iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** Issue I **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Dynamic Approach for Frequent Pattern Mining Using Database Characteristics

Iona Sudheendran[1], Ganesh Kumar R[2]
[1,2]Department of Computer Science and Engineering, Christ University, Bangalore, India

Abstract— Association rule mining is one of the important tasks in data mining. The task to find the frequent patterns is playing an essential role in mining associations and many other interesting features among the variables in the transactional database. But this task is computationally intensive and uses quite a large amount of memory. There are many elements that involve the functioning of a frequent pattern mining algorithm. One of the factors that have a significant effect is the characteristics of the database being analysed. The popular algorithm works differently on sparse and dense database. Two algorithms are being applied to the database according to the data characteristics of the dataset. FEM(FP-Tree and Eclat Method) employs a fixed threshold as a switching condition between the two mining strategies whereas DFEM(Dynamic FP-Tree and Eclat Method) applies a threshold dynamically at runtime to efficiently fit the characteristics of the database during the mining process. The performance of these algorithms is also compared with other efficient algorithms.
Keywords— Data Mining, Association Rule Mining, Frequent Patterns, Frequent Pattern Mining Algorithm.

I. INTRODUCTION

One of the important tasks in data mining is association rule mining, which focuses on finding rules that specify the occurrence of the items in the subsets in databases. [1] Frequent pattern mining is an important task, used to find the different types of relationship among variables in large database. Its main focus is to search for itemsets, sub sequences that co-occur with a minimum frequency greater than the user-specified support count. Apriori, FP-growth, Eclat are some of the widely used algorithms. Apriori algorithm uses the property that an itemset occur frequently if and only if all of its sub-items are frequent. It utilizes a level-wise or breadth-first approach of the itemset search space which significantly prunes all the superset. It also avoids the generation of any candidate generation that has any infrequent subset.FP-growth indexes the database for fast computation of the support count via the use of a data structure called the frequent pattern tree or the FP-tree, it then recursively mines these trees to find the frequent itemset [2]. Counting the support count can be improved significantly if the database is indexed in such a way that it allows fast frequency computations. In level-wise approach, to compute the support count, it is needed to generate subsets of each transaction and check if they exist in the prefix tree. Unlike the level-wise approach Eclat algorithm uses the vertical TID list to find the frequent itemset by intersecting these TID list and then computing their resultant support count [5]. From various experiments performed on different databases it is proven that the ARM methods worked well for a peculiar type of databases [5], [2], [6], [7]and [4]. The methods either worked for sparse or dense database and poorly on both. In this paper we discuss two efficient algorithms , FEM (FP-growth &Eclat Mining) and DFEM (Dynamic FEM) which is a combination of FP-growth and Eclat algorithm [8] and [9]. It uses FP-tree to store the database compactly. The main feature of these algorithms is that they dynamically switch between the two algorithms by considering the data characteristics. If the conditional base is smaller than it uses the TID-list for mining, else it uses the FP-growth. The switching decisions are made based on the threshold k whose value is dynamic and evaluated during runtime.

II. BACKGROUND

In this section, we discuss the problem statement, review FP-growth and Eclat mining and analyse their different aspects like strength and weaknesses.

A. Problem Statement

The mining problem can be stated as follows: Let $I = \{ i_1, i_2, \ldots . i_n \}$ be the set of all unique items present in the database D. The *support* count of an itemset X in database D is the number of transactions in D that contain X. An itemset k having support count α is said to be frequent if α is greater than or equal to the user specified minimum support count. The *confidence* of a rule is the conditional probability that a transaction contains *Y* given that it contains *X*. Given a database and a user specified minimum support count, the task is to find sets of all frequent item sets in the database D. For example given a dataset in Table 1 and minimum support count = 25%, the 1-frequent itemset includes a,b,c,d,e where as f is infrequent as the support count is 11%. Similarly the 2-frequent itemset and 3-frequent itemsets are computed.

TABLE 1: 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 a very efficient algorithm proposed by Han et al. [2] for finding frequent patterns. It uses the data structure called FP-tree (Frequent Pattern tree) to discover the patterns that occur frequently. This tree is an advanced prefix-tree structure which considers the database horizontally and efficiently compresses and stores data in the memory. The tree consists of a root node which is usually kept as null, and leaf nodes that consists of the items and a header table that consist of the frequent item and its support count. It then mines the tree recursively to find the frequent patterns without generating a large number of candidates. This when compared to Apriori , it performs better in terms of efficiency. When the database is dense or the minimum support count is set to very low value the number of frequent patterns generated is very large. Thus, the cost for generating a large number of frequent patterns results in performance degradation. In such cases, it's better to use Eclat [4] and [2].

