COMPARATIVE ANALYSIS OF ASSOCIATION RULE MINING ALGORITHMS

Dr. S.Vijayarani[1], *Assistant Professor, Department of Computer Science, Bharathiar University, Coimbatore, vijimohan_2000@yahoo.com,*

Ms. S.Sharmila[2] *Ph.D Research Scholar Department of Computer Science Bharathiar University, Coimbatore sharmilasathyanathan@gmail.com,*

***Abstract-Data mining is one of the significant research domains in the field of computer science and it is defined as the extraction of hidden knowledge from the large data repositories. Important data mining techniques are classification, clustering, association rule generation, summarization, time series analysis and etc. Association rule is used to determine frequent patterns, association and correlations among sets of items in transactional databases, relational databases, and other information repositories. The two important steps of association rule mining are frequent item generation and association rule generation. Frequent item generation discovers all frequent sets of items; it is defined as itemsets that have at least minimum support. From these frequent items, association rules are generated. The main objective of this research is to analyze the performance of various association rule generation algorithms. The algorithms considered for this comparative analysis are Apriori, Eclat, Dclat, FP-growth, FIN, AprioriTID, Relim and H-Mine. Performance factors used are number of frequent items generated (different thresholds), memory requirements, execution time, number of rules generated (different threshold values of support and confidence) and different sizes of datasets are used. Finally from this comparative analysis we observed that the DCLAT algorithm has produced good results than other algorithms*.**

Keywords- Association rules, Apriori, Dclat, Eclat, FP-growth, FIN, AprioriTID, Relim, H-Mine.

I. INTRODUCTION

Association rule mining is the most efficient data mining technique to discover hidden patterns from large volume of data [7]. Different types of association rule mining algorithms are available. Popular algorithms are apriori, partition, pincer-search, dynamic item set counting, FP-tree growth, H-Mine, FIN, RElim and etc. [8]. An association rule has two factors, an antecedent (if) and a consequent (then). An antecedent is an item found in the data [7]. A consequent is an item that is found in sequence of antecedent. Association rules are discovered by analyzing data for frequent if/then patterns and applying the benchmark support and confidence to describe the most significant correlation. Support indicates how frequently the items appear in the database [7]. Confidence indicates the number of times the if/then statements have been found to be true [9].

In a database of transactions D with a set of n binary items I, a rule is defined as an implication of the form X =>Y where X, Y subset I and $X \cap Y = Ø$ [8]. The sets of items X and Y are called antecedent (left-hand-side) and consequent (right-hand-side) of the rule. The support is defined as supp(X) of an item set X as the proportion of transactions in the data set which contain the item set. The confidence of a rule is defined [8]

$$\text{Support (XY)} = \frac{\text{Support total of XY}}{\text{Total number of transactions D}}$$

$$\text{Confidence (X|Y)} = \frac{\text{Support (XY)}}{\text{Support (X)}}$$

Conf (X =>Y) = supp (XUY) / supp(X). Following the original definition given by Agrawal et al [5], association rules are implication rules that instruct the customers about items that most feasibly occur in transactions of a database [9]. To obtain this, association rule generation has two-steps. First, minimum support is applied to find all frequent item-sets in a database. In a second step, these frequent item-sets and the minimum confidence constraints are used to form rules [10]. At the same time the second step is done directly but further discussions are needed for step one. The main objective of this research work is to analyze the performance efficiency of the eight association rule mining algorithms. The algorithms are Apriori, Eclat, Dclat, FP-growth, FIN, AprioriTID, Relim and H-Mine.

Remaining section of the paper is organized as follows. Section 2 describes the related works. Section 3 illustrates the methodology and discusses various association rule mining algorithms. Section 4 represents results and finally conclusion is given in Section 5.

II. LITERATURE REVIEW

Smita R.Sankhe Kavita Kelkar et.al [11] discussed optimization of execution time for classical Apriori and an improved Apriori algorithm (DFR Direct Fined and Remove) to increase the efficiency of generating association rules. This algorithm adopted a new method to reduce the redundant generation of sub-itemsets during pruning the candidate itemsets, which can form directly the set of frequent itemsets and eliminated candidates having a subset that is not frequent in the meantime. This algorithm had raised the probability of obtaining information in scanning database and reduced the potential scale of item sets.

