ISSN 2320 – 2602
Volume 4 No.4, April 2015
International Journal of Advances in Computer Science and Technology
Available Online at http://warse.org/ijacst/static/pdf/file/ijacst03442015.pdf

SURVEY OF FREQUENT PATTERN MINING ALGORITHMS IN HORIZONTAL AND VERTICAL DATA LAYOUTS

A.Meenakshi
Department of Computer Applications,
School of Information Technology,
Madurai Kamaraj University, Madurai.
aa_meena@yahoo.com

ABSTRACT

In the modern digital world, there is an accumulation of data for every day. Really, the researchers are bewildered by the massive influx of data .To manage all these massive data is an open challenge for all the researchers in frequent pattern mining. This paper gives an account of the brief history of earlier works of frequent patterns with a detailed description of the core algorithms, highlighting the significant contributions of the different algorithms putforth by different authors. A comprehensive survey of most influential algorithms of literature has been explained and compared with them in the horizontal and vertical data layouts. It covers the main aspects of earlier works of Frequent Pattern Mining, which include a) pros/cons of the existing algorithms b) performance of the prior algorithms and memory space and c) Visualization of the existing algorithms.

KEYWORDS: Frequent Pattern Mining, Association Rule Mining, Horizontal and Vertical data layouts.

1. INTRODUCTION

Over a couple of decades, the modern computing technology has significantly influenced our day-to-day practical lives and has the direct consequences in Business Data Processing and Scientific Computing. Modern Information and communication technology is capable of collecting and generating large amounts of data that need to be analyzed to become useful or profitable. As hardware costs go down, data owners constantly seek the betterment of usage of data mining tools to extract useful knowledge and patterns from the data. In fact, these amounts quickly become too large for immediate human understanding, leading to a situation in which “We are drowning in data, but starved for Knowledge”.

The goal in data mining is to analyze these large amounts of data and discover patterns, rules, and trends that are useful for decision support. It is also called as Knowledge Discovery in Databases (KDD).It was first coined by Gregory Piatetsky-Shapiro in 1989 to describe the process of finding interesting, interpreted, useful and novel data [26]. The Berry and Linoff in 2000 defined the data mining as the process of exploration and analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns and rules. Data mining is an effective and efficient tool for discovery. By mining, the hidden patterns are seen behind the data more accurately, more systematically and more efficiently. However, it is the data miner’s responsibility to distinguish the gold from the dust. Data mining is the process of posing various queries and extracting useful information, patterns and trends often previously unknown from large quantities of data possibly stored in databases. There are two tasks of data mining: Descriptive mining and Predictive mining. Descriptive mining focuses on finding human – interpretable patterns describing the data. Prediction involves using some variables or fields in the database to predict unknown or future values of other variables of interest.

1.1 FREQUENT PATTERN MINING (FPM)

An active research area in data mining is the efficient discovery of frequent patterns from the large databases. Frequent pattern mining is a core research topic in data mining for many past years. Frequent pattern mining was first proposed by Agrawal et. al. in the year 1993 for Market Basket Analysis in the form of association rule mining. Frequent itemsets play an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems.[1,2] The original motivation for searching association rules came from the need to analyze so called supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. Frequent patterns are itemsets or substructures that exist in a data set with frequency no less than a user specified threshold.

The main objective of FPM is to find the frequently occurring items in a large database. Frequent Patterns are itemsets, subsequences or substructures appearing in a dataset with frequency. They can be classified as a) Candidate generation algorithms b) Pattern growth algorithms. Of particularly no order, this could be categorized into several forms and they are as follows; a) Data Structures b)Traversal Strategy i.e Breadth First Strategy (BFS) or Depth first strategy(DFS) [21].Mining frequent patterns is to discover the groups of items appearing always together in excess of a user specified threshold. A transaction database consists of a set of transactions. A transaction is a set of items purchased by a customer at the same time. A transaction t contains an itemset X if every item in X is in t. A transaction database is a collection of sets of items (transactions). A set of items is called an itemset. The number of items in an itemset is called the length of an itemset. An itemset of length k is called a k-itemset and a frequent itemset of length k a frequent k-itemset [6,12]. A frequent pattern or a frequent itemset is an itemset whose support is no less than a certain user-specified minimum support threshold.

