Pak. J. Biotechnol. Vol. 13 special issue II (International Conference on Engineering and Technology Systems (ICET'16) Pp. 77 - 82 (2016)

EXPERIMENTAL EVALUATION OF APRIORI AND EQUIVALENCE CLASS CLUSTERING AND BOTTOM UP LATTICE TRAVERSAL (ECLAT) ALGORITHMS

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

Department of CSE, Annamalai University, Chidambaram, India
sinthujamuthu@gmail.com, arunapuvi@yahoo.co.in, npuvi2410@yahoo.in

ABSTRACT

Frequent pattern mining is the beginning of association rule mining. Association rule mining is the strongly scrutinized techniques in data mining. The basic algorithms of Apriori and ECLAT are the most identified algorithms for mining frequent patterns in association rule mining. This paper describes the application of these two algorithms that use many to achieve maximum efficiency with regards to turnaround time and memory capacity. Both algorithms are executed using discrete data sets and are further analyzed based on their performances. The performance analysis is based on different parameters such as support, speedup etc., with different quantities of datasets.

Keywords— Apriori Algorithm; Association Rule Mining Algorithm; Data mining; ECLAT, Minimum support; Pruning.

I. INTRODUCTION

The vital logic that people were attracted by IT is the discovery of useful information from huge collection of data industry towards the domain of Data Mining [1, 2]. From the huge data, we barely explore useful knowledge for decision analysis in the business. Vast collection of data can be in distinct formats like audio, video, numbers, text, figures and hypertext formats. To perform data mining task expertise and learning are fundamental need because the victory and loss of data mining projects is highly dependent on the person who are administrating the procedure due to lack of standard protocol. The lifecycle of data mining is of six steps they are Data cleaning, Data integration, Data Selection, Data transformation, Data Mining, Knowledge discovery shown in Fig. 1.

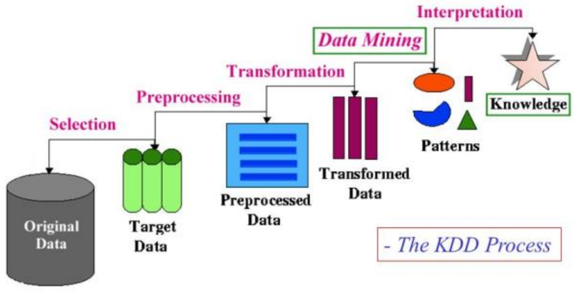


Fig. 1. Process of KDD

Among the techniques of data mining (classification, clustering, prediction and sequential pattern discovery, etc.) The most important one is the mining association rules [3, 8]. The goal of association rule mining is to explore frequent patterns.

Two Step Approaches of Association Rule

i. Frequent Itemset Generation
ii. Rule Generation

Two basic attributes of Association Rule Mining (ARM) are

i. Support
ii. Confidence

Support(*s*) of an association rule is described as the fraction of records that contains the assortment of both anterior and posterior to the overall transaction in the database collection.

Confidence(c) is defined as the proportion of the number of transaction that encompasses anterior and after the overall records that contain D.

Frequent pattern mining, which is the most important field in association rule mining, was first introduced for Market Basket Analysis [4]. The goal of frequent pattern mining is to discover frequent patterns whose support is greater than or equal to the minimum support threshold. Pattern mining algorithm can be enforced on various data such as transaction databases etc. Frequent Patterns are itemsets, substructures that appear in a database with high frequency. They are Candidate generation and Pattern growth. Various types of algorithms used in association rule are shown in Fig. 2.