Patel Tushar S, Panchal Mayur et.al [15] discussed about few unifying features and internal working of various mining algorithms. The strengths and weaknesses of Apriori, DHP, Partitioning, Sampling, DIC, and Eclat, FP-growth, and H-mine algorithms were analyzed [15]. The total execution times of these algorithms were calculated with different support threshold using an adult data set. The execution time is decreased when the support threshold is increased. Author proved that the split and merge (SaM) algorithm is better than other algorithms. The Apriori takes more time as compared to other algorithms.

Amit Mittal, Ashutosh Nagar et.al [16] reviewed about different frequent mining algorithms like Apriori, FPgrowth and DIC. A brief description of each technique has been provided. Different frequent pattern mining techniques are compared based on various parameters and real time dataset. From this work it is observed that FP-growth algorithm gives the better results than other algorithms.

Anitha Modi, Radhika Krishnan et.al [18] discussed about mining frequent itemset in transactional database. The main objective of this comparative analysis is to reduce the number of scans and improve the efficiency. The strength and weakness of Apriori, DHP, Partitioning, Sampling, DIC, Eclat, FP-growth, and H-mine algorithms were analyzed. Finally it had observed that RELIM algorithm works better than all other algorithms because of number generation of candidate sets but require pre-processing on database.

J.R.Jeba Dr.S.P.Victor et.al [19] analyzed the implementations of the frequent item set Mining algorithms such as SMine and Apriori Algorithms. The performances of the algorithms were experimented with different support threshold values. This paper proposed an efficient SMine (Sorted Mine) Algorithm for finding frequent item sets. The proposed method has reduced the number of scans in the database.

III. METHODOLOGY

Figure 1 shows the system architecture of this comparative analysis.

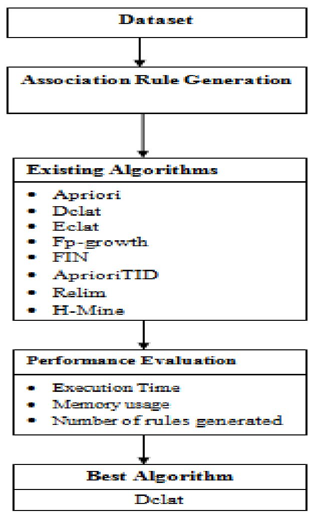

Fig. 1 System Architecture

Dataset-Dataset is generated for three different transactions such as 1000, 5000 and 10000 with 10 items. Performance metrics such as total execution time is calculated in milliseconds, number of association rules and memory usage is calculated in megabytes for different dataset, minimum support and confidence values for all the eight algorithms. Finally from the analysis it is observed that Dclat algorithm is best comparing to all other algorithms.

A. Apriori Algorithm

Apriori Algorithm was first introduced by R. Agrawal. This algorithm is used to discover frequent item set. The main principle of Apriori algorithm is a level-wise search, mine frequent itemsets from transactional database [13]. In this algorithm, frequent subsets are extended one item at a time and this step is known as candidate generation process. Then groups of candidates are tested with the data. Apriori uses breadth-first search method (level-wise search) and a hash tree structure to count candidate item sets efficiently in the search space. There are several key concepts used in Apriori algorithm such as Frequent Itemsets, Apriori Property and Join Operation [13].

