# Efficiently Mining Frequent Itemsets using Various Approaches: A Survey

C. A. Dhote, PhD.
System Manager
PRMITR
Badnera, India

Sheetal Rathi
PhD Research Scholar
SGBAU
Amravati, India

## ABSTRACT

In this paper we present the various elementary traversal approaches for mining association rules. We start with a formal definition of association rule and its basic algorithm. We then discuss the association rule mining algorithms from several perspectives such as breadth first approach, depth first approach and Hybrid approach. Comparison of the various approaches is done in terms of time complexity and I/O overhead on CPU. Finally, this paper prospects the association rule mining and discuss the areas where there is scope for scalability.

## General Terms

Data Mining, Association Rule Mining.

## Keywords

Frequent itemset mining; breadth first; depth first; hybrid approach.

## 1. INTRODUCTION

Data Mining and Knowledge Discovery in Databases (KDD) is an interdisciplinary field merging ideas from statistics, machine learning, databases, and parallel computing. Data mining refers to the overall process of discovering new patterns or building models from a given dataset. There are many techniques for mining a dataset such as: Rule discovery, classification and regression, sequential patterns and clustering. Out of all these, discovery of association rules is an important data mining problem.

Business organizations are depending on sophisticated decision-making information to maintain their competitiveness in today's demanding and fast changing marketplace. Inferring valuable high-level information based on large volumes of routine business data is becoming critical for making sound business decisions. One of the most difficult problems in database mining is the large volume of data that needs to be handled in a medium sized business; it is not uncommon to collect hundreds of megabytes to a few gigabytes of data [1]. Database mining applications often perform long-running, complex data analysis over the entire database. Given the large database sizes, one of the main challenges in database mining is developing fast and efficient algorithms that can handle large volumes of data. Mining for association rules involves extracting patterns from large databases and inferring useful rules from them. The prototypical application of association rules is the analysis of sales or market basket data [2]. Basket data consists of items bought by a customer along with the transaction identifier. The most typical example of association rules cases is: "80 percent of customers buy beers also buy diapers at the same time", its intuitive meaning is, how larger the tendency of customers buy certain products while they will buy other goods. Association rule mining is to help find the relationship between the itemset in a large number of databases. By describing the potential rules between the items in databases, dependencies between multiple domains which meet the given support and the confidence threshold are found. Association rules have been shown to be useful in domains that range from decision support to telecommunications alarm diagnosis, and prediction.

The structure of the rest of the paper is as follows: In section II we describe the basic association rule mining. In section III, we explore the various scope areas on which we can focus to enhance the speed of existing association rule algorithms .Section IV gives a description of some algorithms which are being used for mining association rules. Section V contains conclusion and pointers for further work.

## 2. BASICS OF ASSOCIATION RULES

Association rule mining has attracted tremendous attention from data mining researchers and as a result several algorithms have been proposed for it. Before going for the various approaches towards association rule mining, the basic concepts of association rule mining are introduced first.

The problem of mining associations over basket data was introduced in [2].The problem can be formulated as follows: Let I = {i1 ,i2... im} be a set of literals, called items. Let database D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with a unique identifier, called TID. Let X is a set of items. A transaction T is said to contain X if and only if $X \subseteq T$. The support of a set of items X given as σ(X) is the number or the percentage of transactions in the database that contain X. The confidence of the rule is the conditional probability that a transaction contains Y, given that it contains X and is given as σ (X U Y) / σ (X). A rule is frequent if its support is greater than min_sup and strong if its confidence is more than a user-specified minimum confidence (min_conf).

Thus the association rule mining is a two-step process:

1) Find all frequent itemset having minimum support.

2) Generate strong rules having minimum confidence, from the frequent itemset.

The first step is relatively more time consuming. It is candidate large itemset generation process and frequent itemset generation process. We call those itemset whose support exceed the support threshold as large or frequent itemset and those itemset that are expected or have the hope to be large or frequent are called candidate itemset. The second step is relatively easy, and the present focus in research is to find highly efficient algorithm in the first step i.e. to effectively find frequent itemset.

