**International Journal of Pure and Applied Mathematics**
**Volume 118 No. 24 2018**
ISSN: 1314-3395 (on-line version)
**url:** http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

# A STUDY OF AN ENHANCED APPROACH TOWARDS FREQUENT PATTERN MINING

K. MOUNIKA[1], V. CHANDRA SHEKHAR RAO[2],
S. KIRAN[3]
[1]Student,[2,3]Assistant Professor,
Department of CSE,
KITS,
Warangal, India.

May 27, 2018

**Abstract**

Association rule mining is one of the imperative errands in data mining. The undertaking to locate the frequent patterns is assuming a fundamental part in mining associations and numerous other intriguing highlights among the factors in the transactional database. In any case, this assignment is computationally escalated and utilizes a significant extensive measure of memory. There are numerous components that include the working of a frequent pattern mining algorithm. One of the variables that have a noteworthy impact is the attributes of the database being examined. The well known algorithm works distinctively on inadequate and thick database. Two algorithms are being connected to the database as indicated by the data attributes of the dataset. FEM(FP-Tree and Eclat Method) utilizes a settled edge as an exchanging condition between the two mining techniques while DFEM(Dynamic FP-Tree and Eclat Method) applies an edge dynamically at runtime to efficiently fit the qualities of the database amid the mining procedure. The execution

of these algorithms is likewise contrasted and other efficient algorithms.

***Key Words*:** Frequent Patterns, Data Mining, Frequent Pattern Mining Algorithm, Association Rule Mining.

# 1 INTRODUCTION

One of the essential undertakings in data mining is association rule mining, which centers around discovering rules that determine the event of the things in the subsets in databases. [1] Frequent pattern mining is an imperative undertaking, used to locate the distinctive sorts of relationship among factors in expansive database. Its fundamental concentration is to look for itemsets, sub successions that co-happen with a base recurrence more noteworthy than the client determined help check. Apriori, FP-development, Eclat are a portion of the generally utilized algorithms. Apriori algorithm utilizes the property that an itemset happen frequently if and just if the majority of its sub-things are frequent. It uses a level-wise or expansiveness first approach of the itemset look space which essentially prunes all the superset. It likewise maintains a strategic distance from the age of any candidate age that has any infrequent subset.FP-development files the database for quick calculation of the help tally by means of the utilization of a data structure called the frequent pattern tree or the FP-tree. Tallying the help check can be enhanced essentially if the database is recorded such that it permits quick recurrence calculations. In level-wise approach, to figure the help tally, it is expected to create subsets of every exchange and check on the off chance that they exist in the prefix tree. Not at all like the level-wise approach Eclat algorithm utilizes the vertical TID rundown to locate the frequent itemset by crossing these TID rundown and then registering their resultant help check [5]. From different trials performed on various databases it is demonstrated that the ARM methods functioned admirably for a curious sort of databases [5], [2], [6], [7]and [4]. The methods either worked for scanty or thick database and inadequately on both. In this paper we talk about two effective algorithms , DFEM (Dynamic FEM) and FEM (FP-development and Eclat Mining) which is a blend of FP-development and Eclat algorithm [8] and [9]. It utilizes FP-tree to store the database minimalistically. The principle highlight of

these algorithms is that they dynamically switch between the two algorithms by thinking about the data qualities. In the event that the restrictive base is littler than it utilizes the TID-list for mining, else it utilizes the FP-development. The exchanging choices are made in light of the edge k whose esteem is dynamic and assessed amid runtime.

# 2 BACKGROUND

In this segment, we talk about the issue proclamation, survey FP-development and Eclat mining and break down their diverse perspectives like quality and shortcomings.