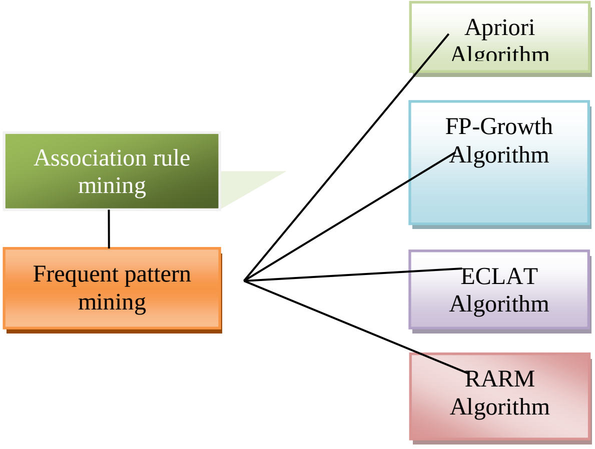


Fig. 2. Various types of algorithms used in association rule

Let us consider the case of the supermarket where the owner may not have input of performance of a product in the market or the most sought product. The data obtained may be used to determine the profitability of a product. In such cases, frequent pattern mining algorithms are applied.

II. LITERATURE SURVEY ON FREQUENT PATTERN MINING ALGORITHMS

Numerous techniques have been experimented for mining association rules in the research studies [13] [14, 15]. 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.

Based on Apriori algorithm, a numerous algorithm has been worked out with some improvements or adjustments 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) [15], Partition algorithm, Sampling algorithm, CARMA (Continuous Association Rule Mining algorithm) [15], DIC algorithm (prefix tree data structure) [15] 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 [6]. It overcomes the bottleneck of apriority algorithm with regards to database scan. Rapid Association Rule mining (RARM) mentioned in [16] 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 [17]. First, it does not generate candidate patterns. Second, database scan is done only twice. It adopts divide and conquer method.

An improved frequent pattern (IFP) growth technique for discovering frequent patterns is proposed [18]. This algorithm requires lower usage of memory and it shows improved results in testing with FP-tree based algorithm.

III. ASSOCIATION RULE MINING ALGORITHMS

Association rule mining adopts the fundamental algorithms of Apriori and ECLAT to discover effective frequent patterns. The comparison is performed for Apriori and ECLAT algorithm in terms of runtime and memory.

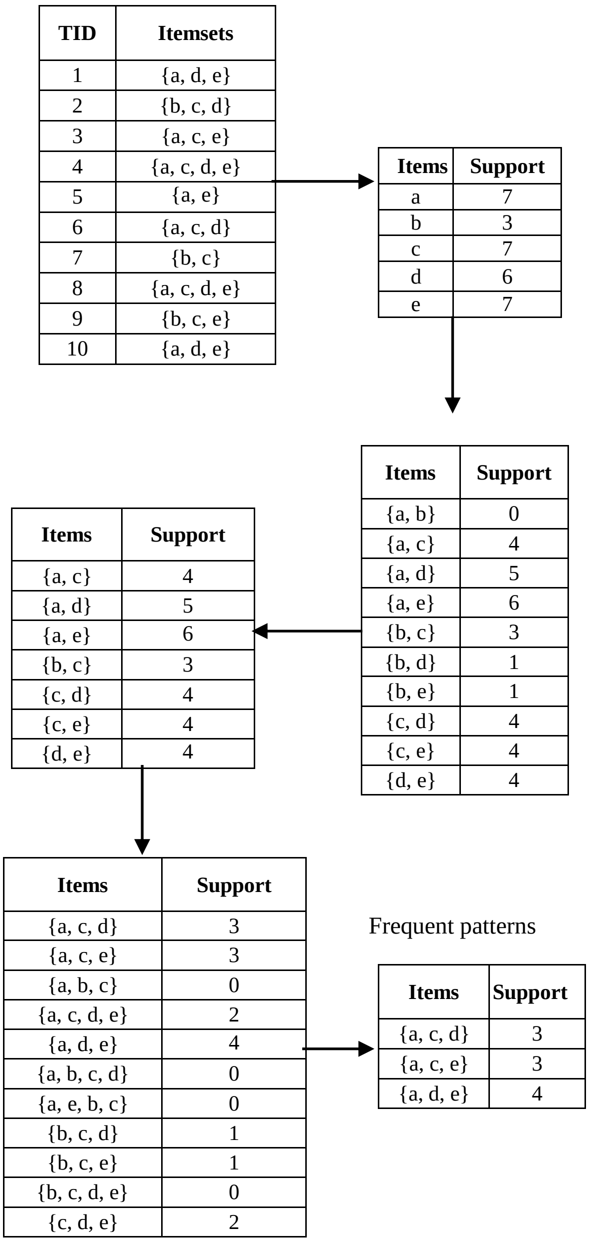
A. Apriori Algorithm

Apriori is the early generated frequent pattern mining algorithm [5, 7]. Apriori apply an iterative approach known as level-wise search [3]. It uses an effective concept called as pruning where pruning eliminates the less occurring items.

