Improved FP-Growth Algorithm Based on Top Down Approach in Mining Frequent Itemsets

[1]M.Sinthuja, [2]Dr. N. Puviarasan, [3]Dr. P. Aruna

[1]Research Scholar, Professor and Head, Chidambaram, India, sinthujamuthu@gmail.com

[2]Associate Professor, Professor and Head , Chidambaram, India, npuvi2410@yahoo.in

[3]Professor and Head, Professor and Head, Chidambaram, India, arunapuvi@yahoo.co.in

Abstract - **In this research paper, the proposed improved FP-Growth algorithm based on top down approach (TD-IFP-Growth) aims to provide solution for efficient mining of frequent itemsets. In the existing FP-growth algorithm, searching of FP-tree adopts top down approach which generates conditional pattern base and sub-FP-trees. Bottleneck of FP-growth algorithm is the requirement of large amount of time and memory in generation of frequent itemsets. The proposed TD-IFP-Growth algorithm overcomes the problem of existing FP-Growth algorithm by searching the IFP-tree in top down approach which is opposite to FP-Growth algorithm. From the experimental results it is observed that the proposed TD-IFP-Growth algorithm consumes lower amount of time and memory as it does not generate conditional pattern base and sub-FP-tree.**

***Keywords* — Association Rule Mining; Bottom Up approach, Frequent Itemset Mining, FP-Growth, IFP-Growth, Minimum Suppor, Pruning, Top down approach**

I. Introduction

The era of automatized data collection applications and advanced database systems have given rise to tonnes of data for research in data warehouses, repositories and databases [1] [3]. Data should lead to useful information. The concept of datamining is used to extract useful and interesting information such as regularities, patterns, limitations and rules in databases. Data mining generates information about formerly accurate independent itemsets and their connection in a big database.

The items that are represented often in a database is called frequent itemsets. Frequent item sets are a group of items that occur often like notebook and pen in a transaction database. Frequent pattern mining bats for robust relationships in a data set. Frequent item sets may be transactional or relational.

This data industry has given birth to lot of data mining applications that can be applied in retail shopping for product showcasing and in internet for search of common words in pair. These applications work out for all items which have specific attributes-patient/symptoms, restaurants, menu, internet/keywords, basket/products, railway/timings etc.

Market basket analysis tells about the products that are often bought in pair and it might provide data for promotion strategies. For example, purchase of a shampoo is accompanied with the purchase of conditioner. Promotion of shampoo might increase the sale of the conditioner. Promotion of both the items in combination might increase the profit.

II. RELATED WORK AND CONTRIBUTION

Numerous techniques have been experimented for mining association rules in the research studies [4] [5]. In the arena of association rule mining the Apriori algorithm is most extensively used algorithm that generates candidate patterns [4]. It is a level-wise search. It mines frequent patterns by countless scans of a database. Based on Apriori algorithm, lots of algorithms have been worked out with some improvements such as AprioriTid algorithm [4]. It recovers more time and usage of memory is minimal. Apriori Hybrid [4], SetM (Set Oriented Mining of association rules) [4], Partition algorithm, Sampling algorithm, CARMA (Continuous Association Rule Mining algorithm) [6], DIC algorithm (prefix tree data structure) are further improved Apriori algorithm which decreases the database scans.

ECLAT uses a vertical layout of a database. Each item is symbolized by a set of transaction ids called tidset [7]. It overcomes the bottleneck of Apriori algorithm with regards to database scan. Rapid Association Rule mining (RARM) mentioned in [8] creates huge itemsets by adopting a tree structure-SOTrieIT and without scanning. Candidate itemset generation is not needed.

UT-Miner [9] is an improvised Apriori algorithm which is special in sparse data. In sparse data most of the transactions are distinct from each other. It uses a structure of array to increase the conduct of mining. But it does not assure in terms of runtime and usage of memory.

Another achievement in the frequent pattern mining is FP-Growth algorithm [10][11]. Han et al. introduced an energetic algorithm called FP-Growth which establishes a frequent pattern tree construction called FP-Tree. It overcomes two flaws of Apriori algorithm [10]. Primarily, it does not discover candidate patterns. Second, database scan is done only twice. Thus, it follows divide and rule method.

