Comparison of Candidate Itemset Generation and Non Candidate Itemset Generation Algorithms in Mining Frequent Patterns

M.Sinthuja[1], N. Puviarasan[2], P. Aruna[3]
[1]Research Scholar, [2]Associate Professor, [3]Professor
[1, 3] Department of Computer Science and Engineering, [2]Department of Computer and Information Science
Annamalai University
Chidambaram, India
[1] *sinthujamuthu@gmail.com*, [2]*npuvi2410@yahoo.co.in*, [3]*arunapuvi@yahoo.co.in*

Abstract—Association rule mining is one of the important techniques of data mining used for exploring fruitful patterns from huge collection of data. Generally, the finding of frequent itemsets is the most significant step in association rules mining, and most of the research will be centered on it. Numerous algorithms have been discovered to find effective frequent itemsets. This paper compares the frequent pattern mining algorithms that use candidate itemset generation and the algorithms without candidate itemset generation. In order to have on field simulation for comparison, a case study algorithm from both types was chosen such as ECLAT and FP-growth algorithms. Equivalence class clustering and bottom up lattice traversal (ECLAT) algorithm accommodates 'Depth First Search' approach and requires the generation of candidate itemset. The FP-growth algorithm follows the 'Divide and Conquer' method and does not require candidate itemset generation. In this paper, the benchmark databases considered for comparison are Breast Cancer, Customer Data, and German Data etc. The performances of both the algorithms have been experimentally evaluated in terms of runtime and memory usage. From the result it is analyzed that the FP-tree algorithm is more advantageous as it does away with the need of generation of candidate patterns.

Keywords*- Association Rule Mining Algorithm; Data Mining; ECLAT; FP-Growth; Minimum Support; Pruning*

*****

I. INTRODUCTION

Large chain of stores like Wal-Mart in U.S has large number of stores worldwide. They have to deal with terabytes of data. It deals with billions of transactions every day. The major companies like telecommunication collect bulk of data. Also the banks and public departments collect huge amount of data. Thus there is need to design techniques that may extracts knowledge from a large database. Such techniques are provided by the data mining [1] [2].

It is the process of analyzing the data from various strata and consolidating into useful information. This useful information is used to generate revenue and cut costs. It is sometimes called as Data or Knowledge Discovery (KDD). Data mining can find interesting things based on consistency and replication. Traditional databases do not focus on intelligence about the data. Data mining provides intelligence essential for business. In normal systems, analysis of hypothesis takes time and money. The role of data mining (KDD) is very important in many of the fields such as analysis of market basket, classification, etc.

The knowledge discovery in data can be achieved by following steps: Data cleaning, Data integration, Data Selection, Data transformation, Data Mining, Knowledge discovery.

Association Rules

Association rule mining is one of important techniques in data mining [3]. The aim is to find out which items are frequently purchased together so that they are arranged accordingly on the rack of the store. This information can also be used in cross selling. Association rule mining finds its applicability in telecommunication and retail marketing.

It consists of if/then statements for linking the data, stored in warehouses, which may on the other side look unrelated. For instance if a person buys a new car he is most likely to get its insurance done. Data is analyzed for finding out frequent patterns to form association rules. The criterion used to discover relationships are support and confidence. The support and confidence are defined as:

- *Support*: It is the sum of times an item occurs in the database.
- *Confidence*: It defines the number of times an item appears provided the other has already occurred.

Association rule is composed of two parts:

- *Antecedent*: It is the 'if' part of the rule. It indicates the item that is found in the database.
- *Consequent*: It is the 'then' part of the rule. It indicates the item that is available when combined with the antecedent.

Association rules are used for the description of rules in the database. They are introduced in [1] to find out the relationship between the large dataset of transactions.

The most important role presented by frequent item set mining is to find out the interlinkage between the various parameters in a database. Exploration of frequent item set is done by association rules [15] [16].

II. RELATED WORKS

Numerous techniques have been experimented for mining association rules in the research studies [4] [7] [8] [9] [17]. In the arena of association rule mining the Apriori algorithm is most extensively used algorithm that generates candidate patterns [6]. It is a level-wise search. It mines frequent patterns by multiple scans of a database.