The performance parameters for comparison will usually be time complexity and I/O overhead incurred on the CPU. Since the amount of data is scalable, time required for mining frequent itemset plays a crucial role in enhancing the efficiency.

## 3. SCOPE FOR SCALABILITY IN MINING FREQUENT ITEMSET

There are three main areas which can be focused upon to find the association rules efficiently. By efficient we mean to reduce time complexity, incur less I/O overhead as well as to efficiently use the storage space. Due to large amount of data available in data warehouses, we should be able to mine data that is scalable i.e. that can grow in size.

As mentioned above, finding frequent itemset is a time consuming process. So the first area of focus should be fast generation and pruning of candidate itemset. Thus by speeding up this step we will enhance the generation of frequent itemset. There are various approaches which have been followed such as depth first search, breadth first search, hybrid, partitioning the database as well as sampling. A large number of increasingly efficient algorithms to mine frequent itemset have been developed over the years [3], [4], [5], [6], [7].

Another area of focus is the data structure which has been used for storing the intermediate candidate itemset in the first step. It should be such that the data which is stored there can be retrieved fast. Several data structures have been used for mining frequent itemset. They can be divided into two main categories as array-based and tree-based. In array-based representation, transactions are stored as arrays of items in the memory. For example, H-Mine uses an array structure called Hstruct [4]. In tree-based representation, variants of prefix trees are used. A prefix tree can be used to organize the transactions by grouping and storing transaction data in memory. In [6], transactions are first grouped using a compressed prefix tree and mapped to an array-based data structure, which is then used for the mining process.

The third issue on which the research is going on is parallel mining. Parallelism can be implemented at various stages. However synchronization can be problem while applying parallelism. But as finding frequent itemset is most expensive, since the number of itemset grows exponentially with the number of items, time complexity can be greatly reduced if parallelism is applied while finding frequent itemset. In this paper we mainly concentrate on the first issue i.e. quickly finding frequent itemset.

In this paper we elaborate on the various traversal approaches which can be used for finding the candidate itemset. However, for mining the frequent itemset efficiently, we have to use a combination of effective traversal scheme, an effective but simple data structure as well as concepts of parallel mining.

## 4. COMMONLY USED ASSOCIATION RULE MINING ALGORITHMS

### 4.1 Breadth First approach

In Breadth First or bottom-up approach, the computation starts from frequent 1-itemsets (the minimum length frequent itemset) and continues until all maximal (length) frequent itemset are found. Because of its way of working, it is also known as hierarchical algorithm. During the execution, every frequent itemset is explicitly considered. Such algorithms perform well when all maximal frequent itemset are short. However, performance drastically decreases when some of the maximal frequent itemset are relatively long. The classical Apriori algorithm proposed by Agrawal et al in 1993[8], its enhancement AprioriTid algorithm [9] are all examples of breadth first approach. Apart from this there are also hash based techniques such as DHP algorithm(Direct hashing and pruning) proposed by Park et al in 1995[10] and Tree projection algorithm[11].The partition algorithm[12] and DIC(Dynamic Itemset counting)[13] are also categorized as breadth first algorithms.

The Apriori and AprioriTid algorithms generate the candidate itemset to be counted in a pass by using only the itemset found large in the previous pass {without considering the transactions in the database.} The basic concept behind this is that any subset of a large itemset must be large. The AprioriTid [9] algorithm has the additional property that the database is not used at all for counting the support of candidate itemset after the first pass. Rather, an encoding of the candidate itemset used in the previous pass is employed for this purpose. AprioriTid has the enhanced feature that it replaces a pass over the original dataset by a pass over the candidate set. Another algorithm DHP [10] is a hash based algorithm and is especially effective for generating candidate 2-itemsets, where the number of candidate 2-itemsets in terms of magnitude is smaller than the previous algorithms, thus resolving the performance bottleneck. The DHP algorithm (Park et al, 1995) tries to reduce the number of candidates by collecting approximate counts in the previous level. Like Apriori it requires as many database passes as the longest itemset. It also trims the transaction database at a much earlier stage of the iteration, thereby reducing the computational cost for later stages significantly. The Tree projection algorithm[11] proposed by Agrawal et al in 2001 represents frequent patterns as nodes of a lexicographic tree and uses the hierarchical structure of the lexicographic tree to successively project transactions and uses matrix counting on the reduced set of transactions for finding frequent patterns. The TreeProjection algorithm outperforms the Apriori method by more than an order of magnitude. At very low levels of support, the Apriori algorithm runs out of memory and is unable to run to completion. This behavior illustrates the memory advantages of using a lexicographic Tree projection algorithm over the hash tree implementation of the Apriori algorithm. As shown in [10], Fig.1 compares the Apriori and DHP over the various passes on a standard dataset T15.14.D100.It is observed that DHP, because of its hashing and timely pruning technique requires less than half the amount of time as compared to classical Apriori over 5 passes of candidate itemset generation.