The following table 1 supports evidence for generating frequent itemsets ; it is necessary to calculate from the following formula such as $2^n -1$

Table 1: Generation of Frequent Pattern Mining

Items	Frequent Patterns	Description
5	$(2^5 -1) = 31$	If there are only 5 items , 31 frequent patterns are generated
10	1023	If there are only 10 items , 1023 frequent patterns are generated
100	$1.2676 * 10^{30}$	If there are only 100 items , $1.2676 * 10^{30}$ frequent patterns are generated

1.2 ASSOCIATION RULE MINING (ARM)

Association rule mining searches for interesting relationships among items in a given dataset. The discovery of interesting association relationships among huge amounts of business transaction records can help in many business decision making processes such as catalog design, cross-marketing and loss-leader analysis. ARM must be emphasized to find out the association rules that satisfy the predefined minimum support and confidence from a given database. The support for an itemset is defined as the ratio of the total number of transactions which contain the itemset to the total number of transactions in the database [5,8,23]. The support count for an itemset is the total number of transactions which contain the itemset.

Support and confidence are two key measures for association rule mining.

$$\text{Support } (A{=>}B) = P\ (AUB)$$
$$\text{Confidence } (A{=>}B) = P\ (B/A)$$

The typical example of association rule mining is Market Basket Analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”.

1.2.1 TASKS OF ASSOCIATION RULE MINING

The task of ARM is to find all strong association rules that satisfy a minimum support threshold (*min sup*) and a minimum confidence threshold (*min conf*).

Mining association rules consists of two phases.

- *In the first phase, all frequent itemsets that satisfy the minimum support are found.*
- *In the second phase, strong association rules are generated from the frequent itemsets found in the first phase.*

The performance for large databases is most influenced by the combinatorial explosion of the number of possible frequent itemsets that must be considered and also by the number of database scans that have to be performed.

The number of possible association rule mining (ARs), given a number of products or items d, is too large, # rules $AR(d) = 3^d - 2^{d+1} +1$. The table 2 gives supporting evidence for the rules of mining sets, for instance, the random variables have been chosen as an example;

Table 2: Rules for Association Rule Mining

# Items	# Rules	Explanation
5	180 ($3^5 – 2^{5+1} +1$)	*If there are only 5 items in the transaction database, it needs 180 rules to be generated.*
10	57002	*If there are only 10 items in the transaction database, it needs 57002 rules to be generated.*
100	5. 1537752 * 10^{47}	*If there are only 100 items in the transaction database, it needs 5. 1537752 * 10^{47} rules to be generated*

When the number of items increases, the rules of association mining also increase sharply. Current databases pose the challenges to store the large amount of data, proliferation, ubiquity and increasing power of computer technology, large database size or datasets, and their complexities, but there is a limitation for storing capacity. Furthermore, there is always hype when a promising new technology appears. Therefore, the new technology causes certain implications in drilling the unwanted data in order to store the necessary data. It is necessary to filter them in some way before trying to analyze their usefulness.

2. LITERATURE SURVEY ON FREQUENT PATTERN MINING ALGORITHMS

Any investigator should be aware of the already existing works. It helps him to identify the key issues in the current state of the knowledge. A good literature is traditional and original at the same time. The English critic, T.S. Eliot putsforth his view that "The historical sense involves a perception not only of the pastness of the past, but of its present"[9]. It helps the researcher to gain the background knowledge of the research topic, to identify the concepts relating to it, to identify the relevant methodology and to learn and identify the data sources and their structure of the reports. Many literatures have been dedicated to this research field and tremendous progress has been made in this direction. With over a decade of substantial and fruitful research, it is time to perform an overview of this flourishing field and examine what more to be done in order to turn this technology a cornerstone approach in data mining applications [18] .Enormous existing algorithms have been developed in regard to scalability, time and space requirement for handling massive collection of databases. Numerous algorithms focus their attention on performance i.e. Runtime execution and memory perceptions.

The brief history of the research algorithms of the Frequent Pattern Mining in Horizontal and Vertical Data Layouts has been discussed in this section. The following table 3 provides the key information on the literature survey.