On the basis of Apriori algorithm, a numerous algorithms have been worked out with some improvements or adjustments such as AprioriTid algorithm [5]. It recovers more time and usage of memory is minimal. Apriori Hybrid [5], SetM (Set Oriented Mining of association rules) [6], Partition algorithm, Sampling algorithm, CARMA (Continuous Association Rule Mining algorithm), DIC algorithm (prefix tree data structure) [6] are further improved Apriori algorithm which decreases the database scans.

ECLAT uses a vertical layout of a database. Each item is symbolized by a group of Tids called tidset [10] [11] [17]. It overcomes the bottleneck of Apriori algorithm with regards to database scan. Rapid Association Rule mining (RARM) mentioned in creates large itemsets by using a tree structure-SOTrieIT and without scanning. It does not generate candidate itemset.

Another achievement in the frequent pattern mining is FP-Growth algorithm. 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 [12] [13] [14] [17]. First, it does not generate candidate patterns. Second, database scan is done only twice. It adopts divide and conquer method.

III. ASSOCIATION RULE MINING ALGORITHMS

This section mainly focused on the mining association rules to discover frequent patterns. Association rule mining algorithms of ECLAT and FP-growth are used in finding frequent itemsets.

A. ECLAT (Equivalence Class Clustering and Bottom Up Lattice Traversal)

Vertical Data Layout: The vertical (or inverted) layout consists of a list of items, with each item followed by its Tid-list the list of all the transaction identifiers containing the item [10].

ECLAT algorithm was developed to overcome the limitation of Apriori algorithm. ECLAT algorithm accommodates 'Depth First Search' approach and requires the generation of candidate itemset. The ECLAT algorithm was constructed to control the shortcomings of the count and candidate distribution algorithms. It uses the aggregate memory of the system by partitioning the candidates into disjoint sets using the equivalence class partitioning. The steps involved in ECLAT are shown in Algorithm 1.

In this example, horizontal database is given is Table I. So, it must be converted into vertical database. Requirement for scanning the database in this algorithm is only once. Support is counted in this algorithm. Confidence is not calculated in this algorithm. In this work, support is considered as greater than or equal to 3.

Each step for the construction of ECLAT is shown in Fig. 1. The number of items in the dataset is scrutinized. The frequencies of each item with respect to tids are listed out. Each item is combined with other item known as join step. Initially, consider the items 'a' and 'b'. The frequency of combination 'a' and 'b' is arrived in the vertical layout. The common TID between the vertical layouts are listed out and then count is noted. The count of {a, b: 1}. Then, the combination of items 'a' and 'c' are taken. The frequency for this combination arrived in vertical layout. The common TID between the vertical layouts are listed and the count is noted. The count of {a, c: 3}. Likewise, the procedure is repeated for all other combinations. In the next phase, an idea of pruning is applied. Minimum support less than 3 is getting eliminated. In this illustration, the itemset {a, c} {a, f} {c, f} are taken for next stage as they satisfy minimum support. The same step is repeated for comparing all other combinations. Finally, the frequent patterns are : {a, c, f: 3} {c, f: 3}.

Algorithm 1: Algorithm of ECLAT

```
Input: F = {I_1..I_n} frequent k Itemsets
Terminology:
  (i) $F_k$ is defined as database having $F_k = \{I_1, I_2,.., In\}$
  (ii) Φ denotes the itemsets where itemset means collection of items in database $F_k$
  (iii) $I_i$ and $I_j$ both should be from same equivalence Class
Output: $F_{|R|}$ Frequent Item Sets

  Bottom-Up ($F_k$):
      Step 1: for all $I_i \in F_k$ do
      Step 2: $F_{k+1} = \phi$;
      Step 3: for all $I_j \in F_k$, i < j do
      Step 4: $N = I_i \cap I_j$ ; // Both should be from
              same equivalence class
      Step 5: if N.sup >=minsup then
      Step 6: $F_{k+1}=F_{k+1} \cup \{N\}$; $F_{|R|} = F_{|R|} \cup \{N\}$
      Step 7: end;
      Step 8: if $F_{k+1}$ != $\phi$; then
      Step 9: Bottom-Up ($F_{k+1}$);
      Step 10: end;
```