The working of Apriori algorithm is adequately depends upon the Apriori property which states that all non-empty subsets of a frequent itemsets must be frequent .It identifies the frequent individual items within the information and extends them to larger item sets as long as those item sets seem to be sufficient in the database. The key idea of Apriori algorithm is to make multiple passes over the database [14][15].

```
The Apriori Algorithm (Pseudo-Code)
Ck: Candidate itemset of size k
Lk : frequent itemset of size k
L1 = {frequent items};
for (k = 1; Lk !=Ø; k++) do begin
      Ck+1 = candidates generated from Lk;
      for each transaction t in database do
              increment the count of all candidates in Ck+1 that are contained in t
      Lk+1 = candidates in Ck+1 with min_support
      end
return Uk Lk;
```

B Eclat Algorithm

Eclat algorithm primarily depends on depth-first search algorithm which uses the set intersection method to find the frequent itemset from the database. It uses a vertical database layout i.e. alternative of explicitly listing all transactions; each item is stored together and uses the intersection based approach to compute the support of an itemset. Hence, the support of an itemset can be easily computed by intersecting any two subsets i.e. ($Z \subseteq X$, such that YU Z = X.).When the database is stored in the vertical layout, the support of a set can be counted much easier by simply intersecting subsets. [18].

In this algorithm each frequent items are added in the output set and new database is created. This is done by first finding every item that frequently occurs together with in itemset [19]. The support is computed by intersecting the both items. If the items are frequent, then that items are inserted into new database. Then the algorithm is called recursive and it is used to find all frequent itemsets in the new database [20].

```
The Eclat Algorithm

Input: D, σ, i ⊆ I
Output: F[I](D, σ)
1: F[I] :={}
2: for all I ∈ I occurring in D do
3: F[I] := F[I] ∪ {I ∪ {i}}
4: // Create Di
5: Di: = {}
6: for all j ∈ I occurring in D such that j>I do
7: C := cover({i}) ∩ cover({j})
8: if |C| >= σ then Di: = Di ∪ {(j, C)}
9: end if
10: end for
11: //Depth-first recursion
12: Compute F[I ∪ {i}](Di, σ)
13: F[I] := F[I] ∪ F[I ∪ {i}]
14: end for
```

D.Dclat Algorithm

Dclat algorithm performs a pure bottom-up search to find the frequent itemset from the transactional database Dclat is a variation of the Eclat algorithm that is implemented using a structure called "diffsets" (the difference of two sets) rather than "tidsets" (set of transaction IDs). It keeps track of only the differences in the tids between each class member and the class prefix itemset. These differences in tids are stored is called as diffset, which is a difference of two tidsets (namely, the prefix tidset and a class member's tidset). Furthermore, these differences are propagated all the way from a node to its children starting from the root. The root node's members can themselves use full tidsets or differences from the empty prefix [26].

Frequent itemsets are generated by computing diffsets for all distinct pairs of itemsets and checking the support of the resulting itemset [27]. A recursive procedure call is made with those itemsets found to be frequent at the current level. This process is repeated until all frequent itemsets have been enumerated. In terms of memory management it is easy to see that it needs memory to store intermediate diffsets for at most two consecutive levels. Once all the frequent itemsets for the next level have been generated, the itemsets at the current level can be deleted [26].

```
DiffEclat ([P]):
for all Xi ∈ [P] do
 for all Xj ∈ [P], with j > i do
R = Xi ∪ Xj ;
d(R) = d(Xj ) − d(Xi);
 if σ(R) ≥ min sup then
          Ti = Ti ∪ {R}; //Ti initially empty
for all Ti 6= ∅ do DiffEclat(Ti)
```

D. FP- growth Algorithm

The main goal of this algorithm is to discard the bottlenecks of the Apriori-Algorithm in generating and testing candidate set. To overcome the problem of Apriori algorithm, FP-growth algorithm was introduced and it also called frequent pattern tree, or FP-tree [20]. FP-growth uses a combination of the vertical and horizontal database layout to store the database in main memory. Instead of storing the every item in the database, it stores the actual transactions from the database in a tree structure and every item has a linked list going through all transactions that contain that item [21].