An improved frequent pattern (IFP) growth technique for generating frequent patterns is proposed [12] [13]. It needs lower usage of memory and it shows improved results in testing with FP-tree based algorithm.

FP-growth-goethals [11] which is optimized by Bart-Goethals is a FP-growth application. To improve the ability of in FP-growth, FP-growth-tiny explores conditional FP-trees makes use of conditional patterns without actually constructing any conditional database.

CT-PRO [8] partitions the database into many projections. Without any recursive calls, this algorithm mines frequent patterns by adopting the information combined in the tree.

III. PROPOSED RESEARCH WORK IMPROVED FP-GROWTH ALGORITHM BASED ON TOP DOWN APPROACH

3.1 Improved FP-growth algorithm

In this paper, the proposed algorithm of improved FP-growth algorithm based on top down approach (TD-IFP-Growth) is introduced. The form of improved FP-growth algorithm has four attribute of item name, count, node link and flag. Node link plays an important role of linking nodes with identical item. It helps to look for specific node rapidly. Thus the proposed algorithm quickly traverses the tree.

3.2 P*roposed Improved FP-growth algorithm based on top down approach*

The advantage of proposed TD-IFP-Growth algorithm is searching of tree in mining frequent patterns i.e., the adoption of top down approach in searching of tree. Block diagram of the proposed TD-IFP-Growth algorithm is shown in Fig 1. The construction of TD-IFP-Growth is as follows. Transaction database displayed in Table 1 is sorted based on support descending order. The items are prioritized in Table II. Minimum support threshold value used for this algorithm is greater than or equal to 2. Similar to FP-Growth algorithm, insertion of transaction in TD-IFP-Growth is infused.

The insertion of first transaction is portrayed in step 1. The itemset “Pen, Pencil, Eraser” is inserted into the tree with count 1 linking to node “root”. As this is the first transaction there is no node to link so link node is stated as “null” and flag value to “false” as there is no branch node. The entry of each item in the header table is updated. Likewise insert second transaction.

In case of third transaction “Pen, Pencil, Eraser” exist in the previous transaction so the count is incremented to 2 and a new node is created from item Eraser with count 1. Simultaneously update node link and flag value. For the insertion of fourth transaction “Pen, Pencil, Eraser, Gum”**,** it does not share any prefix of the tree a new node is created linking to node “root”. Likewise insert remaining items into the tree.

The searching of TD-IFP-tree in mining frequent itemset is as follows. In FP-growth algorithm the task of constructing conditional pattern base and conditional FP-tree affects the mining of FP-growth algorithm. To overcome this problem, in this paper proposed approach of top down search is adopted which avoids conditional pattern base and conditional FP-tree. Sub-header table is created for each item which reduces the runtime and memory. After the insertion of all transaction in IFP-tree portrayed in Fig. 2 applying top down approach.