Table I. Transaction Database

TID	Item bought
1	d, c, a, f
2	g, c, a, f, e
3	b, a, c, f, h
4	g, b, f
5	c, b

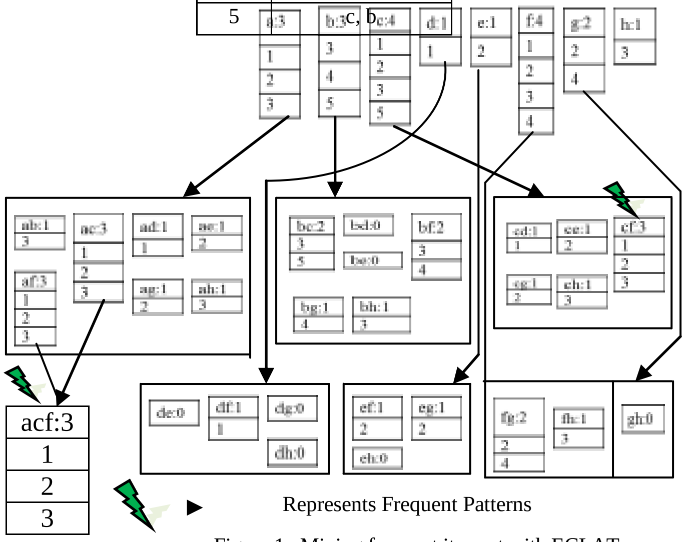


Figure 1. Mining frequent itemset with ECLAT

B. FP-growth method

Even though ECLAT is improvised algorithm of Apriori, it still has some limitation like the requirement of virtual memory to process the transaction. To overcome the limitation of the above described algorithm FP-growth algorithm is introduced. FP-growth algorithm discovers frequent itemsets without generating candidate itemset [12]. It has two major steps

1. Construct a compact data structure called FP-tree. Build two passes over the database.
2. Discover frequent itemsets directly from the FP-tree.

Step 1: FP-Tree Construction

FP-Tree is constructed using two passes over the data-set:

Pass 1:

- Scrutinize the database and explore support for each item.
- Remove infrequent items.
- Based on items support sort items in decreasing order.
- While constructing FP-tree use this above procedure.

Pass 2:

Nodes correspond to items and have a counter

- FP-Growth reads first transaction at a time and maps it to a path.
- Fixed order is used, so paths can overlap when transactions share items.
  — In this case, counters are incremented.
- Pointers are used between nodes to manage the same items.
- FP-tree may fit in memory.

Step 2: Generation of Frequent Itemset

- Frequent itemsets are extracted from FP-tree.
- Bottom-up algorithm is used from the bottom towards the top.
- Divide and conquer method is used.

Fig. 2 shows the complete FP-tree which is constructed by following the above steps using the database portrayed in Table I. Table II displays the frequent itemsets generated from the Fig. 2. Steps involved in FP-growth are given in Algorithm 2.

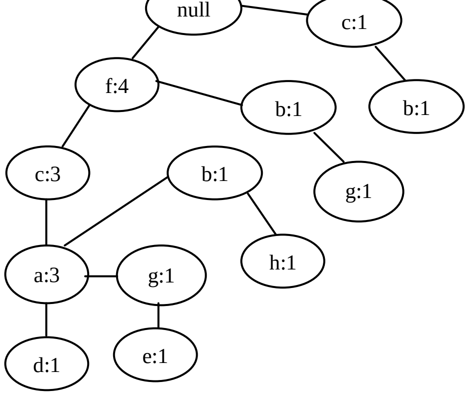


Figure 2. Complete FP- tree

Table II: Generated Frequent Itemsets

Items	Conditional pattern base	Conditional FP-tree	Frequent patterns generated
h	{f, c, b, a:1}	-	No
e	{f, c, a, g:1}	-	No
d	{ f, a, c:1}	-	No
g	{f,b:1} {f, c, a:1}	{f:2}	No
b	{c:1}{f:1} {f, c, a:1}	{f, c:2}	No
a	{f, c:3}	{f,c:3}	{f,c,a:3}
c	{f:3}	{f:3}	{f, c:3}
f	-	-	No