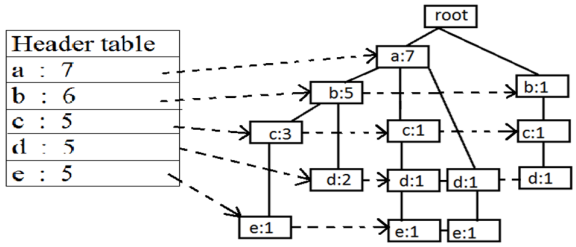


Figure 1. FP-tree constructed from the dataset in Table 1

C. Eclat algorithm

Eclat mining algorithm is also a well-known algorithm for mining frequent patterns developed by Zaki et al. [3]. It makes use of a data structure called TID-list (transaction id list). It contains the IDs of the transaction of a particular item or an itemset. This list is used to represent the database in vertical format. Therefore, it deploys depth-first search strategy. This algorithm scans the database twice. In the first scan all the frequent items are been discovered and in the second scan, it generates the TID-list of frequent items. This algorithm will then organize the itemsets into disjoint equivalence classes according to the prefixes. By creating the TID-list it is easier to find the support of the candidate itemset as it can be computed by simply intersecting the TID-list of the two component subset. The resultant TID-list can be checked easily to find whether it is frequently occurring 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 consist of a group of items occurring very frequently that the same items. This will occur more in dense database and less in sparse. Removal of the less frequent item from the database will give the database the dense characteristics and the removed portion sparse characteristics. In the FP-tree the root node contains the most frequent item and it is at the top most level, whereas the rest of the nodes are added into the tree as the leaf nodes. Finally the tree will contain all the items in the descending order of the frequency of occurrence.

In this paper, we discuss the algorithm that is suitable for mining databases considering the data characteristics. The algorithm will adapt itself to the sparse and dense characteristics of the data set under consideration. From previous studies it is clear that sparse data can be best handled by FP-growth mining and the dense data can be handled by Eclat algorithm. The minimum threshold that decides whether to construct FP-tree or to construct Bit vector is calculated dynamically.

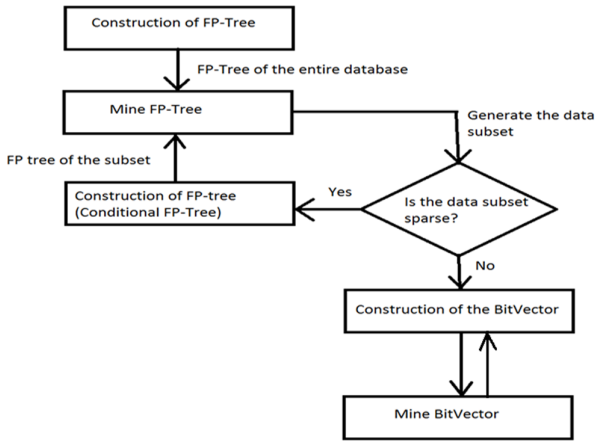


Figure 3. Mining model for frequent pattern mining

E. Overview of the algorithm

DFEM combines FP-growth and Eclat algorithm strategies for mining. FP-tree is used to store the database in the memory in a compact manner. During the mining process this tree is being used recursively to find the frequent pattern. The switching between the FP-growth and Eclat algorithm happens based on the threshold being defined. The algorithm consists of four major parts:

1) Construction of FP-tree: Database is scanned to find all the frequent items and the header table is created. The database is scanned once more to get the frequent items such that the FP - tree can be constructed.

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 uses the FP-growth algorithm to find all the frequent patterns from the conditional tree constructed recursively. Before the construction of the conditional tree the size is to be verified. If the size is small then Bit Vector will be generated, otherwise the FP - tree will be created.

Volume 3, Special Issue-1, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

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 y's conditional FP-tree and call FP-tree mining again Step 10: else transform C into bit vector V and weight vector W and call Bit Vector mining.

3) Mining Bit Vector: It will collect all the TID bit vector from the database and searches for frequent pattern by logically ANDing these bit vectors recursively. The new patterns created by concatenating the suffix pattern from the previous steps.