To illustrate, the transaction database is shown in Table 1, the database has ten transactions. Fig. 3 portray the generation of frequent patterns using Apriori algorithm. The first stage is analysis of database. By scanning the database, the frequency of each item is found. In the second stage, joining step is performed that is combining each item with the other items. This is called as candidate itemset 1.

Table 1: Transaction Database

Scan the database to count each item



TID	Itemsets
1	{a, d, e}
2	{b, c, d}
3	{a, c, e}
4	{a, c, d, e}
5	{a, e}
6	{a, c, d}
7	{b, c}
8	{a, c, d, e}
9	{b, c, e}
10	{a, d, e}

Items	Support
a	7
b	3
c	7
d	6
e	7

Items	Support
{a, b}	0
{a, c}	4
{a, d}	5
{a, e}	6
{b, c}	3
{b, d}	1
{b, e}	1
{c, d}	4
{c, e}	4
{d, e}	4

Items	Support
{a, c}	4
{a, d}	5
{a, e}	6
{b, c}	3
{c, d}	4
{c, e}	4
{d, e}	4

Items	Support
{a, c, d}	3
{a, c, e}	3
{a, b, c}	0
{a, c, d, e}	2
{a, d, e}	4
{a, b, c, d}	0
{a, e, b, c}	0
{b, c, d}	1
{b, c, e}	1
{b, c, d, e}	0
{c, d, e}	2

Frequent patterns

Items	Support
{a, c, d}	3
{a, c, e}	3
{a, d, e}	4

Fig. 3. Steps to derive frequent item sets using Apriori

Combining the item 'a' with 'b', then 'a' with 'c'. Similarly, combining all the items. Again, scanning the database to find the occurrence of combinations of each item. The count of {a, b} is 0 and {a, c} is 4. Likewise find the support for each combination. In the succeeding stages concept of pruning is performed. Here the minimum support is greater than 2.The itemset greater than 2 is taken for the next step and <= 2 are eliminated. So itemsets of {a, c} {a, d}{a, e}{b, c}{c, d}{c, e}{d, e} are taken for the next step called as frequent itemset 1. In the next stage perform join step to find candidate itemset 2. Again the database is scanned to find the frequency for each itemset. Apply pruning and eliminate the itemset less than 2 and equal to 2.Finally the frequent itemsets are {a,c,d: 3}{a,c,e: 3}{a,d,e: 4}.

The drawback of Apriori algorithm is that if the frequent patterns are longer then, the algorithm must perform more iteration. Thus, the performance of

Minimum Support >2

Apriori reduces. Multiple scanning of database is needed. It is not useful for real time applications. The duration of the protocol is long.

Algorithm 1: Apriori

```
Input:  Database of Transactions D= {$t_1$, $t_{2 ...}$ $t_n$}
        Set if Items I= {$I_1$, $I_2$.... $I_k$}
        Frequent (Large) Itemset L
        Support
        Confidence
Output: Association Rule satisfying Support & Confidence
Method:
    Step 1:  C1 = Itemsets of size one in I;
    Step 2:  Determine all large itemsets of size 1, $L_1$;
    Step 3:  i = 1;
    Step 4:  Repeat
    Step 5:  i = i + 1;
    Step 6:  $C_i$ = Apriori-Gen($L_i$-1);
    Step 7:  Apriori-Gen($L_i$-1)
    Step 8:  Generate candidates of size i+1 from
             large    itemsets of size i.
    Step 9:  Join large itemsets of size i if they
             agree on i-1.
    Step 10: Prune candidates who have subsets
             that are not large.
    Step 11: Count $C_i$ to determine $L_i$;
```

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

Improvised algorithm of Apriori is Equivalence Class Clustering and Bottom up Lattice Traversal (ECLAT) [6,7,8]. ECLAT proposed in [11,12]. ECLAT overthrown the limitation of Apriori in case of database scan, it requires only one database scan. This algorithm converts horizontal database to vertical layout where the Apriori algorithm uses horizontal database [9, 10]. Like Apriori algorithm it has the concept of candidate generation and pruning.

