A Taxonomy of Classical Frequent Item set Mining Algorithms

Bharat Gupta and Deepak Garg

***Abstract*—These instructions Frequent itemsets mining is one of the most important and crucial part in today's world for every transactional database. Many researchers have introduced many algorithms for mining frequent itemsets over the last few decades. Firstly the most known and powerful horizontal database layout based algorithm introduced is Apriori algorithm then various improvements also been introduced on the basis of this approach. Then the tree projection based algorithms are introduced for the efficient storage and retrieval of the datasets. Tree projection based algorithms include FP-Growth, H-Mine and many more. Various hybrid algorithms also been introduced for taking the advantage of vertical as well as tree projection algorithms. This classification aims to enhance the understanding of various present techniques and direction of research in this area. This article attempts to provide theoretical aspects of the key techniques.**

***Index Terms*—Data mining, Frequent itemsets, Apriori algorithms, FP-Tree algorithms.**

I. Introduction

Association rule mining is one of the most important and crucial task in data mining. A database stores large number of records according to the transaction ids. Association rule mining depends upon the relations between the records and items which can be efficiently mined by the frequent itemset mining. A frequent itemset is a pattern which occurs greater than the minimum support in the database [1]. Like in the grocery retailer shop each customer purchase a number of items like tea, coffee, sugar and many more and another customer purchase may other or same items for the grocery shop. These items are stored in the database according to the transaction ids. Thus data mining used to analyze the behavior of customers according to their purchasing behavior, occurrence of items and various other patterns. like if two items say bread and milk purchased by many customers many times which is greater than the minimum support and as considered as frequent then according to association rules, we can say that if bread is purchased then milk is also purchased most of the cases and vice versa. In this way pattern mining algorithms mines the repeated patterns known as frequent patterns. This type of analysis may apply for disease treatments, natural disasters, web access patterns and many more by just collecting the historical data. In this way association rule mining merely depends upon the frequent itemset mining than make association rule according to frequent patterns.

Many algorithms have been proposed from many decades, like horizontal layout based algorithms [1, 2, 4, 7, 10], vertical layout based algorithms [9] and projected layout based algorithms [5, 8]. In this article, will discuss important classical layout based algorithms and keep our article clean and focused by excluding closed, minimal, maximal or incremental algorithms. This article provides a deep discussion of features of algorithms from each layout based category along with strengths and weakness. This article tries to explain the following:

Approaches: The various approaches categorized into vertical layout based algorithms like Apriori and all its variations, horizontal layout based algorithms like éclat and projected based layout based algorithms like FP-Tree algorithm, H-Mine algorithm and others.

Issues: The various scalability and handling issues attached with various techniques, the functionality of specific techniques corresponding to specific dataset. The time and space constraints attached to specific approach.

The further organization of this paper is as follows. In Section 2, we briefly define the problem statement for finding the frequent itemsets from transactional database. Sections 3 define the literature survey of existing techniques based upon the horizontal layout based database, vertical layout based database and projected layout based database.. Section 4 defines the analysis and discussing of these algorithms. Section 5 compares the existing techniques. Section 6 concludes the paper.

II. Problem Statement

The problem of mining association rules over market basket analysis was introduced in [2] .i.e. finding associations between the items that are present in the transaction from the database. The database may be from any retail shop, medical or from any other applications [11]. As defined in [3] the problem is stated as follows: Let $I = i_1, i_2, \ldots . i_m$ be a set of literals, called items and m is considered the dimensionality of the problem. Let D be a set of transactions, where each transaction T is a set of items such that $T \in I$. A unique identifier TID is given to each transaction. A transaction T is said to contain X, a set of items in I. $X \in T$ An association rule is an implication of the form "$X \rightarrow Y$ ", where $\in I$, $Y \in I$ and $X \cap Y = \emptyset$. An itemset X is said to be large or frequent if its support s is greater or equal than a given minimum support threshold σ. An itemset X satisfies a constraint C if and only if C(X) is true. The rule $X \rightarrow Y$ has a support s in the transaction set D if s% of the transactions in D contain $X \cup Y$. In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases. It is said that the rule $X \rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that

Manuscript received May 22, 2011; revised July 18, 2011.