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 v_i in V Step 3: Output= v_i U suffix Step 4: For each vector v_k in V, k<i Step 5: u_k=v_i AND v_k Step6: sup_k= support of u_k based on w Step7: If all u_k in U are identical to v_i 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 $k=\{k_0,k_1...k_n\}$ be the set of all values of k being applied. R_i is the ratio indicating the difference between the previous pattern P_{i-1} and the current pattern P_i and is computed as $R_i = P_{i-1}/Pi,(i=1...N)$. The best k will satisfy the condition $R_i<2 \ni(\nexists R_j> 2,\forall j>)$ [8] [9].

Update threshold
Input : NewPattern and Size Output : updated value of threshold K Step 1: If UpdateK 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 P_i=P_i+NewPattern Step 6: else exit loop Step 7: K=0 Step 8: For i=N-1 to 1 Step 9: If R_i>=2 then K=(i+1)*Step and exit loop.

III. EXPERIMENTAL RESULTS

The algorithm is being benchmarked with popular algorithm such as Apriori, FP-growth, and Eclat. The performance comparison of the algorithm shows that the algorithm is more stable and performs better that the popular algorithms. The dataset which is used for comparing the performance are accident dataset which is a moderate type of dataset and retail dataset which is a sparse dataset. Apriori runs the slowest when compared to the rest of the algorithms on both the datasets. Eclat performs better for the accident dataset and runs slower for retail dataset, where as FP-growth works fairly better for retail when compared to Eclat. But the algorithm discussed in this paper works better when compared to all the algorithms.

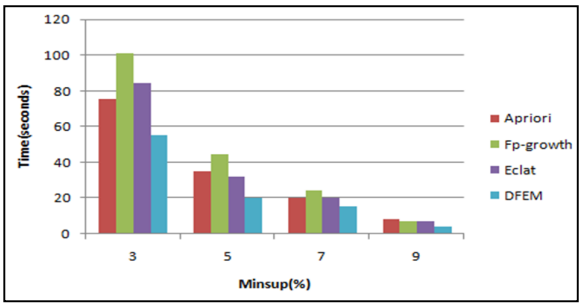


Figure 4. Execution time of DFEM and other algorithms with accident(moderate) dataset.

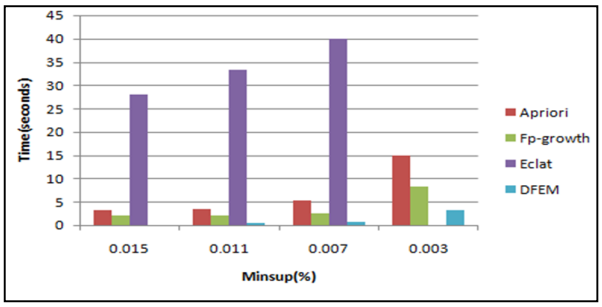


Figure 4. Execution time of DFEM and other algorithms with retail(sparse) dataset.

IV. CONCLUSION

In this paper, we discussed about DFEM algorithm, the dynamic way in which it computes the threshold according to the characteristics of the dataset and then adapts itself. It combines features of the two major algorithm that is the FP-growth algorithm and Eclat algorithm. The switching between the algorithms is being decided based upon the threshold value. The future work includes improving the efficiency of Eclat algorithm by shrinking the size of the intermediate TID sets.

REFERENCES

[1] R. Srikant R. Agrawal, "Fast algorithms for mining association rules," in *Proc. of the Int. Conf. on Very large databases*, 1994, pp. 487-499.

[2] J. Pei, Y.Yin J. Han, "Mining frequent patterns without candidate generation," in *In Proc. of the 2000 ACM Sigmod Int. Conf. on Mgt. of Data. vol 9. issue 2*, 2000, pp. 1-12.

[3] S. Parthasarathy, M. Ogihara,W.Li M. Zaki, "New algorithms for fast discovery of association rules," in *In Proc. of Knowledge Discovery and Data Mining* , 1997, pp. 283-286.

[4] J.Zhu G. Grahne, "Efficiently using prefix-trees in mining frequent itemsets," in *In Proc. of the 2003 Workshop on frequent pattern mining implementations*, 2003, pp. 132-132.

[5] B. Racz, "Nonordfp: An FP-growth variation without rebuilding the FP-tree," in *In Proc. of The IEEE ICDM workshop pn frequent itemset mining implementations*, 2004.