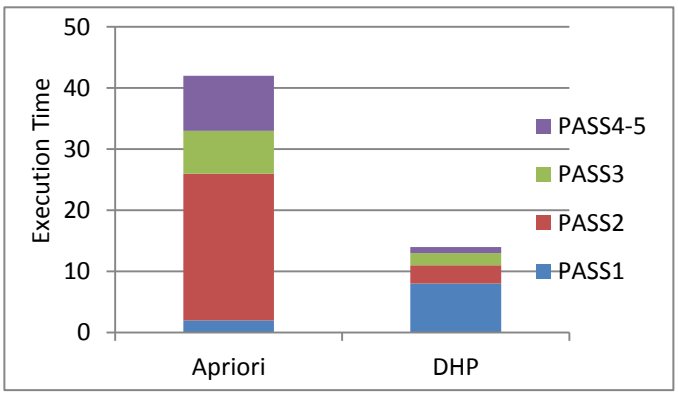


**Fig 1: Execution time of Apriori and DHP**

The partition algorithm proposed in 1995 reads the database at most two times to generate all significant association rules. Contrast this with the previous algorithms, where the database is not only scanned multiple times but the number of scans cannot even be determined in advance. All the partition algorithms such as the classic partition algorithm proposed by Saverese et al [12] and DIC(Direct Itemset counting) proposed by Brin [13] work well on large datasets. In partition algorithm [12], during the first scan, algorithm generates a set of all potentially large itemsets by scanning the database once. This set is a superset of all large item sets, i.e. it may contain false positives, but no false negatives are reported. During the second scan, counters for each of these item sets are setup and their actual support is measured in one scan of the database. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase. One major problem with Partition is that as the number of partitions increases, the number of locally frequent itemsets, increases. While this can be reduced by randomizing the partition selection, randomized partitions will have a large number of frequent itemsets in common. Partition can thus spend a lot of time in performing these redundant intersections.DIC [13] algorithm is also adopted by the thought of dividing the database into several partitions, each partition is marked at the beginning in the process of scanning the database, candidate items can be added in each marked partition, the support can be calculated when the itemsets are calculated.DIC separates the strict restriction between counting and generating candidates. Whenever a candidate reaches min-sup, DIC starts generating additional candidates based on it. The main bottleneck of data set partitioning algorithm is that the execution time is long and frequent itemsets generated are not very accurate. But this type of algorithm has a high degree of parallelism; just scanning the database twice, greatly reduces I / O operation to improve the efficiency of the algorithm.

Overall, we can conclude that the breadth-first algorithm has the shortcoming of generating a large number of candidate items and need to repeatedly scan the database. This is definitely not suitable for databases with large number of itemsets.

## 4.2 Depth First Approach

Depth first search has the advantage that it is not necessary to re-create the projected transactions for each level-k of a search tree. The depth first projection technique provides locality of data access, which can exploit multiple levels of cache. Some of the depth first algorithms are FP-growth [14], OIP [15] DepthProject [16] and Eclat [17].