The authors are with Computer Science Department, Thapar University, Punjab, India. (e-mail:bharatgupta35@gmail.com)

contain X also contain Y . In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedentX. The problem of discovering all association rules from a set of transactions D consists of generating the rules that have a support and confidence greater than a given threshold. These rules are called Strong Rules. This association-mining task can be broken into two steps:

- Finding the frequent k-itemset from the large database.
- Generate the association rule from these frequent item sets.

In this paper, we focus exclusively on the first step: generating frequent itemsets algorithms.

III. Literature Riview

A. Algorithms for Mining from Horizontal Layout Database

In this type of database, each row of database represents a transaction which has a transaction identifier (TID), followed by a set of items. One example of horizontal layout dataset is shown in diagram:

TABLE I: Horizontal Layout based Database

TID	ITEMS
T1	I1, I2, I3, I4, I5, I6
T2	I1, I2, I4, I7
T3	I1, I2, I4, I5, I6
T4	I1, I2, I3
T5	I3, I5

1) Apriori Algorithm

Apriori algorithm [3] is one of the most important algorithms which set the breed of all frequent mining algorithms. The working of Apriori algorithm is fairly depends upon the Apriori property which states that" All nonempty subsets of a frequent itemsets must be frequent" [3]. It also described the anti monotonic property which says if the system cannot pass the minimum support test, all its supersets will fail to pass the test [3]. Therefore if the one set is infrequent then all its supersets are also frequent and vice versa. This property is used to prune the infrequent candidate elements. Apriori employs a bottom up Breadth, First approach and also an iterative approach to mine the frequent elements, where k-itemsets are used to explore (k+1)-itemsets. The feature first invented by [2] in Apriori algorithm is used by the many algorithms for frequent pattern generation. The basic steps to mine the frequent elements are as follows:

- Generate and test: In this first find the 1-itemset frequent elements L_1 by scanning the database and removing all those elements from C_1 which cannot satisfy the minimum support criteria.
- Join step: To attain the next level elements C_k join the previous frequent elements by self join i.e. $L_{k-1} * L_{k-1}$ known as Cartesian product of L_{k-1}.
- Prune step: C_k is the superset of L_k so members of C_k may or may not be frequent but all $K-1$ frequent itemsets are included in C_k thus prunes the C_k to fnd K frequent itemsets with the help of Apriori property. The set of frequent elements known as L_k. Step 2 and 3 is repeated until no new candidate set is generated.

To illustrate this, suppose n frequent 1-itemsets and minimum support is 1 then according to Apriori based algorithm will generate $n^2 + (n\ 2)\text{candidate}\ 2 - \text{itemset}\ (n\ 3)\text{candidate}\ 3 - \text{itemset and so on}$ [11]. The total number of candidates generated is greater than $\sum_{k=1}^{n}(n\ k)$Therefore suppose there are 1000 elements then 1499500 candidate are produced in 2 itemset frequent and 166167000 are produced in 3-itemset frequent [11]. The Apriori property can be illustrate as follows: Given the set of frequent 2-itemsets {I1, I2}; {I1, I3}; {I1 I4}; {I3, I4}, by joining 2-itemsets, the set of potential candidates of size 3, {I1, I2, I3}; {I1, I3, I4} can be generated. Because that the subset {I2, I3} of {I1, I2, I3} is not frequent, {I1, I2, I3} surely cannot be frequent and is pruned from the set of candidates. In this way, Apriori algorithm avoids wasting computation in counting the itemsets that must not be frequent by judging from their subsets.

It is no doubt that Apriori algorithms successfully find the frequent elements from the database. But as the dimensionality of the database increase with the number of items then:

- More search space is needed and I/O cost will increase.
- Number of database scan is increased thus candidate generation will increase results in increase in computational cost.