Algorithm 2: Algorithm of FP-Growth

```
Input: D – Transaction database
       s – Minimum support threshold
Output: The complete set of frequent pattern
Function:
        Call FP-tree (FP-tree, null).
        Procedure FP-growth (Tree, A)
      {
Step 1: If Tree contains a single path P, then
Step 2: For each combination (denoted as B) of the nodes
      in the path P Generate pattern B∪A with
      support count = minimum support of nodes in B
Step 3: Else for each $a_i$ (Item) in the header of the Tree
     do
     {
Step 4: Generate pattern B = $a_i$∪A with support = ai.support;
Step 5: Construct B's conditional pattern base and
        B's conditional FP-tree $Tree_B$;
Step 6: If $Tree_B \neq Ø$ then
Step 7: Call FP-growth ($Tree_B$, B) }
```

IV. METHODOLOGY AND RESULTS

In this research paper, the performances of ECLAT and FP-growth algorithms are compared. The ECLAT algorithm generates candidate itemset whereas FP-Growth algorithm does not generate candidate itemset. The experiment is conducted on Intel® corei3™ CPU, 2.13 GHz, and 2GB of RAM computer. Implementation is done in Java. The proposed methodology to generate frequent itemset is shown in Fig. 3.

A. Dataset Description

In this work, eleven standard datasets from different applications are used. These datasets have been obtained from Tunedit Machine Learning Repository. Each datasets contains different instance and attribute. Mushroom accommodates 8124 instances and 23 attributes i.e. cap shape, cap surface, cap color, class etc. Supermarket contains 4627 transactions and 217 attributes especially Grocery, baby needs, coupons, breakfast food etc. German consists of 1000 instances and 21 attributes especially personal status and sex, telephone, housing, property etc. Primary tumor accommodates 339 transactions and 18 attributes i.e. brain, skin, neck, abdominal, liver, age, sex etc.