*A. Problem Statement*

The mining issue can be expressed as takes after: Let I = i1,i2,.....in be the arrangement of every single one of a kind thing present in the database D. The help include of an item set X database D is the quantity of exchanges in D that contain X. An item set k having bolster tally is said to be frequent if is more prominent than or equivalent to the client indicated least help check. The certainty of a rule is the restrictive likelihood that an exchange contains Y given that it contains X. Given a database and a client indicated least help check, the undertaking is to discover sets of all frequent thing sets in the database D. For instance given a dataset in Table 1 and least help check = 25%, the 1-frequent item set incorporates a,b,c,d,e where as f is infrequent as the help tally is 11%. Also the 2-frequent item set and 3-frequent item sets are figured.

TABLE I DATASET WITH MINIMUM SUPPORT COUNT = 25%

| TID | Items | Sorted Items |
|---|---|---|
| 1 | b,d,a | a,b,d |
| 2 | c,d,b | b,c,d |
| 3 | c,d,a,e | a,c,d,e |
| 4 | d,a,e | a,d,e |
| 5 | c,d,a | a,c,d |
| 6 | c,a,d | a,c,d |
| 7 | f | |
| 8 | b,d,a | a,b,d |
| 9 | c,a,b,e | a,b,c,e |

*B. FP-growth algorithm*

It is an extremely efficient algorithm proposed by Han et al. [2] for identifying frequent patterns. It uses FP-tree (Frequent Pattern tree) to find the patterns that happen frequently. This tree is a propelled prefix-tree structure which considers the database on a level plane and efficiently packs and stores data in the memory. The tree comprises of a root hub which is typically kept as invalid, and leaf hubs that comprises of the things and a header table that comprise of the frequent thing and its help check. It at that point mines the tree recursively to locate the frequent patterns without producing an extensive number of candidates. This when contrasted with Apriori , it performs better as far as productivity. At the point when the database is thick or the base help check is set to low esteem the quantity of frequent patterns produced is huge. Subsequently, the cost for creating an extensive number of frequent patterns brings about execution debasement. In such cases, it's smarter to utilize Eclat [4] and [2].

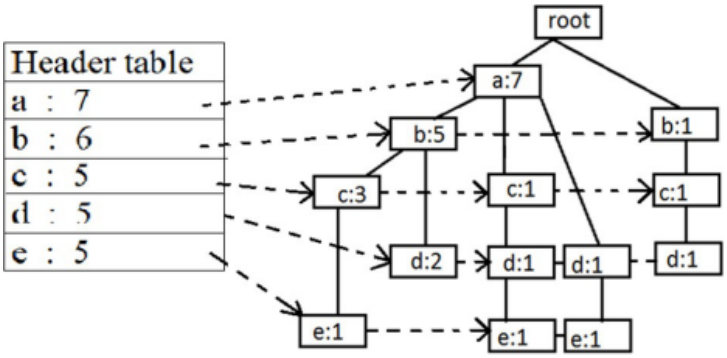


Figure 1 FP-tree which is structured from dataset in Table 1

*C. Eclat algorithm*

Eclat mining algorithm is a notified algorithm for mining frequent patterns created by Zaki et al. [3]. It influences utilization of a data to structure called TID-list (exchange id list). It contains the IDs of the exchange of a specific thing or an itemset. This rundown is utilized to speak to the database in vertical arrangement. Along these lines, it sends profundity first pursuit system.

This algorithm examines the database twice. In the principal check all the frequent things are been found and in the second output, it produces the TID-rundown of frequent things. This algorithm will then sort out the itemsets into disjoint comparability classes as indicated by the prefixes. By making the TID-show it is less demanding to discover the help of the candidate itemset as it can be registered by basically meeting the TID-rundown of the two part subset. The resultant TID-rundown can be checked effortlessly to discover whether it is frequently happening or not.

| TID | Frequent items |
|---|---|
| 1 | a,b,d |
| 2 | b,c,d |
| 3 | a,c,d,e |
| 4 | a,d,e |
| 5 | a,b,c |
| 6 | a,b,c |
| 7 | |
| 8 | a,b,d |
| 9 | a,b,c,e |

→

| a | b | c | d | e |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Figure 2 Bit vector generated from the dataset in Table 1

*D. Mining approach for Frequent Pattern Mining*

The databases comprise of a gathering of things happening frequently that similar things. This will happen more in thick database and less in meager. Expulsion of the less frequent thing from the database will give the database the thick qualities and the expelled divide inadequate attributes. In the FP-tree the root hub contains the most frequent thing and it is at the best most level, while whatever is left of the hubs are included into the tree as the leaf hubs. At long last the tree will contain every one of the things in the dropping request of the recurrence of event.