Therefore many variations have been takes place in the Apriori algorithm to minimize the above limitations arises due to increase in size of database. These subsequently proposed algorithms adopt similar database scan level by level as in Apriori algorithm, while the methods of candidate generation and pruning, support counting and candidate representation may differ. The algorithms improve the Apriori algorithms by:

- Reduce passes of transaction database scans
- Shrink number of candidates
- Facilitate support counting of candidates

These algorithms are as follows:

2) Direct hashing and pruning (DHP):

It is absorbed that reducing the candidate items from the database is one of the important task for increasing the efficiency. Thus a DHP technique was proposed [7] to reduce the number of candidates in the early passes C_k for $k > 1$ and thus the size of database. In this method, support is counted by mapping the items from the candidate list into the buckets which is divided according to support known as Hash table structure. As the new itemset is encountered if item exist earlier then increase the bucket count else insert into new bucket. Thus in the end the bucket whose support count is less the minimum support is removed from the candidate set. In this way it reduce the generation of candidate sets in the earlier stages but as the level increase the size of bucket also increase thus difficult to manage hash table as well candidate set.

3) Partitioning algorithm:

Partitioning algorithm [1] is based to find the frequent elements on the basis partitioning of database in n parts. It overcomes the memory problem for large database which do

not fit into main memory because small parts of database easily fit into main memory. This algorithm divides into two passes,

1. In the first pass whole database is divided into n number of parts.
2. Each partitioned database is loaded into main memory one by one and local frequent elements are found.
3. Combine the all locally frequent elements and make it globally candidate set.
4. Find the globally frequent elements from this candidate set.

It should be noted that if the minimum support for transactions in whole database is min_sup then the minimum support for partitioned transactions is min-sup number of transaction in that partition.

A local frequent itemset may or may not be frequent with respect to the entire database thus any itemset which is potentially frequent must include in any one of the frequent partition.

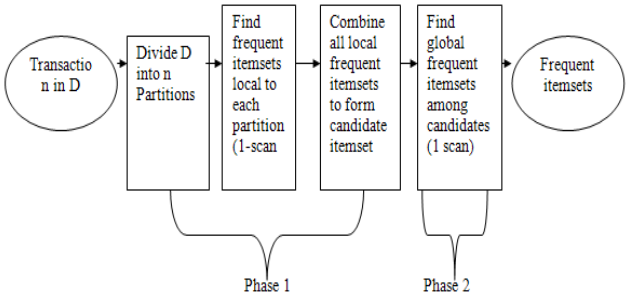

Fig. 1. Mining by partitioning the data [13]

As this algorithm able to reduce the database scan for generating frequent itemsets but in some cases, the time needed to compute the frequency of candidate generates in each partitions is greater than the database scan thus results in increased computational cost.

4) Sampling algorithm:

This algorithm [10] is used to overcome the limitation of I/O overhead by not considering the whole database for checking the frequency. It is just based in the idea to pick a random sample of itemset R from the database instead of whole database D. The sample is picked in such a way that whole sample is accommodated in the main memory. In this way we try to find the frequent elements for the sample only and there is chance to miss the global frequent elements in that sample therefore lower threshold support is used instead of actual minimum support to find the frequent elements local to sample. In the best case only one pass is needed to find all frequent elements if all the elements included in sample and if elements missed in sample then second pass are needed to find the itemsets missed in first pass or in sample [13].

Thus this approach is beneficial if efficiency is more important than the accuracy because this approach gives the result in very less scan or time and overcome the limitation of memory consumption arises due to generation of large amount of datasets but results are not as much accurate.

5) Dynamic Itemset Counting (DIC):

This algorithm [4] also used to reduce the number of database scan. It is based upon the downward disclosure property in which adds the candidate itemsets at different point of time during the scan. In this dynamic blocks are formed from the database marked by start points and unlike the previous techniques of Apriori it dynamically changes the sets of candidates during the database scan. Unlike the Apriori it cannot start the next level scan at the end of first level scan, it start the scan by starting label attached to each dynamic partition of candidate sets.