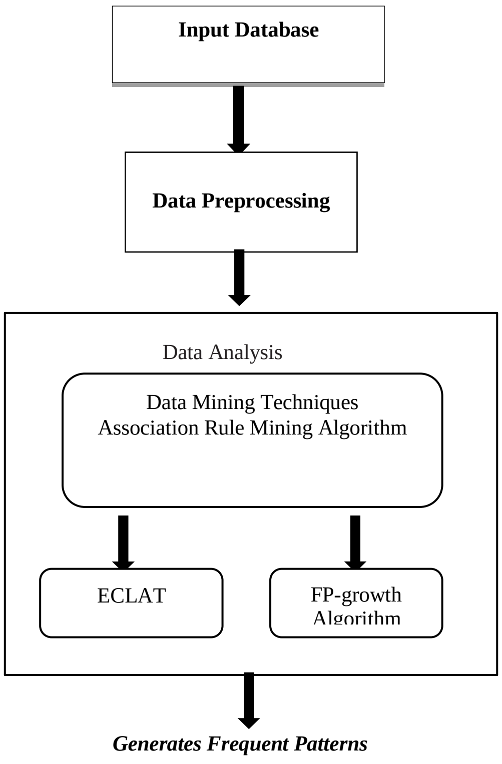


Figure 3. Methodology to Generate Frequent Items

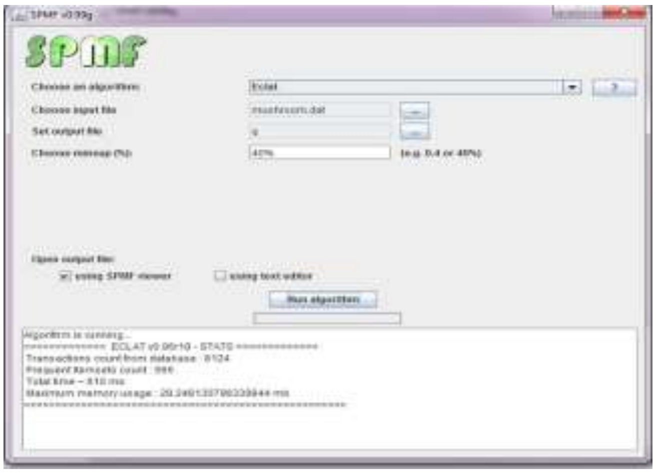
B. Performance Comparison

Experiments are conducted for the analysis to appraise the conduct of FP-growth and ECLAT algorithms. The performance of experiment is measured using total execution time and memory usage for generating itemset. In this comparison distinct dataset with different threshold support values are considered.

Table III: Execution of Runtime

Datasets	**Runtime in milliseconds**		
	Support	***ECLAT***	***FP-Growth***
Breast Cancer	0.1	31	16
Supermarket	0.2	237	203
Customer data	0.2	3	4
German	0.3	93	63
Weather	0.4	5	100
Spect	0.4	31	31
Mushroom	0.4	610	219
Vote	0.5	31	16
Splice	0.6	156	78
Primary Tumor	0.7	15	47
Test	0.8	47	62
Average		**1259**	**839**

Table III depicts the execution time of ECLAT and FP-growth with different datasets with discrete support threshold values. The processing time of ECLAT for the dataset of breast cancer is 31ms and FP-growth is 16ms for the support value of 0.1. The runtime of customer dataset for ECLAT is 3ms and FP-growth is 2ms for the support value of 0.2. The average execution time of ECLAT algorithm for all databases is 1259ms whereas the average execution time of FP-growth algorithm for all databases is 839ms. It is inferred that the execution time of FP-growth algorithm without candidate itemset generation gives better performance than ECLAT algorithm which generate candidate itemset.



Choose an algorithm: Eclat
Choose input file: mushroom.dat
Set output file:
Choose minsup (%): 40% (e.g. 0.4 or 40%)
Open output file: [✓ using SPMF viewer] using text editor

```
Algorithm is running...
============= ECLAT v0.96r10 - STATS =============
Transactions count from database : 8124
Frequent itemsets count : 986
Total time ~ 810 ms
Maximum memory usage : 20.24013379833944 mb
===================================================
```

Figure 4. Input Layout

With reference to Fig. 4, the type of algorithm and dataset are chosen first. Minimum support threshold value of 40% is specified by the user which is used for the generation of frequent patterns. It includes the preference for layout of viewer to display the results containing frequent patterns. It also holds the information of total runtime and memory required and the count of number of frequent itemsets.