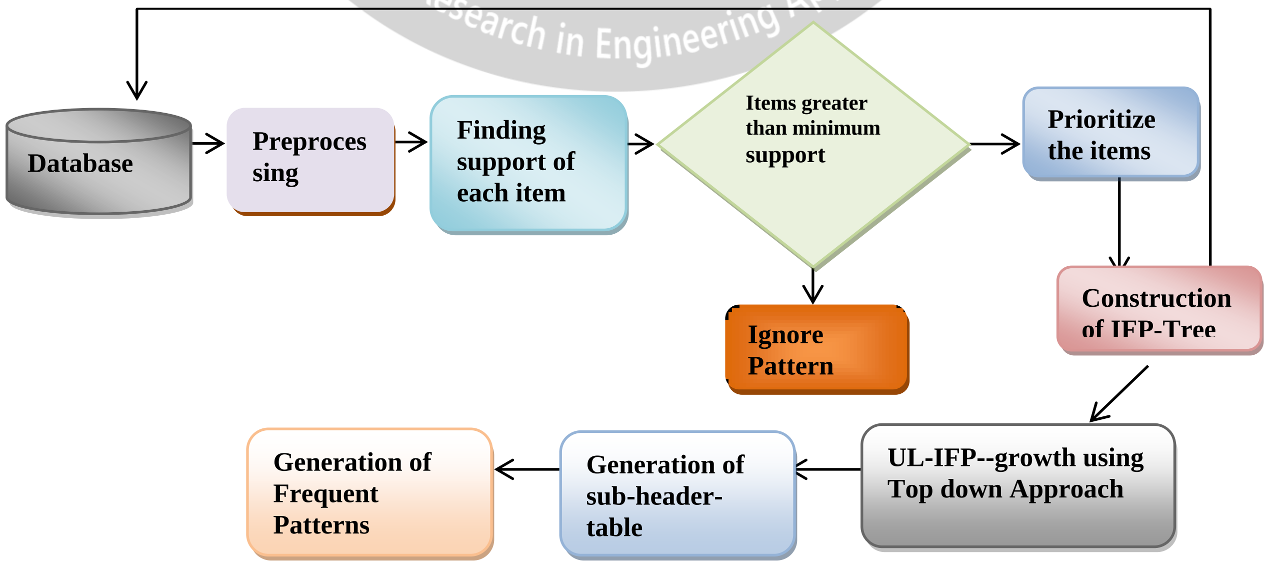

Database
Preprocessing
Finding support of each item
Items greater than minimum support
Prioritize the items
Ignore Pattern
Construction of IFP-Tree
UL-IFP--growth using Top down Approach
Generation of sub-header-table
Generation of Frequent Patterns

Figure 1 Block diagram of TD-IFP-Growth algorithm

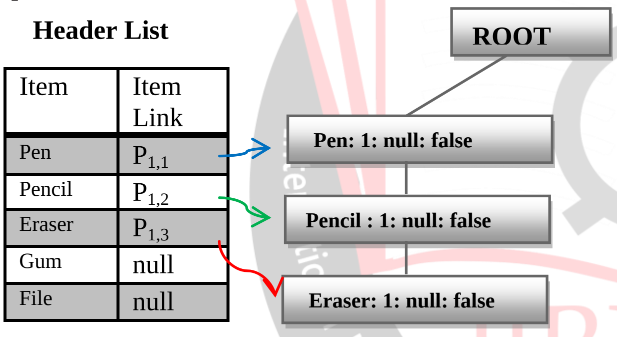
Table 1 Transaction Database

TID	Item bought	Ordered item
100	Paper, Pencil, Eraser, Pen	Pen, Pencil, Eraser
200	File, Pencil, Eraser, Pen, Note	Pen, Pencil, Eraser , File
300	Gum, Eraser, Pencil, Pen, Whitener	Pen, Pencil, Eraser, Gum
400	File, Gum, Pen	Pen, Gum , File
500	Pencil, Gum	Pencil, Gum

Table 2 Prioritize the items

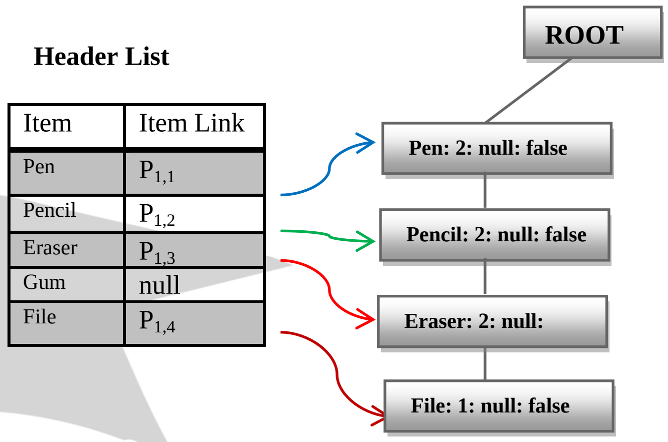
Items	Support
{Pen}	4
{Pencil}	4
{Eraser}	3
{Gum}	3
{File}	2
{Paper}	1
{Note}	1
{ Gum }	1