In this paper, we examine the algorithm that is reasonable for mining databases thinking about the data attributes. The algorithm will adjust to the scanty and thick attributes of the data set under thought. From past examinations unmistakably inadequate data can be best handled by FP-development mining and the thick data can be handled by Eclat algorithm. The base limit

that chooses whether to build FP-tree or to develop Bit vector is ascertained dynamically.

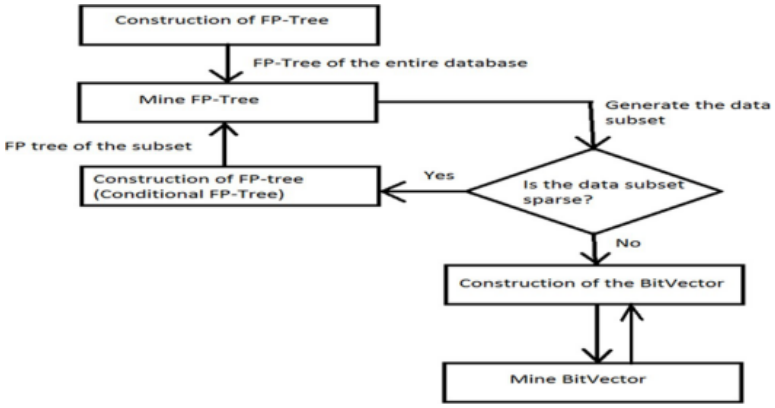


Figure 3 Mining model for frequent pattern mining

*E. Overview of thealgorithm*

DFEM joins FP-development and Eclat algorithm systems for mining. FP-tree is utilized to store the database in the memory in a minimized way. Amid the mining procedure this tree is being utilized recursively to locate the frequent pattern. The exchanging between the FP-development and Eclat algorithm happens in view of the edge being characterized. The algorithm comprises of four noteworthy parts:

1. Development of FP-tree: Database is checked to discover all the frequent things and the header table is made. The database is examined again to get the frequent things to such an extent that the FP - tree can be built.
   Construction of FP-tree :
   Input: Database and the min-support
   Output : Complete set of frequent patterns
   Step 1: Scan the database and find the frequent items.
   Step 2: Scan the database to construct FP-tree
   Step 3: Call FP-tree mining.

2. Mining FP-tree: It utilizes the FP-development algorithm to discover all the frequent patterns from the contingent tree built recursively. Prior to the development of the restrictive tree the size is to be checked. On the off chance that the size is little at that point Bit Vector will be produced, generally the FP - tree will be made.
   FP-tree mining
   Input : Conditional FP-tree ,min-support , suffix
   Output: Set of frequent patterns
   Step 1: If the FP-tree consists only single path P
   Step 2: Then for each combinations of x of the nodes in P
   Step 3: Output = x U suffix
   Step 4: else for each item y in the header table of FP-tree
   Step 5: Output= y U suffix
   Step 6: Construct y conditional pattern base C
   Step 7: size = number of nodes in the y
   Step 8: if size¿k
   Step 9: then construct ys conditional FP-tree and call FP-tree miningagain
   Step 10: else transform C into bit vector V and weight vector W and call Bit Vector mining.