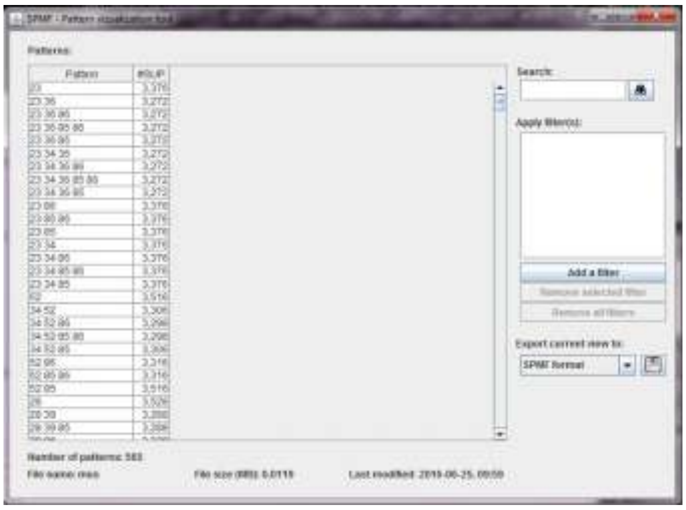


Figure 5. Results of Frequent patterns found using ECLAT for mushroom dataset

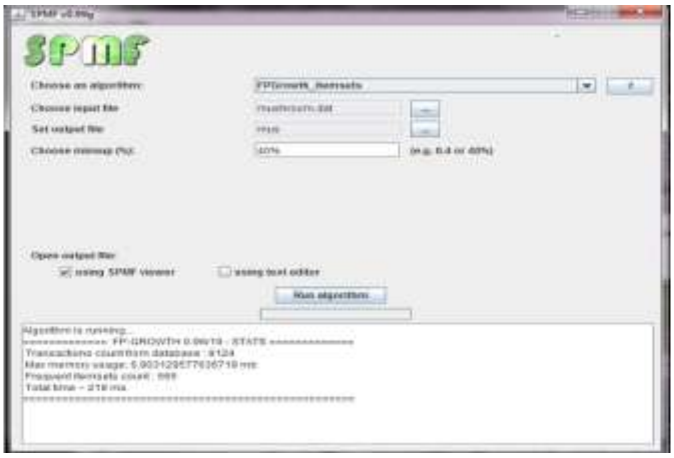
Fig. 5 illustrates the visualization of frequent patterns of ECLAT algorithm for mushroom based on minimum support threshold value.

Table IV represents comparative results for memory usage of ECALT and FP-Growth algorithms for different datasets. Usage of memory of ECLAT for the dataset breast cancer is 60.56MB and FP-Growth is 53.07MB for the support threshold value of 0.1. Memory space required for ECLAT is

43.67MB and FP-Growth is 23.51MB for the dataset customer data with support threshold value of 0.2. The average memory usage of ECLAT algorithm for all databases is 502.40MB whereas the average memory usage of FP-growth algorithm for all databases is 363.99MB. It is found that memory usage of FP-growth algorithm without candidate itemset generation gives better performance than ECLAT algorithm which generates candidate itemset.

Table IV: Memory usage

Datasets	Memory usage in MB		
	Support	*ECLAT*	*FP-Growth*
Breast Cancer	0.1	60.56	53.07
Supermarket	0.2	29.39	18.93
Customer data	0.2	43.67	23.51
German	0.3	20.75	12.85
Weather	0.4	9.73	7.73
Spect	0.4	27.26	26.64
Mushroom	0.4	28.24	6.9
Vote	0.5	35.91	35.97
Splice	0.6	93.88	30.39
Primary Tumor	0.7	77.08	71.97
Test	0.8	75.93	76.03
Average		**502.4**	**363.99**



```
Algorithm is running...
============= FP-GROWTH 0.96r19 - STATS =============
Transactions count from database : 8124
Max memory usage: 5.903129577636719 mb
Frequent itemsets count : 565
Total time ~ 218 ms
===================================================
```

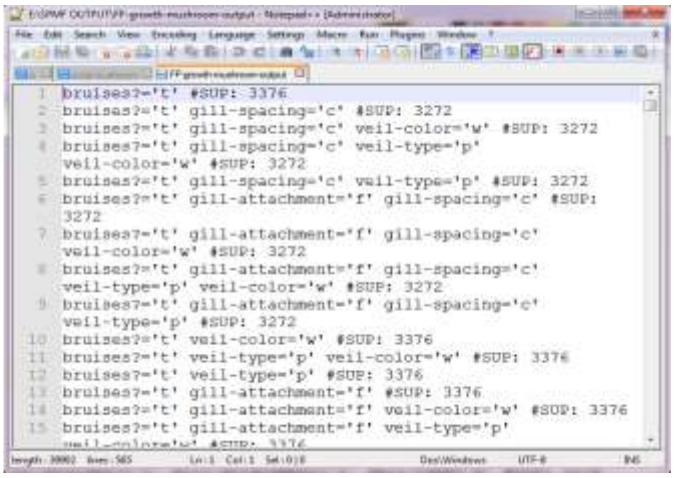
Figure 6. Input Layout

Fig. 6 portrays the option for choosing the type of algorithm and dataset. Any threshold value of minimum support can be stated by the user. Based on threshold value frequent patterns are generated. It has the option to point out the type of viewer to display the results containing frequent patterns. It also displays the total execution time and memory and the count of number of frequent itemsets.