Step 1: Insertion of itemset { Pen, Pencil, Eraser }

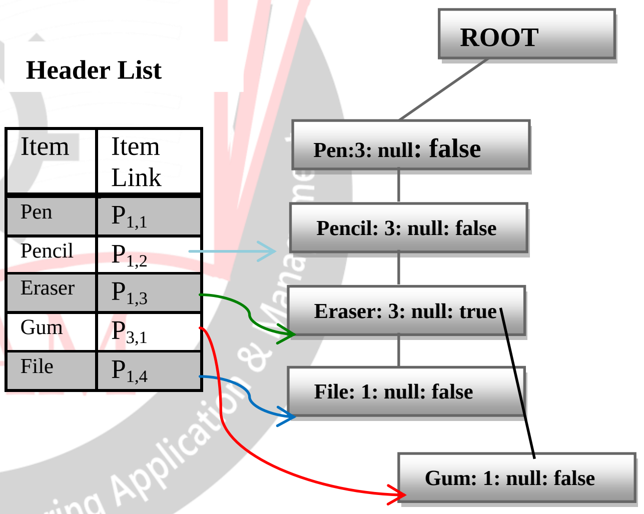


Start searching of TD-IFP-tree from item "Pen" which is present on top position of the header table. Here, by moving through the path of node link "Pen" it is found that there is no path. In this case item "pen" is alone frequent as it satisfies minimum support threshold value as portrayed in Fig 2.

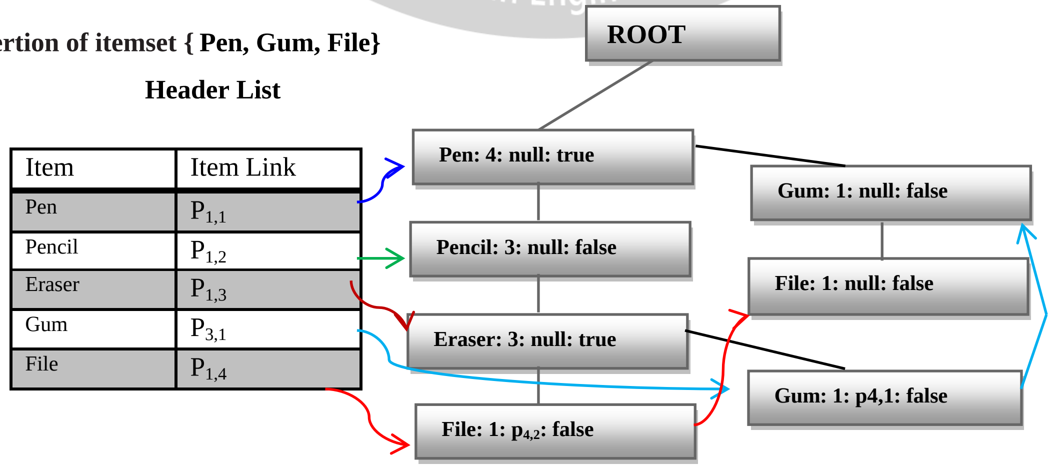
Step 2: Insertion of itemset { Pen, Pencil, Eraser , File }



Step 3: Insertion of itemset { Pen, Pencil, Eraser, Gum }



Step 4: Insertion of itemset { Pen, Gum, File}



Step 5: Insertion of itemset { Pencil, Gum }

Header List

Item	Item Link
Pen	$P_{1,1}$
Pencil	$P_{1,2}$
Eraser	$P_{1,3}$
Gum	$P_{3,1}$
File	$P_{1,4}$