Fig. 4 illustrates the construction of ECLAT with sample database. Table 2 represents the sample database which is horizontal in structure. The first challenge is to convert the horizontal database into a vertical one which is shown in Figure 3. Each item is simulated by group of transaction ids which is called tidset and finding the occurrence of each item.

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 another item known as join step. Let us consider items 'a' and 'b'. The frequency of combination 'a' and 'b' is found out in the vertical layout. There is no occurrence of combination 'a' and 'b' so its frequency is {a, b: 0}. Let us consider the combination of 'a' and 'c'. The common tid between the vertical layouts are listed out and count noted down. The count of {a,c: 4}. Similarly, the process is repeated for all other combination.

In the next stage, the concept of pruning is applied i.e minimum support less than or equal to 2 is eliminated. In this case, itemset {a, b} {b, d} {b, c} gets eliminated. The same procedure is repeated for comparing all the combinations to get trio combinations and so on. Finally, the most frequent patterns are identified. The most frequent patterns are {a,c,d,3} {a,c,e,3}{a,d,e,4}.

Algorithm 2: ECLAT

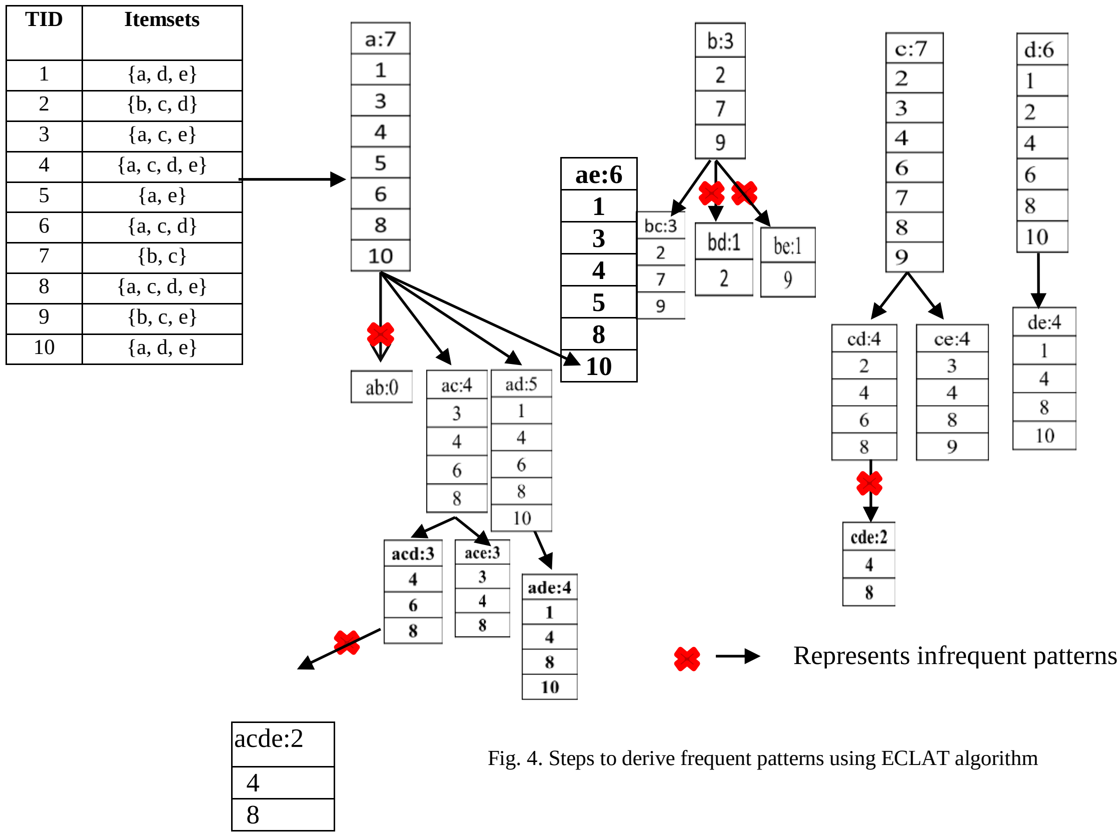
```
Input: F = {$I_{1..}$.$I_n$} frequent k Itemsets
Terminology:
 (i)   $F_k$ is defined as database having $F_k = \{I_1, I_2,$
       ..., $I$n}
 (ii)  Φ denotes the itemsets .where itemsets means collection of items in database $F_k$
 (iii) $I_i$ and $I_j$ both should be from same
           equivalence
       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: $F_{k+1}$=$F_{k+1}$ ᵕ{N}; $F_{|R|}$ = $F_{|R|}$ᵕ{N}
    Step 7: end;
    Step 8: if $F_{k+1}$ != ϕ; then
    Step 9: Bottom-Up ($F_{k+1}$);
    Step 10: end;
```