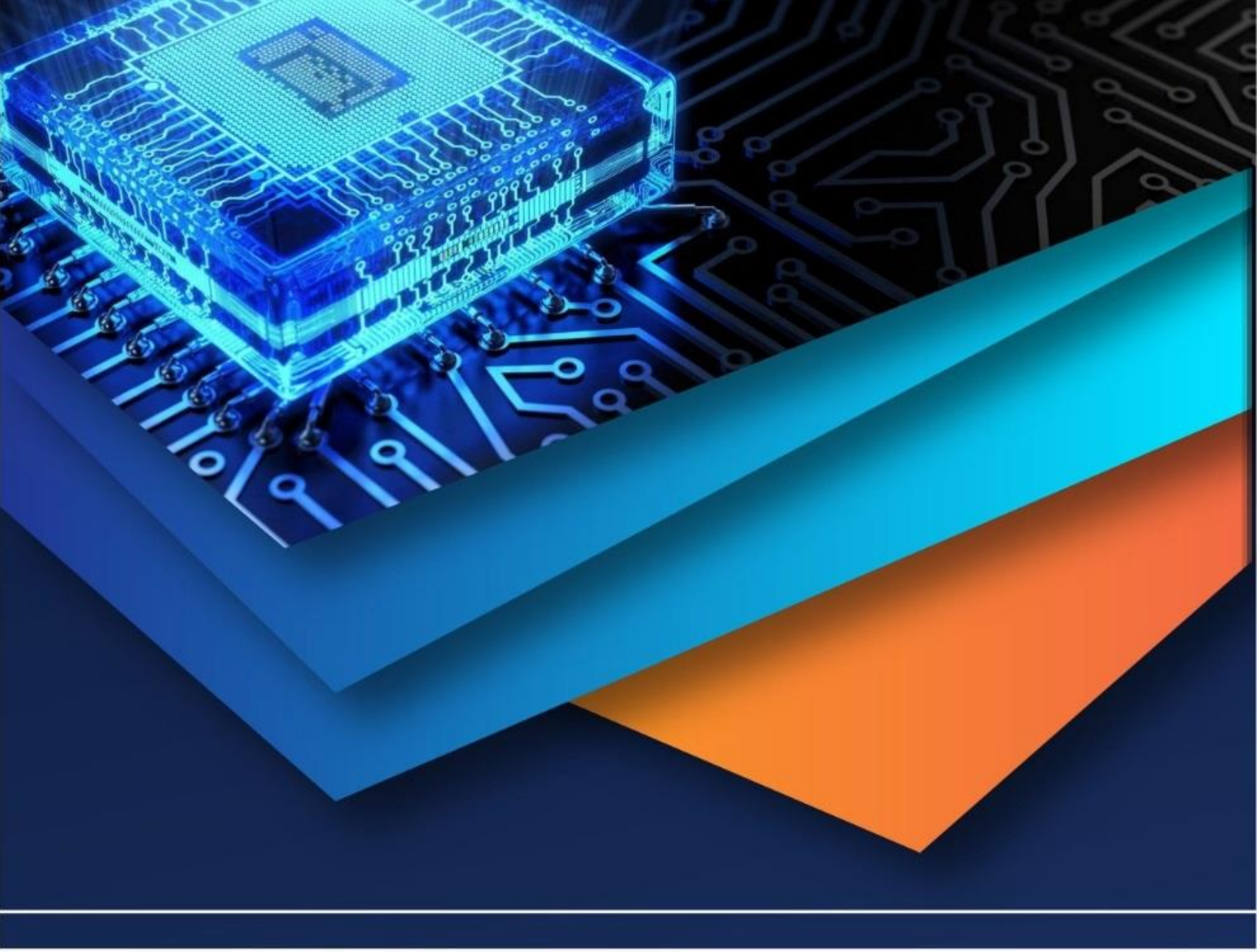
[6] J.Pei et al, "Hmine:Hyper-structure mining of frequent patterns in large databases," in *In Proc. of the IEEE Int. conf. on data mining*, 2001, pp. 441-448.

[7] G. Alaghband L. Vu, "A fast algorithm combining FP-tree and TID-list for frequent pattern mining," in *In Proc. of the 2011 Int. Conf. on Inf. and Knowledge Engineering*, 2011, pp. 472-477.

[8] G. Alaghband L. Vu, "Mining frequent patterns based on data characteristics," in *Int. Conf. on Information and Knowledge Engineering*, 2012, pp. 369-375.

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

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



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY