect to the one frequent item sets. If the itemset participate in making the COFI tree then it means that item set is globally frequent but this doesn't mean that item set is locally frequent with respect to the particular item.

4.1.1 Algorithm: Create a COFI-Tree

1. Take FP-tree as an input with bidirectional link and threshold value.
2. Consider the least frequent item from the header table let it be X.
3. Compute the frequency that share the path of item X and remove all non frequent items for the frequent list of item X.
4. Create COFI tree for X known as X-COFI tree with support-count and participation=0
5. If items on Y which is locally frequent with respect to X form a new prefix path of X-COFI tree
6. DO, Set support count= support of X and participation count =0 for all nodes in a path.
7. Else adjust the frequency count and pointers of header list until all the nodes are not visited.
8. Repeat step 2 goes on until all frequent items not found.
9. Mine the X-COFI tree.

Support count and the participation count are used to find candidate frequent pattern and stored in the temporary list, which will be more clear after example.

Let the above FP-tree in figure 1 is the input for making COFI tree. Consider the links between the nodes are bidirectional. Then according to algorithm the COFI tree forms are first consider the itemset I9 as is least frequent item set, when scan the FP-tree the first branch is (I9,I1) has frequency 1, therefore frequency of the branch is frequency of the test item, which is I9. Now count the frequency of each item in a path with respect to I9. It is found that (I7, I3, I4, I5, I1) occur 4, 2, 4, 2, 3 times respectively thus according to anti-monotone property I9 will never appear in any frequent pattern expect itself. In the same way find the global frequent items for (I7, I3, I4, I5) which are also locally frequent, like for I7 two items I3 and I5 globally frequent item are also found locally frequent with frequency 5and 6 respectively thus a branch is created for each such items with parent I7. If multiple items share same prefix they are merged into one branch and counter is adjusted. COFI trees for items are follows:

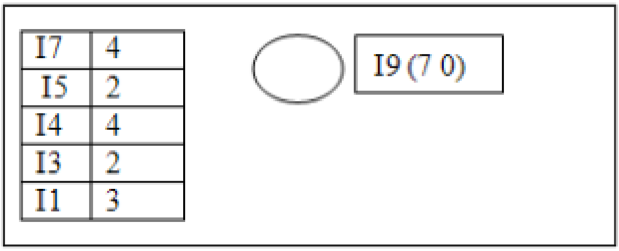


Figure 2. I9 COFI Tree

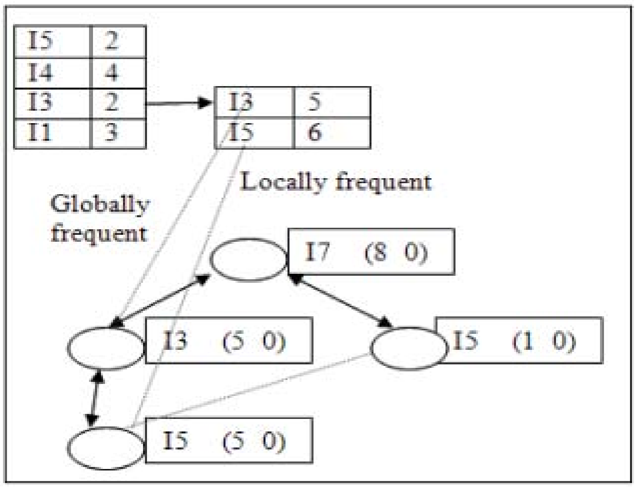


Figure 3. I7 COFI Tree

Similarly COFI tree is built for I3, I4, and I5. First after the globally frequent, find the itemsets which are also locally frequent , than find the support counter and make participation counter always equal to ‘0’. We are representing only for I7 in above example.

Once the COFI trees have built then we have to mine the frequent items from these COFI trees with the help of following procedure:

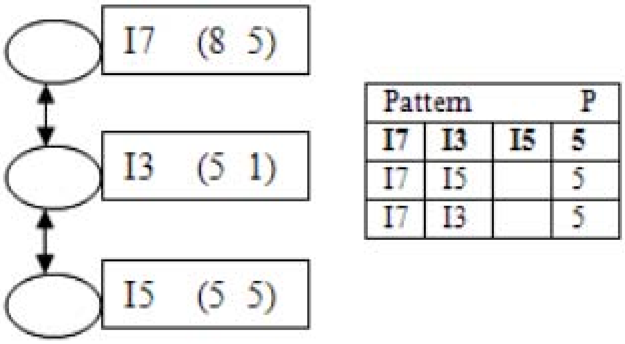
4.1.2 Algorithm: MINE X-COFI Tree

1. For node X select the item from the most frequent to least frequent with its chain
2. Until there are no node left, select all nodes from node X to root save in D list and in list F save the frequency count and participation count.
3. Generate all non discarded patterns Z from items D.
4. Add the list with frequency =F whose patterns not exist in X candidate list, else increment the frequency by F.
5. Increment the participation value by F for all items in D.
6. Select the next node and repeat step 2 until there is no node left.
7. Remove all non frequent items from X-COFI tree.

The COFI trees of all frequent items are mined independently one by one [8], first tree is discarded before the next COFI tree is come into picture for mining. Based on the support count and participation count frequent patterns are identified and non frequent items are discarded in the end of processing.

Let’s take an example for I7 COFI-Tree

As mining process start from the most local frequent item I5 and as from the figure 3, I5 exist in two branches I5, I3, I7 and I5, I7. Therefore

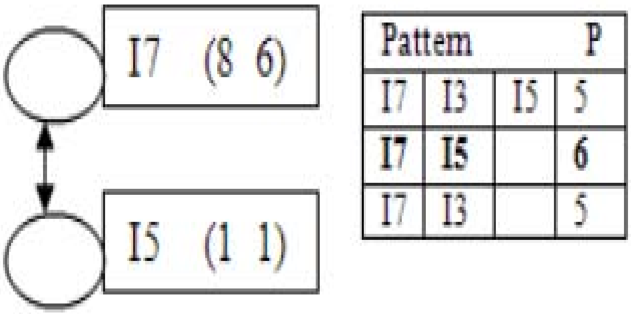


Pattem			P
I7	I3	I5	5
I7	I5		5
I7	I3		5

Figure 4. Mining E COFI tree for branch (I7,I3, I5)

As the frequency for each branch is equal to frequency of first item minus participation count of that node, Thus I5 has frequency 5 and participation count is 0 therefore first frequent set is found i.e. (I7, I3, I5:5).

Participation value is incremented by 1 for branch (I7, I3, I5) increment by 5 same as the frequency of I5. the pattern I7,I5 generates a pattern 1 which is already exist therefore increment the previous participation for I7, I5 by 1.therefore for I7 is become 6 and for I5 it become 1 for branch I7,I5.



Pattem			P
I7	I3	I5	5
I7	I5		6
I7	I3		5

Figure 5. Mining I7 COFI tree for branch (I7, I5)

Similarly mine for sub branch (I7, I3) and it is found that frequent patterns **(I7, I3: 5), (I7, I5: 6), (I7, I3, I5: 5)** for COFI tree I7. Similarly main the frequent patterns for I3, I4, and I5 itemsets.

In this way COFI tree mine the frequent item sets very easily then the FP-growth algorithm with the help of FP-tree[8]. It saves to memory space as well as time in comparison to the FP-growth algorithm. It mines the large transactional database with minimal usage of memory. It does not produce any conditional pattern base. Only simple traversal is needed in the mining process to find all the frequent item sets. It is based upon the locally and globally frequent item sets thus easily remove the frequent item sets in the early stages and don't allow any locally non frequent elements to takes part in next stage.

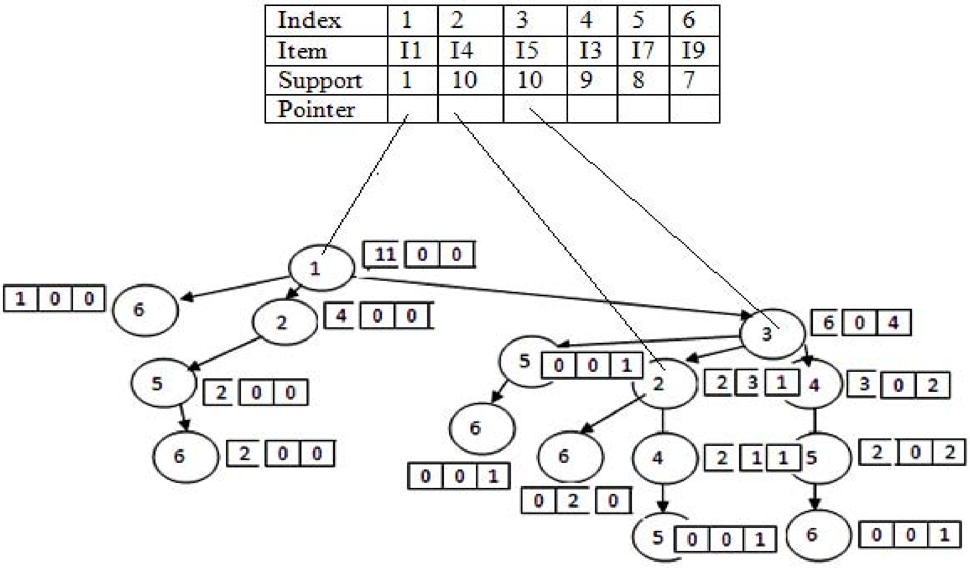
4.2 CT-PRO Algorithm

CT-PRO is also the variation of classic FP-tree algorithm [12]. It is based upon the compact tree structure [9, 11, 14]. It traverses the tree in bottom up fashion. It is based upon the non-recursive based technique [12]. Compress tree structure is also the prefix tree in which all the items are stored in the descending order of the frequency with the field index, frequency, pointer, item-id [11]. In this all the items if the databases after finding the frequency of items and items whose frequency is greater than minimum support are mapped into the index forms according to the occurrence of items in the transaction. Root of the tree is always at index '0' with maximum frequency elements. The CT-PRO uses the compact data structure known as CFP-tree i e. compact frequent pattern tree so that all the items of the transactions can be represented in the main memory [6, 12, 13].