The drawback of ECLAT is the requirement of virtual memory to process the transaction.

Table 2: Transaction DB

Vertical layout of the initial database

Each item is symbolized by set of transaction ids called tidset



TID	Itemsets
1	{a, d, e}
2	{b, c, d}
3	{a, c, e}
4	{a, c, d, e}
5	{a, e}
6	{a, c, d}
7	{b, c}
8	{a, c, d, e}
9	{b, c, e}
10	{a, d, e}

Fig. 4. Steps to derive frequent patterns using ECLAT algorithm

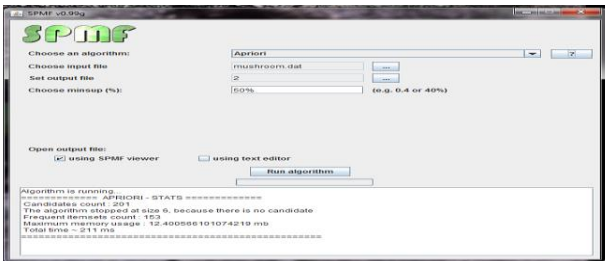
IV. PERFORMANCE EVALUATION

A. Dataset Description

The above specified algorithms are implemented using different standard datasets of different domains namely mushroom, supermarket, German, Eucalyptus, Primary tumor where it is available in Tunedit Machine Learning Repository. 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. Eucalyptus contains 736 transactions and 20 attributes namely locality, latitude, altitude and year etc.

B. Experimental Results

In this research paper, we compare the performances of ECLAT and Apriori 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**.**



Choose an algorithm: Apriori

Choose input file mushroom.dat

Set output file 2

Choose minsup (%): 50% (e.g. 0.4 or 40%)

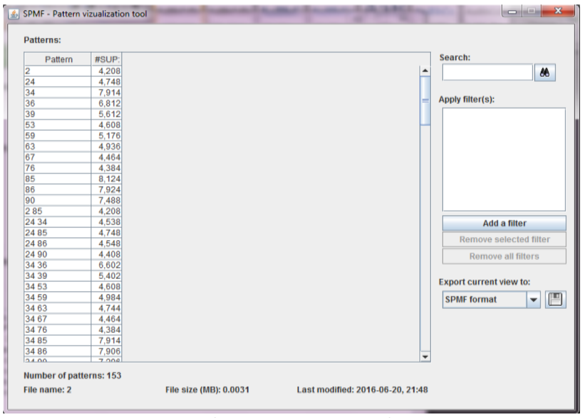
Open output file:

☑ using SPMF viewer ☐ using text editor

```
Algorithm is running...
============= APRIORI - STATS =============
 Candidates count : 201
 The algorithm stopped at size 6, because there is no candidate
 Frequent itemsets count : 153
 Maximum memory usage : 12.400566101074219 mb
 Total time ~ 211 ms
===================================================
```

Fig. 5. Graphical user interface

Fig. 5 illustrates the graphical user interface of Apriori which displays the details of minimum support, dataset and algorithm used. It displays the amount of time required to execute the algorithm, utilization of memory space and the number of frequent patterns.