Table -3. Comparative Analysis of the different existing algorithms

S.No.	First Author's Name	Algorithms	Data Structure/ Layout	Search Direction	Year of Publications
1	Agrawal	APRIORI	Hash tree (Horizontal)	BFS	1993
2	Shenoy	VIPER	Vertical	BFS	2000
3	Zaki	ECLAT	Vertical	DFS	2000
4	Han	FP-Growth	Prefix tree (Horizontal)	DFS	2000
5	Burdick	MAFIA	Vertical	BFS	2001
6	YabuXu	PP-Mine	Prefix tree (Horizontal)	DFS	2002
7	El-Hajj	COFI	Prefix tree (Horizontal)	DFS	2003
8	Zaki	DIFFSET	Vertical	BFS	2003
9	Song	TM	Vertical	DFS	2006
10	Show-Jane Yen	TFP	Prefix tree (Horizontal)	Hybrid	2009
11	Show-Jane Yen	SSR	Horizontal	DFS	2012

The data structures used play an important role in the performance of FIM algorithms. The various data structures used by FIM algorithms can be categorized as either candidate generation or pattern growth method. The majority of classic algorithms are candidate generation, where candidate itemsets are constructed and then validated. Pattern growth techniques, however, eliminate the need for candidate generation by constructing complex hyper structures that contain representations of the itemsets within the dataset.

The very first algorithm was AIS (Agrawal, Imielinski, and Swami) algorithm. It was proposed to address the problem of association rule mining. This is a multi-pass algorithm in which candidate itemsets are generated while scanning the database by extending known-frequent itemsets with items from each transaction. The main problem of the AIS algorithm is that it generates too many candidates, there is no clear specification of data structures for maintaining frequent and candidate item sets , consumes more memory space and multi-passes over the whole database.

Agrawal and his colleagues modified the algorithm and renamed it as Apriori in which a prior knowledge about frequent itemset was used. The Apriori algorithm is one of the classical algorithms in the association rule mining. It uses simple steps to discover frequent itemsets. An interesting downward closure property, called Apriori, among frequent k itemsets: A k-itemset is frequent only if all of its sub-itemsets are frequent. This implies that frequent itemsets can be mined by first scanning the database to find the frequent 1-itemsets, then using the frequent 1-itemsets to generate candidate frequent 2-itemsets and check against the database to obtain the frequent 2-itemsets. This process iterates until no more frequent k- itemsets can be generated for some k. This is the essence of the Apriori algorithm .The prior knowledge is that if an itemset is not frequent, then all of its supersets can never be frequent [2]. The main characteristics of this algorithm are Iterative level wise search, Breadth –First search, downward closure property.

There are two bottlenecks of the Apriori algorithm such as (1) generating a huge number of candidate sets and (2) repeatedly multiple scanning of the database and checking the candidates by pattern matching. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements such as AprioriTid (1994) algorithm which uses an encoding scheme for calculating the support of candidate itemsets after the first pass. It saves much time and occupies minimal amount of space.

Apriori Hybrid (1994), SetM (Set Oriented Mining of association rules) (1993), DHP (Direct Hashing and Pruning, 2001) by Park, Partition algorithm, Sampling algorithm, CARMA (Continuous Association Rule Mining algorithm, 1995) by Hidber (compute large itemsets online), DIC (1997) algorithm (prefix tree datastructure)-these algorithms are the further improvement of Apriori algorithm and reduce the number of database scans. These algorithms swallow time for calculating the supports for a large number of candidate itemsets for every pass. Pincer-Search algorithm (1998) by Lin et.al reduces the number of scans by traversing through top down search as well as bottom-up manner at the same time. Max Miner (1998) by Bayardo is an efficient algorithm for pruning based on look aheads to quickly narrow the search for finding maximal elements. This algorithm uses set enumeration tree to discover all frequent itemsets and utilizes breadth-first traversal of the search space.