In 2000 Han et al proposed Frequent pattern tree (FP-tree) structure, which is an extended prefix tree structure, used for mining the complete set of frequent patterns by pattern fragment growth. It encodes the dataset using a compact data structure FP-tree and extracts frequent itemsets directly from this structure. This algorithm is about an order of magnitude faster than the Apriori algorithm. However, the number of conditional FP-trees is in the same order of magnitude as number of frequent itemsets. The algorithm is not scalable to sparse and very large databases. The OIP algorithm [15] OpportuneProject mines complete set of frequent item sets, which is efficient on both sparse and dense databases at all levels of support threshold, and scalable to very large databases. It opportunistically chooses between two different data structures, array based or tree-based, to represent projected transaction subsets, and heuristically decides to build unfiltered pseudo projection or to make a filtered copy according to features of the subsets.DepthProject [16] employs a selective projection and uses the horizontal bit string representation for projected transaction subsets. Fig.2 [14] shows the comparison of Tree Projection(Breadth first) and FP-Tree(Depth first) approach of runtime on T25:I10:D10K with 1K items as dataset. It shows that FP-growth is better than TreeProjection when support threshold is very low and database is quite large.

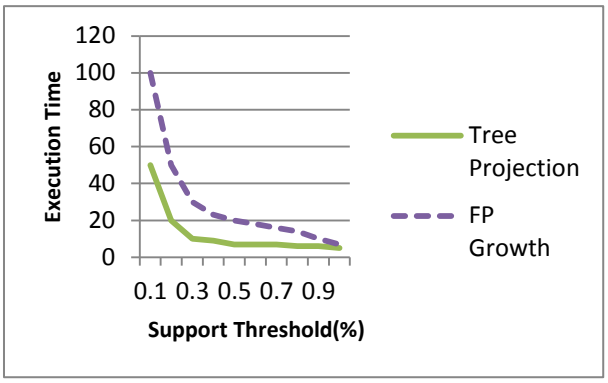


**Fig 2: Execution time of FP Growth and Tree Projection**

The Eclat algorithm[17] proposed by Zaki et al is based on a parallel approach and partitions the database. It incorporates some features of clustering as well as parallel mining.The algorithm uses a scheme to cluster related frequent itemsets together, and to partition them among the processors. At the same time it also uses a different database layout which clusters related transactions together, and selectively replicates the database so that the portion of the database needed for the computation of association rules is local to each processor. After the initial set-up phase, the algorithm eliminates the need for further communication or synchronization thus eliminating the problem of syncronization faced in parallel mining. The algorithm further scans the local database partition only three times, thus minimizing I/O overheads.The gist is that depth first approach works well for large amount of data as compared with bredath first approach and so it can be applied for scalable datasets.

## 4.3 Hybrid Approach

There is also a hybrid approach wherein at some point breadth first approach is used and at some point depth first approach is applied in the same process. Some of the algorithms which work on hybrid approach are AprioriHybrid [18], Viper[19] , Pincer Search [20]and MaxClique [21].Fig.3 below shows an example of the two approaches[20].