Frequent pattern tree was denoted by data structure basically, all transactions are stored in a tree data structure. A frequent pattern tree is a tree structure which is defined in two steps .It consists of one root labeled as "root", (sub trees as the children of the root), and a frequent-item header. Each node in the item prefix sub-tree consists of three fields: item-name, count and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree. [22]

```
The FP-growth Algorithm

Input: constructed FP-tree
Output: complete set of frequent patterns
Method: Call FP-growth (FP-tree, null)
Procedure FP-growth (Tree, α)
{
    1) if Tree contains a single path P then
    2)  for each combination do generate pattern β
        α with support = minimum support of nodes in β.
    3) Else For each header ai in the header of Tree
    do {
    4) Generate pattern β = ai α with support = ai.support;
    5) 5) Construct β.s conditional pattern base and then β.s conditional FP-tree Tree β
    6) 6) If Tree β = null
    7) 7) Then call FP-growth (Tree β, β)
    8) }
```

E.FIN Algorithm

The framework of FIN algorithm consists of three steps: (1) Construct the POC-tree and identify all frequent 1-itemsets; (2) scan the POC-tree to find all frequent 2-itemsets and their Nodeset; (3) mine all the frequent itemset from the transactional database [21]. The FIN algorithm adopts the pruning strategy to enhance the efficiency of mining frequent itemsets, which is based on superset equivalence property. FIN constructs a set-enumeration tree to represent the search space [22]. First, the root of the tree is created. Secondly, the child nodes of the root are created, therefore. Thirdly, a node represents itemset and registering nodes are created. Finally, the set-enumeration tree is built by executing the third step repeatedly until all leaf nodes are created [23].

F. RElim Algorithm

The RElim (Recursive Elimination) algorithm [21] basically operates with horizontal transaction representation, but separates the transactions suffixes according to their leading item; it also introduces a vertical representation feature to mine preprocessed data from the transaction database. Rather than listing all transactions in one array, they are grouped according to their leading item. In addition the transactions are organized as lists, arrays would also be possible. These lists are sorted in descending frequency of their associated items in the transaction database: The first list is associated with the most frequent item, the last list with the least frequent item [24].

H. AprioriTID Algorithm

AprioriTID is an algorithm for discovering frequent itemsets (groups of items appearing frequently) in a transaction database. It was proposed by Agrawal & Srikant (1993). AprioriTID is a variation of the Apriori algorithm. It was proposed in the same article as Apriori as an alternative implementation of Apriori. It produces the same output as Apriori. But it uses a different mechanism for counting the support of itemsets [24].

The database is not used for counting the support of candidate itemsets after the first pass. The candidate itemsets are generated the same way as in Apriori algorithm. Another new set is generated for which each member has the TID of each transaction and the large itemsets present in this transaction. This set is used to count the support of each candidate itemset. The advantage is that the number of entries in a new set may be smaller than the number of transactions in the database, especially in the later passes [25].

I. H-Mine Algorithm

H-Mine algorithm uses a pattern-growth approach to discover frequent itemsets, it is also designed with H-struct for fast mining and it is efficient than Apriori and FP-Growth algorithm. It can easily predict memory space overhead and the main drawback of H-Mine algorithm is it can be used for datasets that can fit into main memory for mining frequent patterns. H-Mine is efficient only when the frequent-item of a transaction database and set of header tables can fit into the main memory. The database partitioning technique is developed when they cannot fit into the memory.

IV. RESULTS AND DISCUSSION

Different sizes of data sets, 1000, 5000 and 10000 transactions with 10 items are used. Performance metrics used are total execution time and memory usage and these are calculated for different dataset and minimum support values for all the algorithms. The execution time is decreased when the support threshold is increased. The AprioriTID and FIN algorithm takes more time compared to other algorithms. Table 1 gives the execution time required for the association rule algorithms.

TABLE I. EXECUTION TIME

Dataset	Min-Support	Apriori	Dclat	Eclat	Fp Growth	Fin	Apriori Tid	Relim	H-Mine
1000	10	31	16	16	31	47	78	31	31
	15	31	16	15	32	32	31	31	32
	20	31	0	16	32	46	31	31	31
	25	15	0	15	32	47	31	32	31
	30	16	15	16	31	47	31	16	15
5000	10	63	47	62	47	47	125	62	62
	15	47	31	47	47	47	94	62	47
	20	47	31	31	47	47	78	63	62
	25	47	31	31	47	47	78	62	62
	30	47	31	31	47	63	78	63	47
10000	10	79	63	78	63	63	125	78	78
	15	63	47	63	63	63	94	79	63
	20	47	47	47	47	63	78	78	79
	25	46	37	47	63	62	78	90	78
	30	47	47	47	63	47	78	78	62

Figure 2 shows the performances analysis of all the algorithms from this observation Dclat is better than other algorithms.