3. Mining Bit Vector: It will gather all the TID bit vector from the database and looks for frequent pattern by sensibly adding these bit vectors recursively. The new patterns made by linking the addition pattern from the past advances.
   Bit Vector mining
   Input : Bit vector V, weight vector W, suffix, min-support
   Output: Set of frequent patterns.
   Step 1: Sort V in descending order of its item support.
   Step 2: For each vector vi in V
   Step 3: Output= vi U suffix
   Step 4: For each vector vk in V, k¡i
   Step 5: uk=vi AND vk
   Step 6: supk= support of uk based on w
   Step 7: If all uk in U are identical to vi
   Step 8: Then for each combination x in U output1 = x U output

Step 9: else if U is empty call Bit vector mining again

4. Updating the threshold: Let k be the threshold being defined to find the frequent patterns being generated by FP -tree mining where k=k0,k1...kn be the set of all values of k being applied. Ri is the ratio indicating the difference between the previous pattern Pi-1 and the current pattern Pi and is computed as Ri=Pi-1/Pi,(i=1...N). The best k will satisfy the condition

$$Ri < 2\exists(Rj > 2, \forall j >)[8][9]. \quad (1)$$

Update threshold

Input : New Pattern and Size
Output : updated value of threshold K
Step 1: If Update K is called for the first time then
Step 2: Create an array P with N elements
Step 3: Initialize the array to zero
Step 4: For i=0 to N-1
Step 5: If Size ¿i*size then Pi=Pi+New Pattern
Step 6: else exit loop
Step 7: K=0
Step 8: For i=N-1 to 1
Step 9: If Ri¿=2 then K=(i+1)*Step and exit loop.

# 3 EXPERIMENTALRESULTS

The algorithm is being benchmarked with mainstream algorithm, for example, Apriori, FP-development, and Eclat. The execution correlation of the algorithm demonstrates that the algorithm is more steady and performs better that the prevalent algorithms. The dataset which is utilized for looking at the execution are mischance dataset which is a direct kind of dataset and retail dataset which is an inadequate dataset. Apriori runs the slowest when contrasted with whatever is left of the algorithms on both the datasets. Eclat performs better for the mishap dataset and runs slower for retail dataset, where as FP-development works genuinely better for retail when contrasted with Eclat. Be that as it may, the algorithm

examined in this paper works better when contrasted with every one of the algorithms.

# 4 CONCLUSION

In this paper, we examined about DFEM algorithm, the dynamic manner by which it figures the limit as indicated by the qualities of the dataset and then adjusts. It joins highlights of the two noteworthy algorithm that is the FP-development algorithm and Eclat algorithm. The exchanging between the algorithms is being chosen in view of the edge esteem. The future work incorporates enhancing the proficiency of Eclat algorithm by contracting the span of the middle TID sets.

# References

[1] R. Srikant R. Agrawal. Fast algorithms for mining association rules. Int. Conf. on Very large databases, 1994.

[2] J. Pei, Y.Yin J. Han. Mining frequent patterns without candidate generation. Int. Conf. on Mgt. of Data., 2000.

[3] S. Parthasarathy, M. Ogihara,W.Li M. Zaki. New algorithms for fast discovery of association rules. Knowledge Discovery and Data Mining , 1997.

[4] J.Zhu G. Grahne. Efficiently using prefix-trees in mining frequent itemsets. Workshop on frequent pattern mining implementations, 2003.

[5] B. Racz. Nonordfp: An FP-growth variation without rebuilding the FP-tree. IEEE ICDM workshop pn frequent itemset mining implementations, 2004.

[6] J.Pei et al. Hmine:Hyper-structure mining of frequent patterns in large databases. IEEE Int. conf. on data mining, 2001.

[7] G. Alaghband L. Vu. A fast algorithm combining FP-tree and TID-list for frequent pattern mining. Int. Conf. on Inf. and Knowledge Engineering, 2011.

[8] G. Alaghband L. Vu. Mining frequent patterns based on data characteristics. Int. Conf. on Information and Knowledge Engineering, 2012.

[9] Gita Alaghband Lan Vu. An Efficient Approach for Mining Association Rules from Sparse and Dense Databases. IEEE, 2014.

[10] Frequent Itemset Mining Implementations Repository. Workshop on Frequent Itemset Mining Implementation,(2003-2004), Available at http://fimi.ua.ac.be