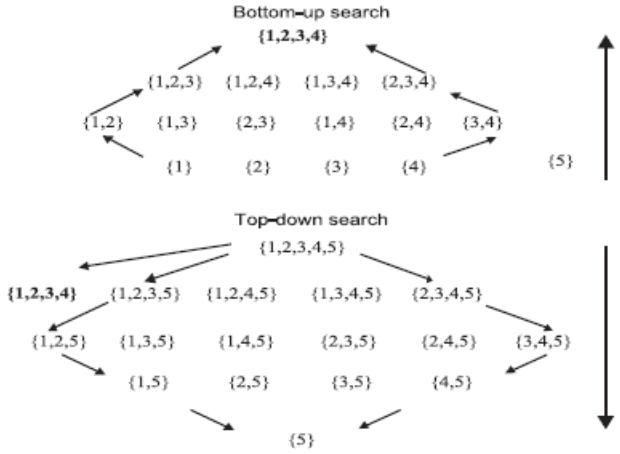


**Fig 3: One Way searches**

AprioriHybrid [18] uses Apriori in initial phase and then switch to AprioriTid in later passes .It performs better than Apriori and AprioriTid in most of the cases. However the challenge to operate hybrid algorithm is to determine the switch over point. The advantage of AprioriHybrid over Apriori depends on how the size of the Ck (candidate k itemset) decline in the later passes. If Ck remains large until nearly the end and then has an abrupt drop, there gain will be no gain by using AprioriHybrid since we can use AprioriTid only for a short period of time after the switch. Another algorithm VIPER [19] is also based on hybrid approach. It is general-purpose, making no special requirements of the underlying database. VIPER stores data in compressed bit-vectors called "snakes" and integrates a number of novel optimizations for efficient snake generation, intersection, counting and storage. There are workload regions where VIPER outperforms an optimal, but practically infeasible, horizontal mining algorithm. The Pincer Search algorithm [20] also combines both the bottom-up and the top-down searches. The primary search direction is still bottom-up, but a restricted search is also conducted in the top-down direction. This search is used only for maintaining and updating a new data structure, the maximum frequent candidate set. It is used to prune early candidates that would normally encounter in the bottom-up search. A very important characteristic of the algorithm is that it does not require explicit examination of every frequent itemset. Therefore the algorithm performs well even when some maximal frequent itemsets are long. As its output, the algorithm produces the maximum frequent set, i.e., the set containing all maximal frequent itemsets, thus specifying immediately all frequent itemsets. MaxClique [21] uses a hybrid traversal which contains a look-ahead phase followed by a pure bottom-up phase. The look-ahead phase consists of extending the frequent 2-itemsets until the extended itemset becomes infrequent. After the look-ahead phase, an Apriori-like traversal is executed.

Table 1 gives the performance comparison for some major algorithms based on various criteria such as traversal technique used,data structures and number of scans.

**Table 1: Comparison of some major algorithms**

| **Algorithm** | **Traversal Technique** | **Data Structure** | **Number of database scans** |
|---|---|---|---|
| Apriori | Breadth First | Hash Tree | K |
| DHP | Breadth First | Hash Tree | K |
| Partition | Breadth First | None | 2 |
| DIC | Breadth First | Prefix tree | <=K |
| FP tree | Depth First | Prefix tree | 2 |
| Eclat | Depth First | None | >=3 |
| Pincer Search | Hybrid | Array and linked list | >=4 |
| MaxClique | Hybrid | None | >=3 |

Fig. 4 shows the current scenario of research done on frequent itemset mining [22] in context to year of publication and its number of citations. However, as the data continue to increase in complexity (which includes size, type, and location of data), parallel computing will be essential in delivering fast interactive solutions for the overall KDD process. Applying parallel mining in collaboration with an effective data structure for storing candidate sets can be exploited for a scalable approach in mining association rules.

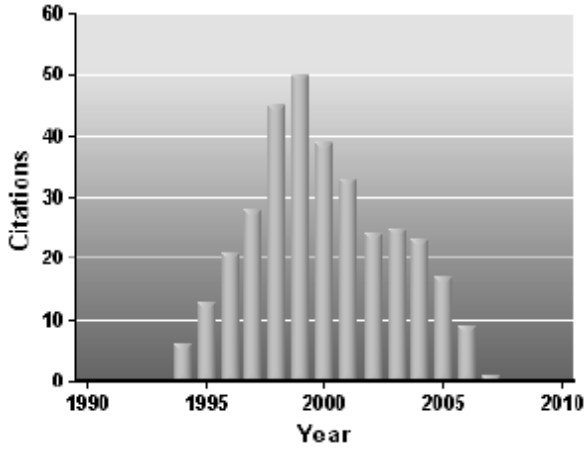


**Fig 4: Current status of research**

## 5. CONCLUSION

In this paper, we have discussed the various algorithmic methods based on the traversal techniques for generating candidate itemset.

We can conclude that there can be a scalable approach for mining frequent itemset if we explore the areas of parallel mining in combination with an efficient data structure. Also if we can efficiently use the main memory for storing the database (such as in partitioning) it will highly reduce the I/O scans thereby further increasing the speed. From the above study we conclude that almost each ARM algorithm has some favorable environment that provides high performance and faster computation of frequent itemset. These factors vary from algorithm to algorithm. Depending upon the application and the amount of data available, these various techniques can be applied in combination with different data structures. Although lot of work has been done in this research area, there can be a hybrid approach wherein we can combine the hybrid traversal technique along with a data structure and mine it in parallel.