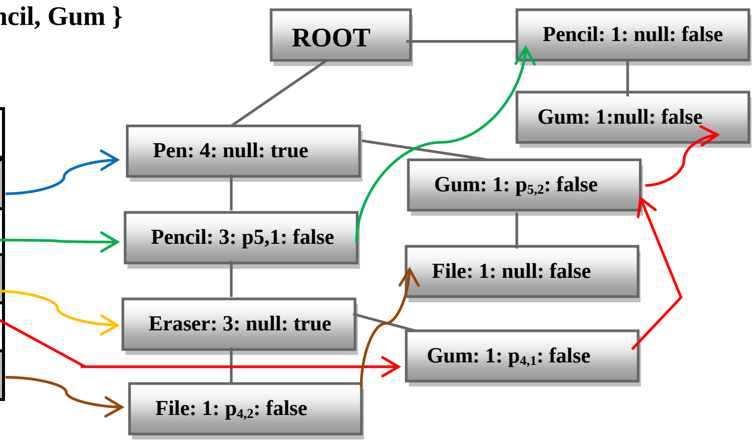


Figure 2 Construction of TD- IFP-Growth

Sub-Header-Table –Pen

Item Name	Count	Item Link
-	-	-

Sub-Header-Table –Pencil

Item Name	Count	Node Link
Pen	3	$P_{1,1}$

Then for the entry of item "Pencil" in header table by following the node link move through the path. One path is found from Root- Pen and it is inserted into sub-header-table-Pencil with count 3 as shown in Fig 2. Since, the minimum support is 2 itemset {Pen, Pencil: 3} is frequent. Then for item "Eraser" by passing through the path of node link "Eraser" one path is found from Root-Pen-Pencil with count 3. It is inserted into sub-header-table-Eraser with count 3. Thus itemset {Pen, Eraser: 3} {Pencil, Eraser: 3} are frequent. For item "Gum" sub-header-table is created by passing through the path. Three paths are found from Root-Pen-Pencil-Eraser, Root-Pen and Root-Pencil. Count for each path is updated in sub-header-table. Finally itemset {Pen, Gum: 2} {Pencil, Gum:2} are frequent. For the last entry of item "File" in header table sub-header-table is created based on the path. By moving through the path of item "File" two paths are found from Root-Pen-Pencil-Eraser and Root-Pen-Gum. Their counts are updated in sub-header-table for item File. Thus the frequent itemsets are {Pen, File:2}. Overall the explored frequent itemsets are {Pen, Pencil: 3} {Pen, Pencil, Eraser: 3} {Pen, Pencil, Gum: 2} {Pen, File:2}.

IV. EXPERIMENTAL STUDY AND ANALYSIS

In this research paper we compare the performances of proposed TD-IFP-Growth and FP-Growth algorithms. Different support levels are used for each datasets of algorithm. The experiments are conducted on Intel® corei3™ CPU, 2.13 GHz, and 2GB of RAM computer.

4.1 Dataset Description

The above specified algorithms are implemented using different standard datasets of different domains namely Chess, Mushroom and Connect where it is available in Tunedit Machine Learning Repository. Chess contains 3195 instances and 114 items. Mushroom accommodates 8124 instances and 58. Connect contains 67557 instances and 115 items.

4.2. Runtime Consumption

From the Fig. 3 y-axis display runtime in millisecond and x-axis display the different minimum support values. It includes the results of execution time of proposed TD-IFP-Growth and FP-growth algorithms on the dataset Chess with different minimum support.

Runtime means the total execution time between input and output. Runtime of proposed TD-IFP-Growth algorithm is 10872ms and FP-Growth algorithm is 12082ms for minimum support threshold value of 200 for the dataset Chess. From the graph it is observed clearly that the proposed TD-IFP-Growth algorithm performs quit well when compared to FP-Growth algorithm. It is because the proposed algorithm adopts top down search of tree. Thus the runtime to mine frequent itemset is fast.

With reference to Fig 4 for the minimum support threshold value of 1110 the runtime of proposed TD-IFP-growth algorithm is 7636ms and FP-Growth algorithm is 8336ms for the dataset Connect. When the minimum support is low generation of frequent itemset is more. Thus, the runtime is more but when compared to FP-Growth algorithm the proposed TD-IFP-Growth algorithm consumes lower memory as shown in figure. The comparison ratio of proposed TD-IFP-Growth and FP-Growth algorithm for the dataset Mushroom is shown in Fig 5. The runtime of both algorithms vary for different minimum support threshold value. Specifically, when the minimum support is high the runtime of proposed TD-IFP-Growth algorithms is fast. Because the generation of frequent itemset is reduced.