Patterns:

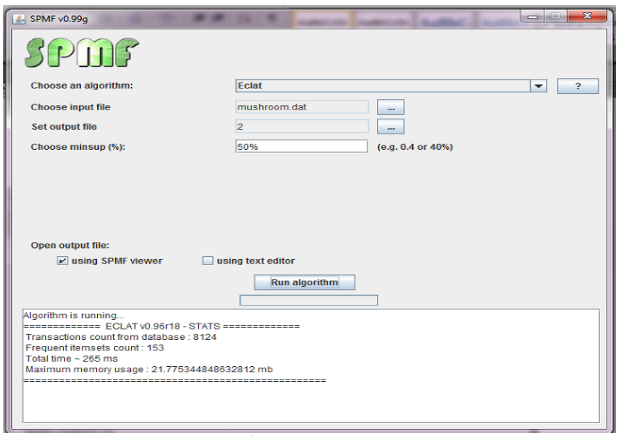
Pattern	#SUP:
2	4,208
24	4,748
34	7,914
36	6,812
39	5,612
53	4,608
59	5,176
63	4,936
67	4,464
76	4,384
85	8,124
86	7,924
90	7,488
2 85	4,208
24 34	4,538
24 85	4,748
24 86	4,548
24 90	4,408
34 36	6,602
34 39	5,402
34 53	4,608
34 59	4,984
34 63	4,744
34 67	4,464
34 76	4,384
34 85	7,914
34 86	7,906

Number of patterns: 153

File name: 2 File size (MB): 0.0031 Last modified: 2016-06-20, 21:48

Fig. 6. Frequent Patterns found using Apriori for mushroom dataset

The frequent patterns generated by Apriori algorithm using mushroom dataset is shown in Fig. 6. Frequent patterns explored from the experiment are based on support.



Choose an algorithm: Eclat
Choose input file mushroom.dat
Set output file 2
Choose minsup (%): 50% (e.g. 0.4 or 40%)
Open output file:
☑ using SPMF viewer ☐ using text editor
Run algorithm

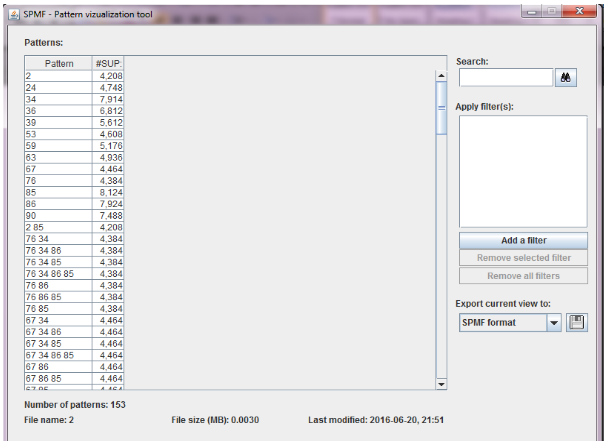
```
Algorithm is running...
============= ECLAT v0.96r18 - STATS =============
 Transactions count from database : 8124
 Frequent itemsets count : 153
 Total time ~ 265 ms
 Maximum memory usage : 21.775344848632812 mb
=======================================================
```

Fig. 7. Graphical user interface

Graphical user interface of ECLAT is displayed in Fig. 7 which contains the information of user specified minimum support, type of dataset and algorithm used. It shows the runtime taken to execute the algorithm, usage of memory space and the total number of frequent patterns.

Regarding Fig. 8, the frequent pattern generated by ECLAT algorithm using mushroom dataset is displayed. Frequent patterns explored from the experiment are based on support.