In this way it reduce the database scan for finding the frequent itemsets by just adding the new candidate at any point of time during the run time. But it generates the large number of candidates and computing their frequencies are the bottleneck of performance while the database scans only take a small part of runtime.

Assumption [12, 13]: The performance of all the above algorithms relies on an implicit assumption that the database is homogenous and thus they will not generate too many extra candidates than Apriori algorithm does. For example, if all partitions in Partition algorithm are not homogenous and nearly completely different sets of local frequent itemsets are generated from them, the performance cannot be good.

B. Algorithms for Mining from Vertical Layout Database

In vertical layout data set, each column corresponds to an item, followed by a TID list, which is the list of rows that the item appears. An example of vertical layout database set is as shown in diagram for the table1.

TABLE II: VERTICAL LAYOUT BASED DATABASE

ITEM	TID_list
I1	T1, T2, T3, T4
I2	T1, T2, T3, T4
I3	T1, T4, T5
I4	T1, T2, T3
I5	T1, T3, T5
I6	T1, T3
I7	T2

1) Eclat algorithm

It is a set intersection, depth first search algorithm [9], unlike the Apriori. It uses vertical layout database and each item use intersection based approach for finding the support. In this way, the support of an itemset P can be easily computed by simply intersecting of any two subsets $Q, R \subseteq P$, such that $P = Q \cup R$.

In this type of algorithm, for each frequent itemset i new database is created Di. This can be done by finding j which is frequent corresponding to i together as a set then j is also added to the created database i.e. each frequent item is added to the output set. It uses the join step like the Apriori only for generating the candidate sets but as the items are arranged in ascending order of their support thus less amount of intersection is needed between the sets. It generates the larger amount of candidates then Apriori because it uses only two sets at a time for intersection [9]. There is reordering step takes place at each recursion point for reducing the candidate itemsets.

In this way by using this algorithm there is no need to find the support of itemsets whose count is greater than 1because Tid-set for each item carry the complete information for the corresponding support. When the database is very large and the itemsets in the database corresponding also very large then it is feasible to handle the Tid list thus it produce good

results but for small databases its performance is not up to mark.

C. Algorithms for mining from projected layout based database

The concept of projected database was proposed and applied to mine the frequent itemsets efficiently because early approaches are able to mine the frequent itemsets but use large amount of memory. This type of database uses divide and conquer strategy to mine itemsets therefore it counts the support more efficiently then Apriori based algorithms. Tree projected layout based approaches use tree structure to store and mines the itemsets. The projected based layout contains the record id separated by column then record.

Tree projection is defined as the lexicographic tree with nodes contains the frequent itemsets [14]. The lexicographic trees usually follow the ascending order for saving the frequent itemsets according to the support for better mining [5].

Tree Projection algorithms based upon two kinds of ordering breadth-first and depth-first. For breath-first order, nodes are constructed level by level in the lexicographic tree for frequent itemsets [11]. In order to compute frequencies of nodes (corresponding frequent itemsets) at k level, tree projection algorithm maintained matrices at nodes of the k-2 level and one database scan was required for counting support [5]. Every transaction is projected by node sequentially. The projected set of transaction for reduced set is used to evaluate frequency.

For depth-first order, database is projected along the lexicographic tree and also requires fitting into main memory [13]. The advantage is that the projected database will become smaller along the branch of the lexicographic tree while the breadth-first needs to project the database from the scratch at each level.

The disadvantage of depth-first is obvious that it needs to load database and projected databases in memory. The breadth-first method will also meet the memory bottleneck when the number of frequent items is large and the matrix is too large to fit in memory [5].

1) FP-Growth Algorithm

FP-tree [6] is based upon recursively divide and conquers strategy for mining the frequent itemsets from the database. FP-tree needs 2 database scan for finding he frequent item set completely. Unlike the Apriori algorithm it mines the frequent itemsets without generation of candidate itemsets. It compresses the whole database in to compressed form as in the structure known as FP-tree and the process is known as FP-Growth. Due to its non generation of candidate itemsets, improves the multi database scan problem.