Visualization of frequent patterns is generated using the layout of text editor for the dataset of mushroom is displayed in Fig. 7.

Fig. 8 shows the runtime of ECLAT algorithm and FP-growth algorithm for all databases. Here, y-axis shows runtime in millisecond x-axis shows the various support threshold values.

Fig. 9 shows the memory usage of ECLAT algorithm and FP-growth algorithm for all databases. Here, y-axis shows memory in MB and x-axis shows the various support threshold values.


E:\SPMF OUTPUT\FP-growth-mushroom-output - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
FP-growth-mushroom-output

```
bruises?='t' #SUP: 3376
bruises?='t' gill-spacing='c' #SUP: 3272
bruises?='t' gill-spacing='c' veil-color='w' #SUP: 3272
bruises?='t' gill-spacing='c' veil-type='p'
veil-color='w' #SUP: 3272
bruises?='t' gill-spacing='c' veil-type='p' #SUP: 3272
bruises?='t' gill-attachment='f' gill-spacing='c' #SUP:
3272
bruises?='t' gill-attachment='f' gill-spacing='c'
veil-color='w' #SUP: 3272
bruises?='t' gill-attachment='f' gill-spacing='c'
veil-type='p' veil-color='w' #SUP: 3272
bruises?='t' gill-attachment='f' gill-spacing='c'
veil-type='p' #SUP: 3272
bruises?='t' veil-color='w' #SUP: 3376
bruises?='t' veil-type='p' veil-color='w' #SUP: 3376
bruises?='t' veil-type='p' #SUP: 3376
bruises?='t' gill-attachment='f' #SUP: 3376
bruises?='t' gill-attachment='f' veil-color='w' #SUP: 3376
bruises?='t' gill-attachment='f' veil-type='p'
veil-color='w' #SUP: 3376
```

```
length : 39802    lines : 565        Ln : 1    Col : 1    Sel : 0 | 0        Dos\Windows    UTF-8    INS
```

Figure 7. Results of Frequent patterns found using FP-growth for mushroom dataset