Table 3 depicts the runtime of each algorithm with different datasets. The processing time of mushroom dataset for Apriori is 211ms and for ECLAT is 265ms. The processing time of supermarket dataset for Apriori is 30ms and ECLAT is 35ms. The processing time of german dataset for Apriori is 1211ms and ECLAT is 303ms. As the minimum support increases ECLAT is poorer in runtime when compared to Apriori. While minimum support decreases Apriori perform faster than ECLAT in terms of execution time.



Patterns:

Pattern	#SUP:
2	4,208
24	4,748
34	7,914
36	6,812
39	5,612
53	4,608
59	5,176
63	4,936
67	4,464
76	4,384
85	8,124
86	7,924
90	7,488
2 85	4,208
76 34	4,384
76 34 86	4,384
76 34 85	4,384
76 34 86 85	4,384
76 86	4,384
76 86 85	4,384
76 85	4,384
67 34	4,464
67 34 86	4,464
67 34 85	4,464
67 34 86 85	4,464
67 86	4,464
67 86 85	4,464

Number of patterns: 153
File name: 2 File size (MB): 0.0030 Last modified: 2016-06-20, 21:51

Fig. 8. Frequent Patterns found using ECLAT for mushroom dataset

Table 3: Execution time of instances

Datasets	Runtime in milliseconds		
	Support	Apriori	ECLAT
Mushroom	0.4	211	265
Supermarket	0.4	30	35
German	0.1	1211	303
Eucalyptus	0.1	102	32
Primary tumor	0.3	334	162
Average	1888		797

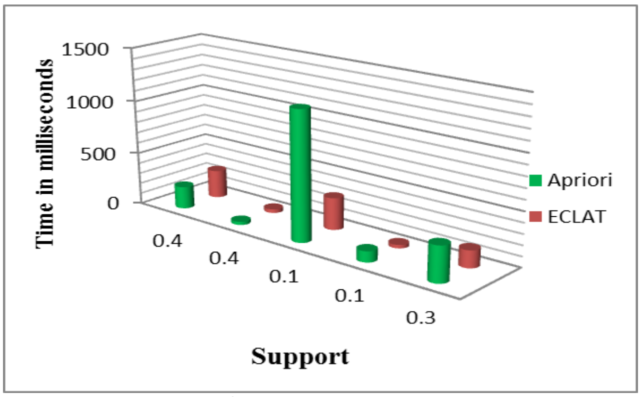


Fig. 9. Runtime of Apriori and ECLAT

Fig. 9 illustrates the performance comparison of Apriori and ECLAT algorithms with different support threshold value for different datasets. Runtime of Apriori is faster than ECLAT in the case of higher minimum support threshold value. While in the case of lower minimum support Apriori is poor runtime when compared to ECLAT.

Table 4 depicts the memory usage of both algorithms with respect to different datasets. Memory space required for Apriori is 12.40mb and ECLAT is 21.77 mb for mushroom dataset. Usage of memory for Apriori is 8.90mb and ECLAT is 17.01mb for supermarket dataset.

Table 4. Memory usage

Datasets	Memory Usage in mb	
	Apriori	ECLAT
Mushroom	12.40	21.77
Supermarket	8.90	17.01
German	25.43	22.97
Eucalyptus	93.03	89.28
Primary tumor	105.51	108.33
Average	255.59	265.28

Fig. 10 portrays the comparative results of Apriori and ECLAT in case of memory usage. Apriori requires less memory than ECLAT in case of maximum minimum support. While using lower minimum support Apriori requires more memory space than ECLAT.