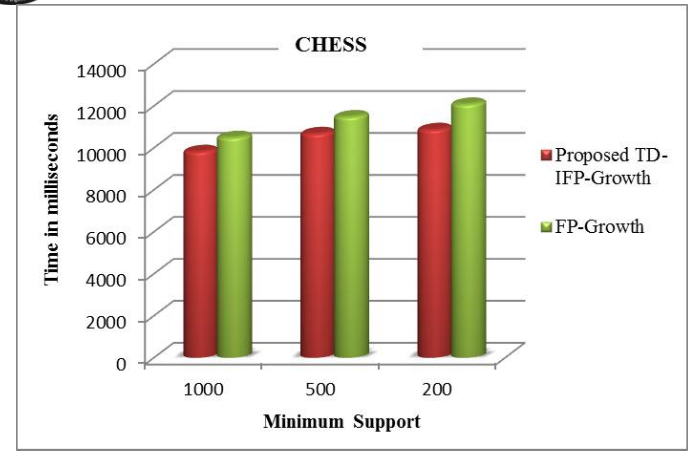


Figure. 3 Runtime of proposed TD-IFP-growth algorithm for the dataset Chess

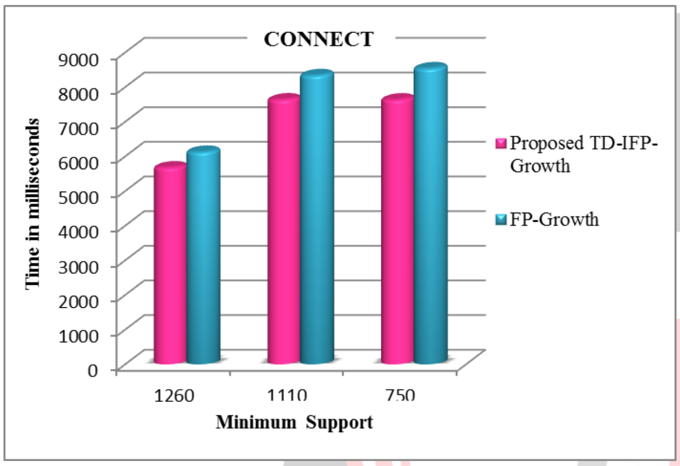


Figure 4 Runtime of proposed TD-IFP-growth algorithm for the dataset Connect

In case of FP-Growth algorithm, time take to run the algorithm is more when compared to proposed TD-IFP-Growth algorithm.

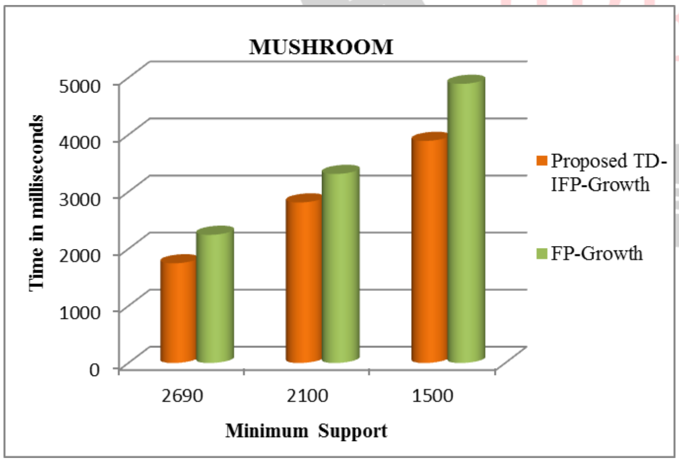


Figure 5 Runtime of proposed TD-IFP-growth algorithm for the dataset Mushroom

4.3. Memory Consumption

From the graph longitudinal axis shows the memory in MB and latitudinal axis shows the different support threshold values. The chart portrayed in Fig.6 shows the memory consumption of proposed TD-IFP-Growth algorithm and FP-Growth algorithm. When the minimum support threshold is fixed from higher to lower the proposed TD-IFP-Growth algorithm outperforms the existing FP-Growth algorithm.