Shenoy (2000) proposed an algorithm called VIPER (Vertical Itemset Partitioning for Efficient Rule-extraction); this vertical mining algorithm stores the data in compressed bit-vectors called “snakes” and integrates a number of novel optimizations for efficient snake generation, DAG based on snake intersection, counting and storage which perform in VIPER. A multipass algorithm outperforms the horizontal mining algorithm called Apriori. VIPER uses the vertical tid-vector (VTV) format for representating an item’s occurrence in the tuples of the database [25].This algorithm also introduces techniques to minimize the number and size of Snakes required and also deletes all Snakes created in previous scans that are no longer required for future computation.Zaki introduced a different approach called as Eclat (2000) of intersection of transaction ids (TID’s) in vertical database representation for finding frequent patterns by a depth-first search [38]. Depth Project (2000) by Ramesh finds long itemsets, using a depth first search of a lexicographic tree of itemsets, and uses a counting method based on transaction projections of bitmaps to improve performance.

Han introduced an efficient algorithm called FP-Growth (2000 & 2004) which constructs a frequent pattern tree structure called FP-Tree [18, 19, 20]. FP-Tree frequent pattern mining is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones.

The efficiency of FP-Tree algorithm accounts for three reasons. First the FP-Tree is a compressed representation of the original database because only those frequent items are used to construct the tree; the other irrelevant information is pruned. Secondly this algorithm only scans the database twice. Thirdly, FP-Tree uses a divide and conquer method that considerably reduced the size of the subsequent conditional FP-Tree. The major drawbacks of FPgrowth algorithm [Sotiris Kotsiantis 2006] are as follows: Though FP-Trees have the reduction in size, the recursive constructing times of sub-trees could be more in number ; it results in not fitting the main memory. During the interactive mining process, users may change the threshold of support according to the rules[30]. However , for FP-Tree the change of support may lead to repetition of the whole mining process. Any incorporation of new datasets in the existing database leads to the repetition of whole process, and poses another limitation in incremental mining process. The main problem in FP-tree is that the construction of the frequent pattern tree is a time consuming activity. Further, FP-tree based approaches do not offer flexibility and reusability of computation during mining process.

PAPG [Primitive Association Pattern Generation, 2001] by Yen et. al. constructs an association graph and scans the database once for recording the related information. It traverses the association graph for generating the frequent itemsets. It takes a lot of execution time, memory space for performing intersections and records the related information. MAFIA (2001) by Burdick et. al. uses vertical bit-vectors for fast itemset counting .Mafia uses a number of pruning methodologies to remove non-maximal itemsets such as;

i) look-ahead pruning,
ii) to check if a new set is subsumed by an existing maximal set and,
iii) if $t(X)$ subset $t(Y)$.

This algorithm mines a superset of the MFI, and requires a post-pruning step to eliminate non-maximal patterns [4]. The most time-consuming step involves the conversion of database into vertical bit-vectors format. PP-Mine (Xu *et. al.*, 2002) finds all the frequent itemsets through a coded prefix-path (PP-tree) which has a node-link-free tree structure[33]. It constructs a large number of sub-header tables recursively. It takes a lot of time to search from the sub-header-tables when push-right and push-down operations occur PatriciaMine (Pietracaprina *et. al.*, 2003) employs a compressed PatriciaMine trie (rooted and labelled tree) to store the data sets. This algorithm is a modification of a regular trie in which maximal chains of nodes that have a common count (support), are coalesced into a single node that inherits the count and stores the items in the same sequence. A trie consists of maximal chains connected to a single edge where chain is the directed path of all inner nodes having only one child. It consumes less memory if the trie contains many chains. Otherwise, it needs more memory, because the labels are represented by vectors. To address the issues which are faced in FP-Growth algorithm, Grahne developed a FP-Growth* in the year 2003.It uses an additional array-based structure to reduce the number of tree traversals required during analysis. This array-based structure saves on general traversal times a FP-Trees.

Later on, Zaki (2003) introduced a novel vertical data representation called Diffset that only keeps track of differences in the tids of a candidate pattern from its generating frequent patterns. This algorithm drastically cuts down the size of memory required to store intermediate results and intersection operations can be performed faster. It outperforms better than Viper and Apriori algorithm [22]. DynGrowth (2003) algorithm proposed by Gyorodi with modification of the original structures of FP-Growth, replaces the single linked list with a doubly linked list for linking the tree nodes to the header and adding a master-table to the same header. COFI-Co-Occurrence Frequent Item Tree algorithm introduced by El-Hajj (2003) is based on the core idea of the FP-Growth. It constructs tree for each frequent item, and generates candidate itemsets and counts their supports from the sub-trees. It avoids recursively generating many sub-trees which are faced in FP-Growth [10] .