The construction of frequent patterns is based upon the construction of Conditional pattern base and the conditional FP-tree. Its first scan is same as the Apriori algorithm i.e. it scan the database and count the frequency of each item then select the 1-itemset frequent. Then all the items are stored in the list according to the descending order of the frequency known as F-list. After the F-list is created create the tree for a every transaction according to the items in the F-list i.e. in the descending order of the frequency. A pointer is maintained for each item from the F-list (header table). For every new item new node is created and if same item is encounter at same place then increment the support value attached with each node separated by column by 1.

Conditional pattern base is created which is the sub database consists of set of prefix path in the FP-tree co-occurring with the suffix pattern [6]. After conditional pattern base conditional FP-tree is constructed for each pattern base and mines the tree recursively for frequent itemsets.

Due to its divide and conquer strategy it transforms the problem of finding long frequent patterns to search for shorter ones recursively [12]. But for the large database it becomes unrealistic to construct the whole FP-tree in the main memory as due to limited size of memory.

2) H-mine Algorithm

H-mine [8] algorithm is the improvement over FP-tree algorithm as in H-mine projected database is created using in-memory pointers. H-mine uses an H-struct new data structure for mining purpose known as hyperlinked structure. It is used upon the dynamic adjustment of pointers which helps to maintain the processed projected tree in main memory therefore H-mine proposed for frequent pattern data mining for data sets that can fit into main memory. It has polynomial space complexity therefore more space efficient then FP-growth and also designed for fast mining purpose. For the large databases, first in partition the database then mine each partition in main memory using H-struct then consolidating global frequent pattern [8]. If the database is dense then it integrates with FP-Growth dynamically by detecting the swapping condition and constructing the FP-tree.

This working ensures that it is scalable for both large and medium size databases and for both sparse and dense datasets [14]. The advantage of using in-memory pointers is that their projected database does not need any memory the memory required only for the set of in-memory pointers.

IV. Analysis and Discussion

- All the algorithms produce frequent itemsets on the basis of minimum support.
- Apriori algorithm is quite successful for market based analysis in which transaction may large but frequent items generated is small in number.
- The Apriori variations (DHP, DIC, Partition, and Sample) algorithms among them DHP tries to reduce candidate itemsets and others try to reduce database scan.
- DHP works well at early stages and performance deteriorates in later stages and also results in I/O overhead.
- For DIC, Partition, sample algorithm performs worse where database scan required is less then generating candidates.
- Vertical Layout based algorithms claims to be faster than Apriori but require larger memory space then horizontal layout based because they needs to load candidate, database and TID list in main memory.
- For projected layout based algorithms, performs better then all discussed above because of no generation of candidate sets but the pointes needed to store in memory require large memory space.

V. Comparision of Algorithms

TABLE III: COMPARISONS OF ALL TECHNIQUE

	Horizontal layout based algorithms					Vertical layout based algorithm	Projected layout based algorithms	
Algorithm / Parameter	Apripri Algorithm	DHP algorithm	Partition Algorithm	DIC algorithm	Sample Algorithm	Eclat algorithm	FP-tree Algorithm	H-mine Algorithm
Storage Structure	Array based	Array Based	Array based	Array based	Array based	Array based	Tree based	Tree based
Technique	Use Apriori property and join and prune method	Use hashing technique for finding frequent itemsets	Partition the database for finding local frequent item first	Based upon dynamic insertion of candidate items.	Pick any random sample for checking frequency of whole database at lower threshold support	Use intersection of Transaction ids list for generating candidate itemsets.	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.	It uses the hyperlink pointers to store the partitioned projected database in main memory.
Memory utilization	Due to large amount of candidate are produced so require large memory space	Require less space at earlier passes but more in later stages	Each partition is easily occupy in main memory	Require different amount of memory at different point of time	Very less amount of memory is needed	Require less amount of memory compare to Apriori if itemsets are small in number	Due to compact structure and no candidates generation require less memory	Memory is utilized according to needs and partitions of projected database
Databases	Suitable for sparse datasets as well as dense datasets	Suitable for medium databases	Suitable for large databases	Suitable for medium and low databases	Suitable for any kind of dataset but mostly not give accurate results	Suitable for medium and dense datasets but not suitable for small datasets.	Suitable for large and medium datasets	Suitable for sparse and dense datasets.
Time	Execution time is more as time wasted in producing candidates at every time	Execution time is small for small databases.	Execution time is more because of finding locally frequent then globally frequent	Execution time is small because dynamic itemset are added according to situation.	Execution time is very much small.	Execution time is small then Apriori algorithm	Execution time is large due to complex compact data structure	Execution time is large then FP-tree and others because of partition the database.