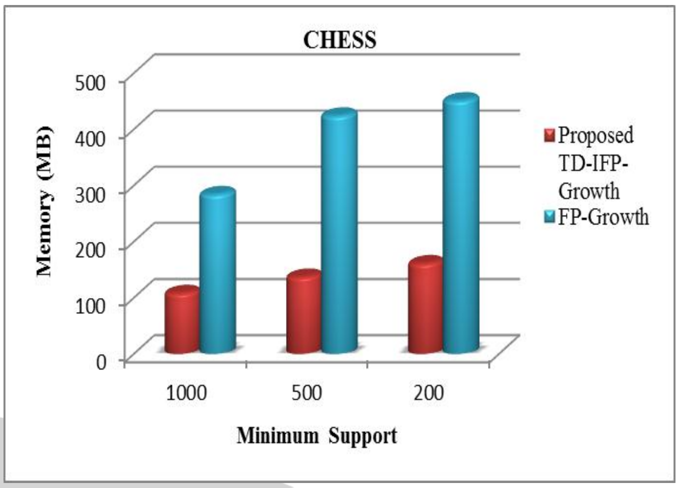


Figure 6 Memory consumption of proposed TD-IFP-growth algorithm for the dataset Chess

With reference to Fig. 7 the memory consumption of proposed TD-IFP-Growth algorithm is 146MB and FP-Growth algorithm is 296MB for the minimum support threshold value of 1110 for the dataset Chess. As the proposed algorithm adopts top down approach the generation of conditional pattern base and conditional sub-tree is not required. Thus the memory to store conditional patterns is avoided.

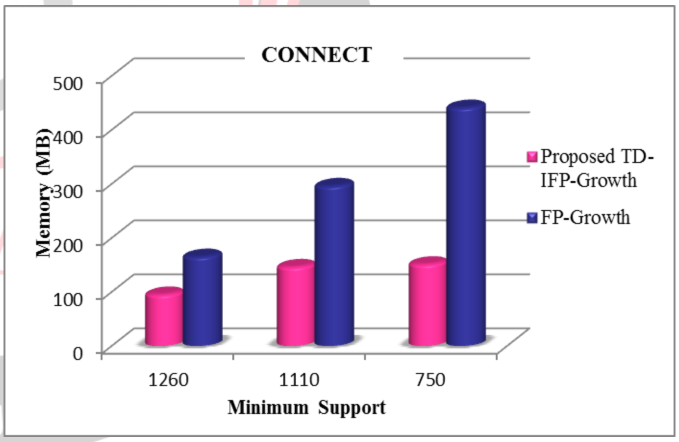


Figure 7 Memory consumption of proposed TD-IFP-growth algorithm for the dataset Connect

With reference to Fig. 8 the memory usage of proposed TD-IFP-Growth algorithm is 186MB and FP-Growth algorithm is 284MB for minimum support threshold value of 2690. It shows that TD-IFP-Growth is faster than FP-growth algorithm. The proposed TD-IFP-Growth algorithm requires less memory than FP-growth algorithm in case of maximum minimum support. In case of lower minimum support memory requirement for proposed TD-IFP-Growth is less when compared to FP-growth algorithm. On the whole the proposed TD-IFP-Growth algorithm is best for all cases.

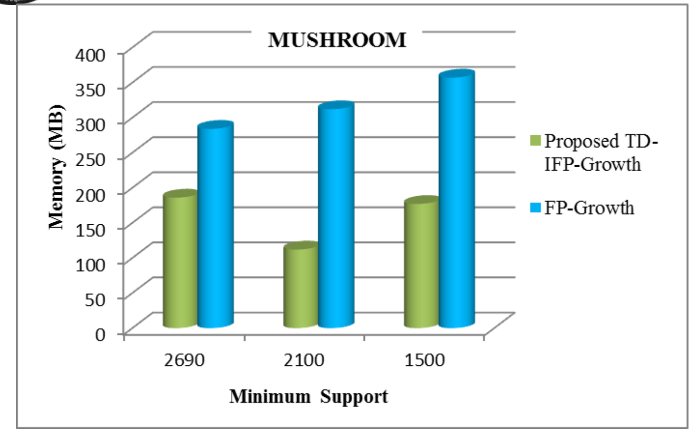


Figure 8 Memory consumption of proposed TD-IFP-growth algorithm for the dataset Mushroom