An efficient algorithm as PRICES (2004) was proposed by Chuan Wang for association rule mining. This algorithm scans the database only once and logical operations are used for that purpose. Chuan proved that this algorithm would perform much better than the traditional algorithm. A different approach for generating large frequent candidate items algorithm is called as Matrix algorithm by Yuan et al (2005). The algorithm generates a matrix and the value must be 1 or 0 by passing over the cruel database only once, and then the frequent candidate sets are obtained from the resulting matrix. Finally association rules are mined from the frequent candidate sets.

GenMax (2005) by Gouda and Zaki uses diffset, progressive focusing to perform maximality checking of itemsets .This algorithm is based on backward search for finding maximal frequent itemsets [13]. In the year 2006, DCI-Closed algorithm was proposed by Lucchese et. al. with the bitmap representation of the data set.

For reducing the redundant computations, the author used the previous computed intersections; the basic operations such as closures, support counts can be performed and duplicate detections are intersections of bitwise tid lists. It outperforms the rest of the algorithms such as Closet+ and FP-Close algorithms.

In the year 2006, Song *et al.* introduced Transaction Mapping(TM) algorithm which deals with a novel approach that maps and compresses the transaction id list of each itemset into an interval list using a transaction tree and counts the support of each itemset by intersecting these interval lists [29]. The frequent itemsets are found in a depth–first order along the lexicographic tree which employs vertical database representation. Nittaya (2007) devised a different approach for executing frequent pattern mining implementation in Haskell language with a functional paradigm in a conciseness manner. Calders proposed an algorithm called XMiner in the year 2007 with new measures for itemsets and association rules, to be used in incomplete databases. It is also used to frequent itemsets in databases with missing values [6]. H-Mine (2004) by Jian Pei explores a hyper-structure mining of frequent patterns. It uses array-based and trie-based data structures to deal with sparse and dense data set respectively [17].

Borgelt proposed an algorithm called Relim (Recursive Elimination) and SAM (Split and Merge) algorithm(2009) which computes a conditional database recursively and finally eliminates the split item from the original (conditional) database. Relim employs depth –first/divide and conquer scheme. SAM algorithm is an improved version of the Relim algorithm, both of which distinguish themselves from other algorithms for frequent item set mining by their simple processing scheme and data structure[5] .TFP (mining frequent patterns by Traversing Frequent Pattern tree) algorithm introduced by Yen 2009 constructs an FP-tree without a header table and item-links and applies merging techniques on the tree after generating all the frequent itemsets for a specific item.TFP can dramatically condense the kernel memory space and reduce the search space without losing any frequent patterns. The drawback of TFP is time-consuming for sub-tree merging and needs to search for all the children of this merged node to find out which children need to be merged [34].

Ke-Chung Lin proposed IFP-growth (improved FP-growth) algorithm to improve the performance of FP-growth in the year 2011. There are three major features of IFP-growth. First, it employs an address-table structure to lower the complexity of forming the entire FP-tree. Second, it uses a new structure called FP-tree+ to reduce the need for building conditional FP-trees recursively. Third, by using address-table and FP-tree+ , the proposed algorithm has less memory requirement and better performance in comparison with FP-tree based algorithms. He also proved his algorithm needs only little memory space during the mining process and also suitable for high performance applications.

Show-Jane Yen(2012) introduced an SSR algorithm which combines the advantages of FP-Growth and Apriori algorithm .It generates a small set of candidates in batch from the sub-tree and results can be presented with the comparative analysis of search time and storage space. She proved her algorithm reduced the search time and storage space in an efficient manner when compared with the existing algorithms PPMine, COFI and TFP algorithms [36, 37]. Worth to mention here, the two bottlenecks in the SSR algorithm are as follows:

- A lot of time has been consumed for creating item-prefix pattern base
- A small set of candidates has been generated every time.

Hsiao-Wei Hu (2013) discussed a different approach of frequent pattern mining in Cloud-Based Environment. Cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. They discussed the key factors such as a) how to reduce CPU time (2) how to reduce data transmissions rate and (3) how to improve data privacy (security). They also suggested the way of reducing the fare of frequent pattern mining both cost and time in cloud environment [15].