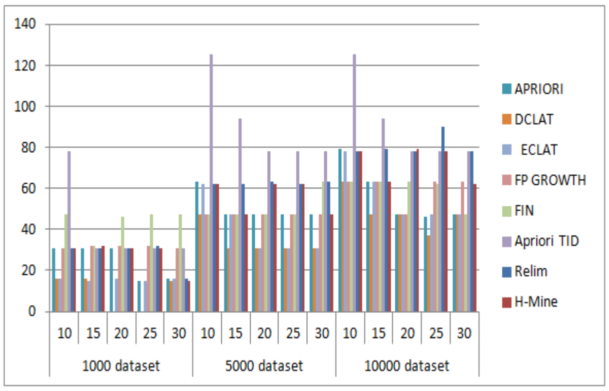

Fig: 2 Execution Time Analysis

Table 2 describes about the Maximum Memory Usage for association rule algorithms for different transactions sizes with different support values.

TABLE II. MAXIMUM MEMORY USAGES FOR DIFFERENT DATASET

Dataset	Min-Support	Apriori	Dclat	Eclat	Fp Growth	Fin	Apriori Tid	Relim	H-Mine
1000	10	1.41	2.11	5.78	5.39	40.18	3.92	2.55	2.58
	15	1.41	1.94	5.48	2.05	40.18	3.92	2.55	2.40
	20	1.32	1.86	4.60	2.05	40.10	3.92	2.37	2.23
	25	1.32	1.68	3.63	1.97	40.10	3.92	2.37	2.14
	30	1.32	1.68	3.36	1.97	40.10	3.92	2.37	2.06
5000	10	3.76	4.51	5.78	1.97	39.69	3.92	4.79	4.87
	15	3.67	3.79	5.48	2.89	39.69	3.92	4.80	4.71
	20	3.76	3.26	4.60	2.78	39.69	3.92	4.72	4.61
	25	3.67	2.75	3.63	2.78	39.69	3.92	4.74	4.04
	30	3.59	2.66	3.36	2.72	39.69	3.92	4.74	3.86
10000	10	3.18	7.45	7.52	4.27	40.24	4.47	5.70	5.72
	15	3.09	6.76	6.82	4.18	40.24	4.47	5.16	5.53
	20	2.91	5.65	6.98	4.18	40.24	4.47	5.18	5.20
	25	2.91	4.68	6.35	4.18	40.24	4.47	5.18	4.95
	30	2.74	4.18	5.39	4.18	40.24	4.47	5.18	5.21

Figure 3 represents the performances analysis of all the algorithms from this observation Apriori is best than all other algorithms.

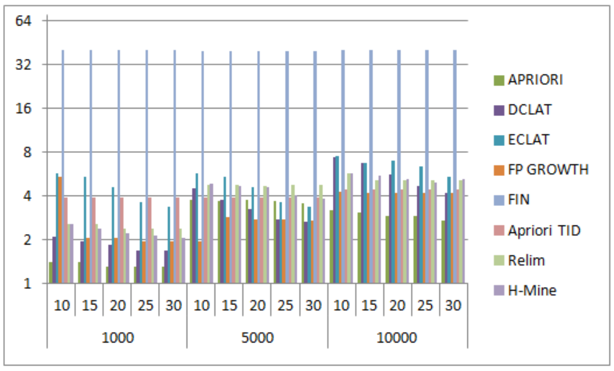

Fig 3 Memory Usages analysis

Table 3 gives the results about generation of frequent items, number of association rules for different transaction, different support and confidence values.