## 6. REFERENCES

[1] R. Agrawal, H.Mannila, R. Srikant, H. Toivonen, and A. I.Verkamo, 1996,Fast discovery of association rules, In

Advances in Knowledge Discovery and Data Mining, MIT Press.

[2] R. Agrawal, T. Imielinski, and A.Swami,1993,Mining association rules between sets of items in large databases, In ACM SIGMOD Intl. Conf. Management of Data.

[3] J. Han, J. Pei, and Y. Yin,2000,Mining Frequent Patterns without Candidate Generation, Proc. of the ACM SIGMOD, Dallas, TX.

[4] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, 2001,HMine:Hyper-Structure Mining of Frequent Patterns in Large Databases, Proc. of IEEE ICDM, San Jose, California.

[5] R. Agrawal and R. Srikant,1994,Fast Algorithms for Mining Association Rules, Proc. of the 20th Int. Conf. on VLDB, Santiago, Chile.

[6] Y. G. Sucahyo and R. P. Gopalan,2003,CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth, Proc. of 14th Australasian Database Conference, Adelaide, Australia.

[7] M. J. Zaki,2000,Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering, (12, May/June 2000) 372-390.

[8] Agrawal R, Imielinski T, Swami A, 1993,Mining Association Rules between Sets of Items in Large Databases, Proc. of the 1993 ACMSIGMOD Conference. Washington D C, 207-216.

[9] Agrawal R, Srikant R,1994,Fast Algorithm for Mining Association Rules, Proc of the 20th Very Large Data Bases (VLDB 94) Conference. Santiago, Chile.

[10] Park J S, Chen M S, Yu P S, 1995, An effective hash based algorithm for mining association rules, In Proc. Of 1995 ACM SIGMOD. San Jose, 175-186.

[11] Agarwal R, Aggarwal C, Prased V V V,2001,A tree projection algorithm for generation of frequent itemsets, Parallel and distributed Computing, 61(3): 350-371.

[12] Savasere A, Omieeinski E, Navathe S,1995,An efficient algorithm for mining association rules in large databases, Proc. of the 21st International Conference on Very Large Databases. Zurich, Switzerland, 432-443.

[13] Brin S, Motwani R, Ullman J D,1997,Dynamic Itemset counting and implication rules for market basket data, Proc. of the 1997 ACM SIGMOD International Conference on Management of Data.

[14] Han J, Pei J, Yin Y, 2000, Mining frequent patterns without candidate generation, In Proc. of the 2000 ACM SIGM0D Conference on Management of Data. Dallas, TX.

[15] Liu J, Pan Y, Wang K, et al, 2002,Mining frequent item sets by opportunistic projection,Proc Of the Eighth ACM SIGKDD Intl. Conf on Knowledge Discovery and Data Ming. Alberta, Canada, 229-238.

[16] R. Agarwal, C. Aggarwal and V. V. V. Prasad,2000, Depth first generation of long patterns, in Proc.of SIGKDD Conference.

[17] Mohammed Javeed Zaki, Srinivasan Parthasarathy and Wei Li,1997,A Localized Algorithm for Parallel Association Mining, Proc. of 9th Annual ACM Symposium on Parallel Algorithms and Architectures, 321-330.

[18] Agrawal R, Srikant R,1994,Fast Algorithm for Mining Association Rules in Large Databases, San Jose, IBM Alma den Research Center.

[19] Pradeep Shenoy, Jayant Harista, S.Sudarshan, Gaurav Bhalotia, Mayank Bawa, Devavrat Shah,2000,Turbo-charging vertical mining of large databases, Proc. of 2000 ACM SIGMOD International conference on Management of Data, Pg 22-33.

[20] Dao-I Lin, Zvi M. Kedem,2002,Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set, IEEE Transactions on Knowledge and Data Engineering, Vol 14 ,(May 2002),Pg. 552-566.

[21] M. Zaki, S. Parthasarathy, M. Ogihara, and W.Li,1997,New algorithms for fast discovery of association rules, In Proc. 3rd KDD.

[22] R.Shrikant and R.Agrawal ,1996,Mining quantitative association rules in large relational tables, SIGMOD 1996.