VI. Conclusion

The main objective of this research survey was to analyze the assessment of mining association rules using frequent item sets. It was discussed various algorithms like Apriori algorithm, partition algorithm, DHP algorithm, DIC algorithm (applicable on Horizontal layout database), Eclat algorithm (applicable in Vertical layout data base) and FP-tree algorithm, H-mine algorithm (based on tree projection). This analysis made a significant to the search of improving the efficiency if frequent itemsets. Various classical algorithms strengths and weakness were discussed and found some algorithms produce candidates sets and some not, various algorithms are the improved version of the classical algorithms. In the future also by using above discussed strengths, weakness, properties of various algorithms can helps to improve the other hybrid or any other frequent mining algorithms.

References

[1] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In Proc. Int'l Conf. Very Large Data Bases (VLDB), pages 432–443, Sept. 1995.

[2] Aggrawal.R, Imielinski.t, Swami.A, Mining Association Rules between Sets of Items in Large Databases, Proceedings of the 1993 ACM SIGMOD Conference Washington DC, USA, May 1993.

[3] Agrawal.R and Srikant.R. Fast algorithms for mining association rules. In Proc.Int'l Conf. Very Large Data Bases (VLDB), pages 487–499, Sept. 1994.

[4] Brin.S, Motwani. R, Ullman. J. D, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), pages 255–264, May 1997.

[5] C. Borgelt. An Implementation of the FP- growth Algorithm. Proc. Workshop Open Software for Data Mining (OSDM'05 at KDD'05, Chicago,IL),1–5.ACMPress, New York, NY, USA 2005.

[6] Han.J, Pei.J, and Yin. Y., mining frequent patterns without candidate generation. In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), 2000

[7] Park. J. S., M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), pages 175–186, San Jose, CA, May 1995.

[8] Pei.J, Han.J, Lu.H, Nishio.S., Tang. S. and Yang. D., H-mine: Hyper-structure mining of frequent patterns in large databases. In Proc. Int'l Conf. Data Mining (ICDM), November 2001.

[9] C.Borgelt. Efficient Implementations of Apriori and Eclat. Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90, Aachen, Germany 2003.

[10] Toivonen.H. Sampling large databases for association rules. In Proc. Int'l Conf. Very Large Data Bases (VLDB), pages 134–145, Bombay, India, Sept. 1996.

[11] Nizar R.Mabrouken, C.I.Ezeife, "A taxonomy of Sequential Pattern Mining Algorithm " Proc in ACM Computing Surveys, Vol 43, No 1, Article 3, Publishing date November 2010.

[12] Yiwu Xie, Yutong Li, Chunli Wang, Mingyu Lu, The Optimization and Improvement of the Apriori Algorithm, In Proc.Int'l Conf International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing 2008

[13] "Data mining Concepts and Techniques" by By Jiawei Han, Micheline Kamber, 2006.

[14] S.P Latha, DR. N.Ramaraj, "Agorithm for Efficient Data Mining", Proceeding on IEEE International Computational Intelligence and Multimedia Aplications 2007, pp. 66-70.