The CT-PRO algorithm consists following basic steps [12]:

1. In the first step all the elements from the transaction are found whose frequency is greater than the minimum user defined support.
2. Mapping the elements according to the index value.
3. Construct the CFP-tree which is known as globally CFP-tree.
4. Mine the Global CFP-tree by making local CFP-tree for each particular index.

In this way by following the above steps can easily find the frequent item sets. The frequency of item sets greater then minimum support which defined as '5' for the sample database present in Table 1. After the mapping the Global Tree formed from the transactions are follows:



Index	1	2	3	4	5	6
Item	I1	I4	I5	I3	I7	I9
Support	1	10	10	9	8	7
Pointer						

Figure 6. CFP-Tree for Table 1

Note: No transaction starts from Item I3, I7, and I9 therefore no pointer is there.

4.2.1 *Steps for Making Global CFP-tree*

1. Take Database and threshold value as input.
2. Find the frequent items from database and sort in descending order in new list.
3. Then map the frequent items of the transaction in the index form, and sort in ascending order of their transaction Id (Tid).
4. Make maximum frequency item is the root node of the tree and makes it for index 1; insert all sub children in the tree.
5. For new index starting items, adjust pointer and build sub trees and give incremented index value or level as in above figure 6.
6. CFP-Tree has been built. Mine the CFP-tree index wise as projections.

After the global CFP-tree, the local CFP-Tree is build for each index item separately. It starts from the least frequency element.

The local CFP-Tree for the index 5 i.e. for the item I7 is found by first counting the other indexes that occur with the index 5. The other indexes are: 1, 3, and 4 which occur 4, 6 and 5 time respectively. As minimum support is 5 thus index 1 is pruned from local tree. The corresponding items are I1, I5 and I3 Item 1 is not locally frequent thus eliminated. Now construct the local CFP-tree projection for item I7 is as follows:

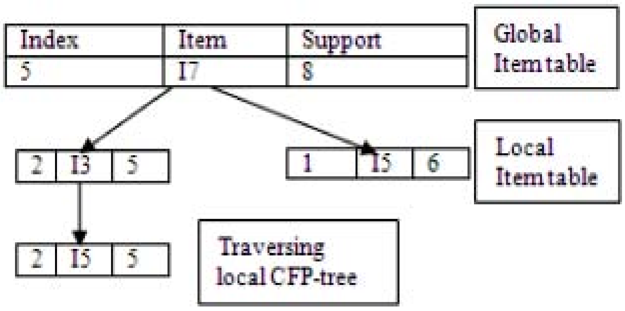


Figure 7. Frequent itemsets in Projection 5

the frequent items are that can be easily found by the above projection for index 5 is as follows:
(I7, I3, I5:5), (I7, I5: 6), (I7, I3: 5)
Similarly the frequent patterns are easily found for other indexes.

4.2.2 *Algorithm: Mine Global CFP-Tree*

1. Take Global CFP-tree as input.
2. Start from least frequent item index, check all the indexes come together in the global tree with the desired index.
3. Count support of all the indexes find on above step.
4. Prune all those indexes whose support is less then minimum support means those are not locally frequent.
5. Construct the local CFP-tree by those remaining indexes by again mapping along with the support.
6. Join the links between the items as same the linkage in the global CFP-tree in between only that item’s index i.e. parent should be parent and child will be child of particular existing projection.
7. The new supports of nodes are the support of frequent items of that particular projection.
8. Repeat the process until no index is left.

In this way CFP-tree Provide facility to easily mine the frequent items with the help of projections which prune the not frequent items locally and utilise the memory space efficiently by mining projections one by one. For large database the items can also easily fit into main memory.