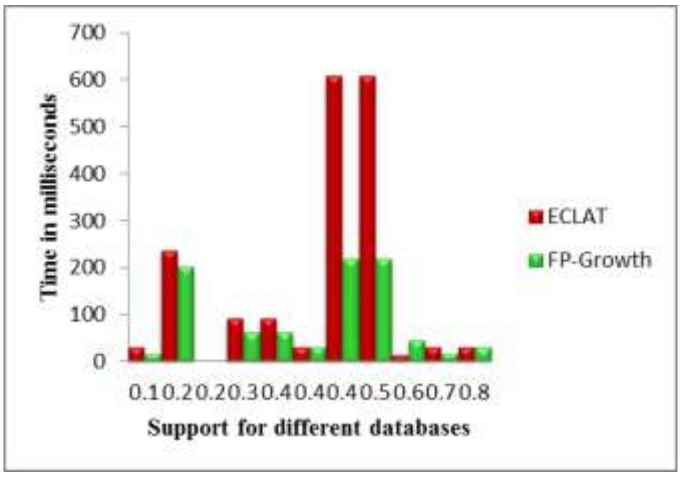


Figure 8. Runtime for ECLAT and FP-Growth

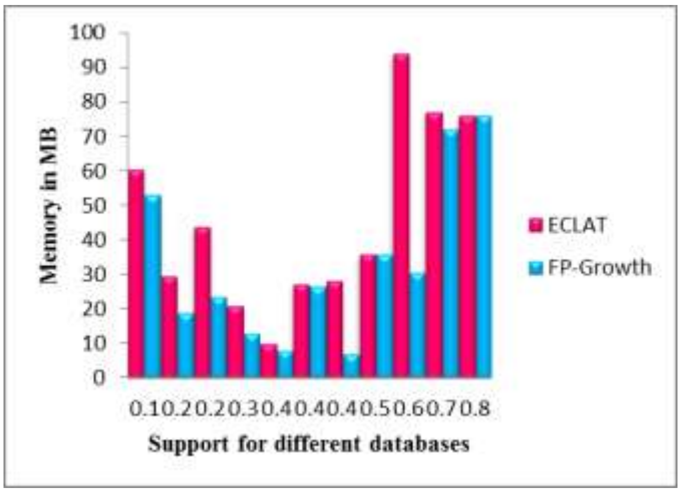


Figure 9. Memory usage based on ECLAT & FP-growth

V. CONCLUSION

In this paper, a detailed contrast has been developed for the frequent pattern mining algorithms of equivalence class clustering and bottom up lattice traversal (ECLAT) and FP-growth. The comparison has been analyzed using numerous standard datasets like mushroom, spect, primary tumor, vote etc. The result shows that the algorithm runtime and memory differs for different datasets. ECLAT algorithm accommodates 'Depth First Search' strategy and requires generating candidate itemset. It doesn't require scanning the database each time. The FP-growth algorithm adopts the 'Divide and Conquer' method and does not require candidate itemset generation. It doesn't undergo repeated scans of the data.

It is found from the experiment on different datasets that the performance of FP-growth algorithm is outstanding than ECLAT in both categories of runtime and memory.

REFERENCES

[1] H. Mannila, Database methods for data mining, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD '98) tutorial, New York, NY, USA, 1998.

[2] 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.

[3] Sotiris Kotsiantis, Dimitris Kanellopoulos Association Rules Mining: A Recent Overview GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), pp. 71-82, 2006.

[4] Imielienskin T. and Swami A. Agrawal R., "Mining Association Rules Between set of items in largedatabases," in Management of Data, 1993, p. 9.

[5] R.Agrawal, R.Srikant, (1994) 'Fast algorithms for mining association rules', Proceedings of the 20th Very Large Databases Conference (VLDB'94), Santiago de Chile, Chile.

[6] M. Chen, and P.S. Yu J.S. Park, "An Effective Hash Based Algorithm for Mining Association Rules," in ACM SIGMOD Int'l Conf. Management of Data, May, 1995

[7] Sanjeev Rao, Priyanka Gupta, "Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm," International Journal of Computer Science and Technology, Vol. 3, Iss ue 1, Jan. - March 2012.

[8] Dr. Telkapalli Murali Krishna, "Effectiveness of various FPM Algorithms in Data Mining," International Journal of Computing Science and Information Technology, Vol.02 (01), 01-05, 2014.

[9] Shamila Nasreen, Muhammad Awais Azam, Khurram Shehzad, Usman Naeem, "Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey," Elsevier, The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, 109 – 116, 2014.

[10] Urvashi Garg, " Advanced eclat algorithm for frequent itemsets generation," International Journal of Applied Engineering Research , 2015.

[11] Manjit kaur, Urvashi Grag, "ECLAT Algorithm for Frequent Itemsets Generation ," International Journal of Computer System, Volume 01–Issue 03, December, 2014.

[12] Sankalp Mitra, Suchit Bande, Shreyas Kudale, Advait Kulkarni, "Efficient FP Growth using Hadoop-(Improved Parallel FP-Growth)," International Journal of Scientific and Research Publications, Volume 4, Issue 7, July 2014.

[13] Akshita Bhandari, Ashutosh Gupta, Debasis Das, "Improvised apriori algorithm using frequent pattern tree for real time applications in data mining," International Conference on Information and Communication Technologies, 644 – 651, 2015.

[14] Sujatha Dandu, B.L.Deekshatulu & Priti Chandra Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree, Global Journal of Computer Science and Technology Software & Data Engineering olume 13 Issue 2 Version 1.0 Year 2013.

[15] H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri R. Agrawal, "Fast Discovery of Association Rules," in Advances in Knowledge Discovery and Data Mining, 1996, pp. 307-328.

[16] M. Chen, and P.S. Yu J.S. Park, "An Effective Hash Based Algorithm for Mining Association Rules," in ACM SIGMOD Int'l Conf. Management of Data, May, 1995.

[17] M. Sinthuja, N. Puviarasan and P. Aruna, "Evaluating the Performance of Association Rule Mining Algorithms," in World Applied Sciences Journal 35 (1): 43-53, 2017.