V. CONCLUSION

In this research paper, the proposed algorithm of TD-IFP-growth is introduced to mine frequent itemsets. The merit of proposed TD-IFP-growth algorithm is to reduce time and memory in mining frequent itemsets. The flow of proposed TD-IFP-growth algorithm is improved by adopting top down approach in generating frequent itemsets. Thus, the generation of conditional pattern base and sub-tree are avoided when compared to FP-Growth algorithm which consumes more time and memory in generating conditional pattern base and sub-tree. This paper analyses the performance of FP-growth and the proposed TD-IFP-growth algorithm. Benchmark databases used for the experimentation are Chess, Connect and Mushroom which is of different characteristics. From the experimental results, it is showed that TD-IFP-growth produced an outstanding performance in terms of runtime and memory usage while comparing with FP-growth algorithm.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their valuable suggestions which help in improving the quality of this paper. We would like to thank the Department of Computer Science and Engineering, Annamalai University for providing the lab facilities. We gratefully acknowledge the funding agency, the University Grant Commission (UGC) of the Government of India, for providing financial support for doing this research work.

REFERENCES

[1] Chung, H. M., Gray, P. (1999), "Special Section: Data Mining". Journal of Managemen Information Systems, (16:1),11-17.

[2] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, R (1996). "The KDD Process for Extracting Useful Knowledge from Volumes of Data," Communications of the ACM, (39:11), pp.27-34.

[3] Neelamadhab Padhy, Dr. Pragnyaban Mishra, and Rasmita Panigrahi, "The Survey of Data Mining Applications And Feature Scope", International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.2, No.3, June 2012.

[4] Agrawal, R., Imielinski, T., and Swami, A. N. 1993. "Mining association rules between sets of items in large databases". In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.

[5] J.W. Han , J. Pei , Y.W. Yin , "Mining frequent patterns without candidate generation", SIGMOD Rec. 29 (2) (20 0 0) 1–12 .

[6] J.W. Han , J. Pei , Y.W. Yin , "Mining frequent patterns without candidate genera tion', SIGMOD Rec. 29 (2) (20 0 0) 1–12 .

[7] M. Sinthuja, P. Aruna N. and Puviarasan, "Experimental evaluation of Apriori and Equivalence Class Clustering and Bottom Up Lattice Traversal (eclat) algorithms", pakistan journal of biotechnology vol. 13 special issue II (International Conference on Engineering and Technology Systems (ICET'16) pp. 77 - 82 (2016)

[8] WeeKeong,YewKwong Amitabha Das.:" Rapid Association Rule Mining", in Information and Knowledge Management, Atlanta, Georgia, (2001), pp. 474-481

[9] M. Hamada, K. Tsuda, T. Kudo, T. Kin, K. Asai, "Mining frequent stem patterns from unaligned RNA sequences", Bioinformatics 22 (20) (2006) 2480–2487.'

[10] M. Sinthuja, N. Puviarasan, and P. Aruna, "Comparative Analysis of Association Rule Mining Algorithms in Mining Frequent Patterns", International Journal of Advanced Research in Computer Science, Volume 8, No. 5, May – June 2017.

[11] M. Sinthuja, N. Puviarasan, and P. Aruna, "Comparison of Candidate Itemset Generation and Non Candidate Itemset Generation Algorithms in Mining Frequent Patterns", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 5 Issue: 3

[12] Ke-Chung Lin, I-En Liao, Zhi-Sheng Chen.: "An improved frequent pattern growth method for mining association rules", Expert Systems with Applications (2011) 5154–5161

[13] M. Sinthuja, N. Puviarasan, and P. Aruna, "Research of Improved FP-Growth (IFP) Algorithm in Association Rules Mining", International Journal of Engineering Science Invention (IJESI) ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726, PP. 24-31.