TABLE III.ASSOCIATION RULES GENERATION

Dataset	MIN.SUP	MIN.CON	FIC	NO OF RULES	TOTAL TIME
	10	30	135	701	0
	10	60	135	316	16
	20	40	48	114	0
	25	35	28	48	0
	30	80	22	4	0
1000	20	70	48	37	0
	10	30	135	857	16
	10	60	135	316	16
	20	40	48	114	0
	25	35	28	48	0
	30	80	22	4	16
5000	20	70	48	37	0
	10	30	63	584	16
	10	60	121	254	0
	20	40	42	83	15
	25	35	27	42	0
	30	80	20	4	0
10000	20	70	42	27	0

Figure 4 represents the number of rule generation and execution time for all the algorithms.

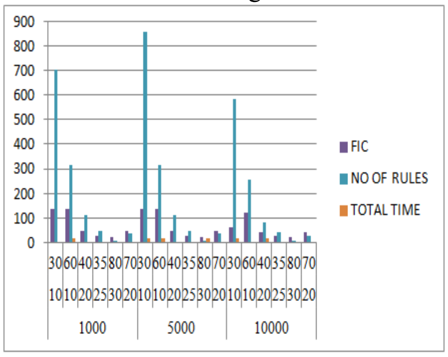

Fig: 4 Analysis Of Association Rule Generation

V. CONCLUSION AND FUTURE ENHANCEMENT

Data mining is an important research domain which focuses on knowledge discovery in databases. This research has analyzed the performances of eight association rule mining algorithms. Algorithm used for analysis are Apriori, Eclat, Dclat, FP-growth, FIN, AprioriTID, Relim and H-Mine .The factors used for the analysis are execution time ,memory usage and number of rules are generated for different dataset and minimum support values.

The execution time is decreased when the support threshold value is increased. The Eclat and Dclat algorithm have produced better results in execution time compared with other algorithms. When compared with memory consumption Apriori algorithm has given better result than other algorithms. The AprioriTID algorithm takes more execution time and FIN algorithm takes more memory usage compared to other algorithms. Finally Association rules are generated for all the eight algorithms with different minimum support and confidence. The comparative analysis has concluded that the Dclat algorithm is best comparing to other algorithms. In future, this work will be implemented with different types of dataset like medical, bioinformatics, CRM, telecommunication etc. Enhanced algorithm will be developed to obtain the best results.

REFERENCE

[1] NogaAlon, PhillipB.Gibbons, YossiMatias,and MarioSzegedy.(1999). Tracking Join and Self-Join Sizes in Limited Storage. In ACM PODS.

[2] S.Vijayarani, Dr. A.Tamilarasi,R. SeethLakshmi Tabu Search based Association Rule Hiding. International Journal of Computer Applications (0975 – 8887) Volume 19– No.1, April 2011.

[3] Bhoj Raj Sharmaa*, Daljeet Kaura and Manjub, Baru Sahib, Sirmour, A Review on Data Mining: Its Challenges, Issues and Applications, International Journal of Current Engineering and Technology ISSN 2277 - 4106 © 2013 INPRESSCO. http://inpressco.com/category/ijcet) Accepted 20 June 2013, Available online 25June 2013, Vol.3, No.2 (June 2013)

[4] Http://www.aspfree.com/c/a/MS-SQL-Server/Using-Data-Mining-for-Business-Intelligence/

[5] Https://www.scribd.com/doc/38677433/How-Evolution-of-Database-Led-to-Data-Mining

[6] Shuxiang Xu ., Launceston, Tas., Australia; Ming ZhangData mining - an adaptive neural network model for financial analysis. 2

[7] https://www.linkedin.com/pulse/what-heck-association-rules-analytics-jeffrey-strickland-ph-d-cmsp

[8] V.Moustakides a, Vassilios S. Verykios b,* Rio, Greece, A MaxMin approach for hiding frequent itemsets Data & Knowledge Engineerin65 (2008) 75–89 George e Received 5 June 2007; accepted 5 June 2007 Available online 18 July 2007