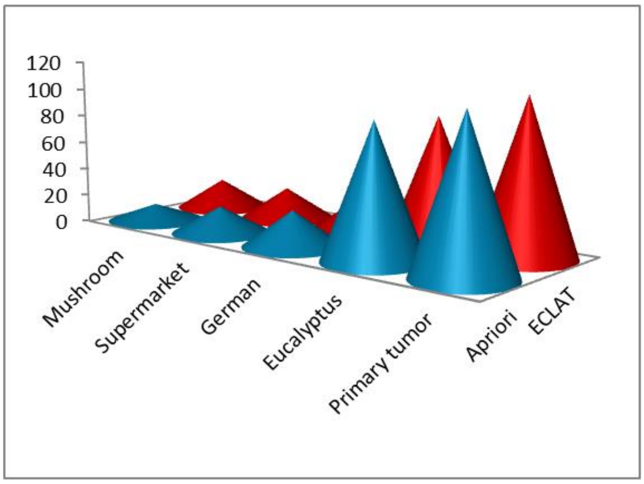


Fig. 10. Memory usage based on Apriori and ECLAT

IV. CONCLUSION

In this paper the researcher surveyed frequent pattern mining algorithms namely Apriori and Equivalence Class Clustering and Bottom Up Lattice Traversal (ECLAT) with distinct database. The goal of Apriori and Equivalence Class Clustering and Bottom up Lattice Traversal is to explore frequent patterns. It is found that Apriori used join and prune technique and ECLAT works on vertical datasets. The major drawback of Apriori is generating large number of candidate patterns and huge number of database scan. The major limitation of ECLAT requires virtual memory to process the transaction.

The average runtime of Apriori and ECLAT shows that Apriori performs worst in runtime when compared to ECLAT in case of different minimum support. The memory usage of both algorithms results that ECLAT is poorer than Apriori.

REFERENCES

[1] Divya Tomar and Sonali Agarwal, A survey on Data Mining approaches for Healthcare. International Journal of Bio-Science and Biotechnology 5(5): 241-266 (2013).

[2] Kittipol Wisaeng, An Empirical Comparison of Data Mining Techniques in Medical Database. International Journal of Computer Application (2012)

[3] Neelamadhab Padhy, Pragnyaban Mishra and Rasmita Panigrahi, The Survey of Data Mining Applications And Feature Scope. International Journal of Computer Science, Engineering and Information Technology 2(3): 43-58 (2012)

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

[5] Pramod S, O.P.Vyas, Performance Evaluation of some Online Association Rule Mining Algorithms for sorted and unsorted Data sets. International Journal of Computer Applications 2(6): 40 - 45 (2010)

[6] M. J. Zaki and K. Gouda. Fast vertical mining using diffsets. Technical report, RPI, 01-1 (2001)

[7] C. Borgelt. Efficient implementations of apriori and eclat, Proceedings of FIMI'03 Workshop on Frequent Itemset Mining Implementation Pp. 1- 9 (2003)

[8] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li., New Algorithms for Fast Discovery of Association Rules, Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), AAAI Press, Menlo Park Pp. 283–296 (1997).

[9] Patel Tushar S., Performance Analysis of Frequent Itemset Finding Techniques using Sparse Datasets, IEEE International Conference on Computer, Communication and Control (2015)

[10] Paresh Tanna and Yogesh Ghodasara, Improved Frequent Pattern Mining Algorithm with Indexing. Journal of Computer Engineering (IOSR-JCE) 16(6): Ver. VII Pp 73-78 (2014)

[11] K.Vani, Comparative Analysis of Association Rule Mining Algorithms Based on Performance Survey. International Journal of Computer Science and Information Technologies 6 (4): 3980-3985 (2015)

[12] Ishita Rana and Amit Rathod, Frequent Item Set Mining In Data Mining: A survey. International Journal of Computer Applications 139(9):15-18 (2016).

[13] Imielienskin T. and Swami A. and R.Agrawal, Mining Association Rules Between set of items in large databases, Management of Data Pp. 9 (1993)

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

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

[16] Wee Keong, Yew Kwong and Amitabha Das, Rapid Association Rule Mining, in Information and Knowledge Management, Atlanta, Georgia Pp. 474-481 (2001)

[17] Sourav S. Bhowmick Qiankun Zhao, Association Rule Mining: A Survey, Nanyang Technological University, Singapore (2003)

[18] Ke-Chung Lin, I-En Liao and Zhi-Sheng Chen, An improved frequent pattern growth method for mining association rules. Expert Systems with Applications 38: 5154–5161 (2011)