5. COMPARISON OF ALGORITHMS

As from reviewing the various techniques i.e. FP-Growth [2], COFI-Tree [8], CT-Pro [12] and many more [6, 9, 11, 13, 14,15], we can differentiate them by the following considerations:

TABLE 3. COMPARISON OF FP-GROWTH, COFI-TREE, CT-PRO ALGORITHMS

Algorithm parameters	FP-Growth	COFI-Tree	CT-PRO
Structure	Simple Tree Based structure.	Uses Bidirectional FP-Tree structure.	Uses compressed FP-Tree data structure.
Approach	Recursive	Non- Recursive	Non- Recursive
Technique	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.	It constructs bidirectional FP-Tree and builds the COFI-Trees for each item then mines the COFI-Tree locally for each item.	It constructs the compact FP-Tree through mapping into index and then mine frequent itemsets according to projections index separately
Memory Utilization	Low as for large database complete Tree structure cannot fit into main memory	Better, Fit into main memory due to mining locally in parts for the complete tree, Thus every part represent in main memory	Best, as Compress FP-tree structure used and mine according to projections separately thus easily fit into main memory
Databases	Good for dense databases	Good for dense as well as Sparse databases. But with low support in sparse databases performance degrades.	Good for dense as well as for Sparse databases.

6. CONCLUSIONS

FP-Growth is the first successful tree base algorithm for mining the frequent itemsets. As for large databases its structure does not fit into main memory therefore new techniques come into pictures which are the variations of the classic FP-Tree. FP-Growth recursively mine the frequent itemsets but some variations COFI-Tree and CT-PRO based upon non recursive. Pruning method consists of removing all the locally non frequent items and also COFI-Tree and CT-PRO need less memory space and comparatively fast in execution then the FP-Tree because of their compact and different mining techniques.

REFERENCES

[1] R.Agrawal, R.Srikant, “Fast algorithms for mining association rules”, Proceedings of the 20th Very Large DataBases Conference (VLDB’94), Santiago de Chile, Chile, 1994, pp. 487-499.

[2] J.Han, J.Pei and Y.Yin., “Mining frequent patterns without candidate Generation”, in: Proceeding of ACM SIGMOD International Conference Management of Data, 2000, pp.1-12

[3] Jiawei Han, M.Kamber, “Data Mining-Concepts and Techniques”, Morgan Kanufmann Publishers, Sam Francisco, 2009.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data, pages 207–216, Washington, D.C., May 1993.

[5] M.-L. Antonie and O. R. Za¨ıane. Text document categorization by term association. In IEEE International Conference on Data Mining, pages 19–26, December 2002.

[6] J. Hipp, U. Guntzer, and G. Nakaeizadeh. Algorithms for association rule mining - a general survey and comparison. ACM SIGKDD Explorations, 2(1):58–64, June 2000.

[7] M. El-Hajj and O. R. Za¨ıane. Inverted matrix: Efficient discovery of frequent items in large datasets in the context of interactive mining. In Proc. 2003 Int’l Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD), August 2003.

[8] M. El-Hajj and O. R. Za¨ıane: COFI-tree Mining:A New Approach to Pattern Growth with Reduced Candidacy Generation. Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA, CEUR Workshop Proceedings, vol. 90 (2003)

[9] R. P. Gopalan and Y. G. Sucahyo, "High Performance Frequent Pattern Extraction using Compressed FPTrees", Proceedings of SIAM International Workshop on High Performance and Distributed Mining (HPDM),Orlando, USA, 2004.

[10] FIMI, "FIMI Repository", 2004, http://fimi.cs.helsinki.fi, last accessed at 20/04/2011.

[11] Y. G. Sucahyo and R. P. Gopalan, "CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth", Proceedings of the 14th Australasian Database Conference, Adelaide, Australia, 2003.

[12] Y. G. Sucahyo and R. P. Gopalan, "CT-PRO: A Bottom Up Non Recursive Frequent Itemset Mining Algorithm Using Compressed FP-Tre Data Structure”. In proc Paper presented at the IEEE ICDM Workshop on Frequent Itemset Mining Implementation (FIMI), Brighton UK, 2004.

[13] A.M.Said , P.P.Dominic, A.B. Abdullah; “ A Comparative Study of FP-Growth Variations”, In Proc. International Journal of Computer Science and Network Security, VOL.9 No.5 may 2009.

[14] S.P Latha, DR. N.Ramaraj, “Agorithm for Efficient Data Mining”, Proceeding on IEEE International Computational Intelligence and Multimedia Aplications 2007, pp. 66-70.

[15] Q.Lan, D.Zhang, B.Wu, “A New Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree”, Proceeding on IEEE International Conference on Global Congress on Intelligent System, 2009, pp.360-364.

AUTHORS PROFILE

Bharat Gupta, Member of IEEE Computer Society Student Chapter. He is student of Masters in Engineering in Computer Science at Thapar University, Patiala (Punjab). His research interests are Algorithm design and complexity issues for Data Mining.

Dr. Deepak Garg, Senior Member of ACM and is Chair, ACM North India SIGACT. He is Senior Member IEEE and Secretary IEEE Delhi Section Computer Society. He has more than 50 publications to his credit. His research interests are Algorithm design and complexity issues. He is having 15 years of research experience.