[9] P.Nithya and G Lakshmipriya, An Overview of data mining and Data warehousing – Architecture, Techniques and applications] , International journal of computer science tends and technology (ijcst).volume 3 issue 1, jan-feb 2015

[10] m.ravikanth1 , g.loshma, parallel multithreaded Apriori algorithm for vertical association rule mining International journal of advanced research in computer and communication engineering vol. 2, issue 12, december 2013 4729

[11] Smita R.Sankhe,Kavita Kelkar, Optimization of Execution Time using Association Rule Mining Algorithms International Journal of Computer Applications (0975 – 8887) Volume 59– No.11, December 2012 18

[12] Darshan.J. , Improved Apriori Algorithm for Mining Association Rules Information Technology and Computer Science, 2014, 07, 15-23 Published Online June 2014 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2014.07.03 Copyright © 2014 MECS I.J. Information Technology and Computer Science, 2014, 07, 15-23

[13] Mining Manasa G* , Mrs. Kulkarni Varsha, Integration of Apriori and FP-Growth Techniques to Personalize Data in Web , International Journal of Scientific and Research Publications, Volume 5, Issue 7, July 2015 1 ISSN 2250-3153 www.ijsrp.org IAFP:

[14] Xingzhi , Beijing, Philip S. Yu Hiding Sensitive Frequent Itemsets by a Border-Based Approach

[15] HEIKKI MANNILA, HANNU TOIVONEN Data Mining and Knowledge Discovery 1, 241–258 (1997) c °1997 Kluwer Academic Publishers. Manufactured in The Netherlands. Level wise Search and Borders of Theories in Knowledge Discovery

[16] Patel Tushar S.1, Panchal Mayur2, Ladumor Dhara2, Kapadiya Jahnvi2, Desai Piyusha2, Prajapati Ashish3 and Prajapati Reecha4, Association An Analytical Study of Various Frequent Itemset Mining Algorithms, Research Journal of Computer and Information Technology Sciences Vol. 1(1), 6-9, February (2013) Res. J. Computer & IT Sci. International Science Congress

[17] Anitha Modi1 , Radhika Krishnan, An Improved Method for Frequent Itemset Mining , International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 5, May 2013) 143

[18] Amit Mittal1, Ashutosh Nagar2, Kartik Gupta3, Rishi Nahar , Comparative Study of Various Frequent Pattern Mining Algorithms International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 4, April 2015 Copyright to IJARCC DOI 10.17148/IJARCCE.2015.44127 550

[19] .J.R.Jeba,Kumaracoil Dr.S.P.Victor, Comparison of Frequent Item Set Mining Algorithms, International Journal of Computer Science and Information Technologies, Vol. 2 (6) , 2011, 2838-2841

[20] Pramod S. O.P. Vyas Survey on Frequent Item set Mining Algorithms International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 15 86

[21] Zhi-Hong Deng , Sheng-Long Lv, Fast mining frequent itemsets using Node sets, Expert Systems with Applications 41 (2014) 4505–4512

[22] jian pei1,∗, jiawei han2, hongjun lu3, shojiro nishio4, shiwei tang5 and dongqing yang .h-mine: fast and space-preserving frequent pattern mining large databases

[23] J. Han, H. Pei, and Y.Yin. Mining Frequent Patterns without Candidate Generation. In: Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX). ACM Press, New York, NY, USA 2000.

[24] P. Prithiviraj, R. Porkod A Comparative Analysis of Association Rule Mining Algorithms in Data Mining: A Study *American journal of computer science and engineering survey*

[25] Charanjeet Kaur, Association Rule Mining using Apriori Algorithm: A Survey International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 6, June 2013 ISSN: 2278 – 1323

[26] Mohammed J. Zaki† and Karam Gouda Fast Vertical Mining Using Diffsets 812-8581 Japan kgouda@csce.kyushu-u.ac.jp

[27] Xiaomei Yu, Hong Wang, Shandong, Shandong. Improvement of Eclat Algorithm Based on Support in Frequent Itemset Mining