Gwangbum Pyun proposed a new algorithm , LP-tree (Linear Prefix – Tree) ; it is composed of array forms and minimizes pointers between nodes in the year 2014.This algorithm requires minimum information required in mining process and linearly accesses corresponding nodes. This results in less usage of memory for building trees and needs less time for traverse in a linear structure[15]. He proved his experimental results that his approach outperforms previous algorithms in terms of the runtime, memory and scalability.

3. VISUALIZATION OF EXECUTION TIMES AND MEMORY USAGE OF THE EXISTING ALGORITHMS:

The following table 4 and Fig. 5 show a clear idea about the results of the different authors and their execution times for mining the frequent itemsets from large databases. This analysis gives a greater scope for the authors to find frequent itemsets in large databases.

3.1 PERFORMANCE EMPHASIS ON EXECUTION TIMES FOR THE DIFFERENT ALGORITHMS

It has pictorial representation of different algorithms for frequent pattern mining. It contains the detailed information about threshold value, average transaction size, execution times of the different algorithms and the year of publication of the algorithms.

Table 4: Execution times of different Horizontal Data Layout Algorithms

Algorithms	Average Transaction Size	Threshold	Execution Time (s)	Year
Apriori	10	1.5	5.3	1993
SETM	5	1	19	1993
Apriori TID	20	1.5	100	1994
Apriori Hybrid	10	0.75	7.5	1994
FPGROWTH	20	3	20.936	2000
PP-Mine	10	1.18	11.437	2002
COFI	20	3.11	12.563	2003
DynGrowth	30	5	8.23	2003
PRICES	10	5	150	2004
TFP	20	3	2.797	2006
SSR	10	1	1.766	2012

Figure1 depicts the graphical representation of the runtime execution of different algorithms in horizontal data layout. The X-axis represents algorithms and y-axis represents the execution times of the corresponding algorithms.

Figure 1: Performance Execution Graph for the different horizontal layout Algorithms with runtime.

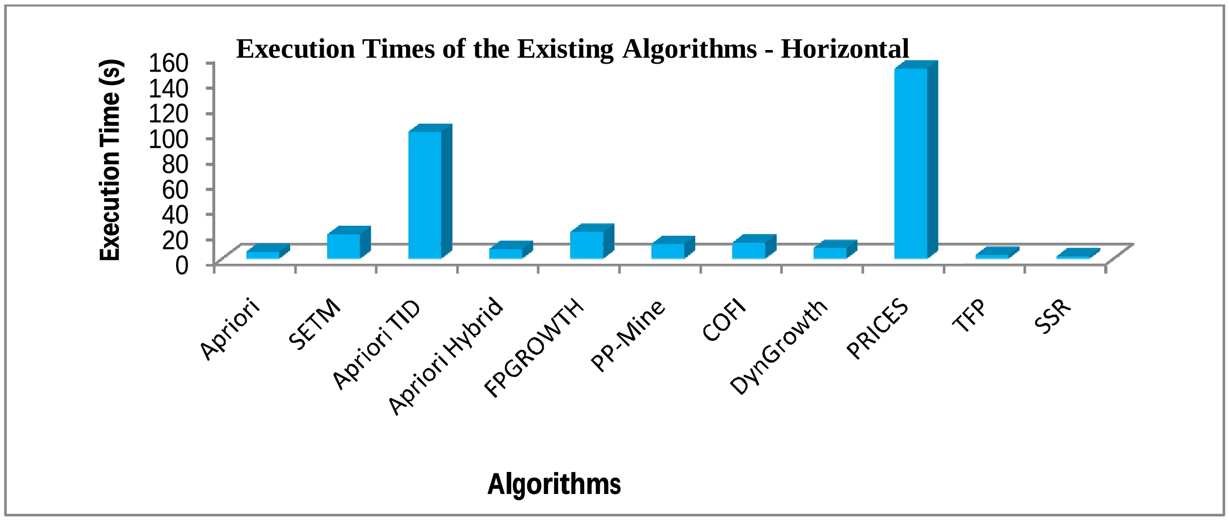


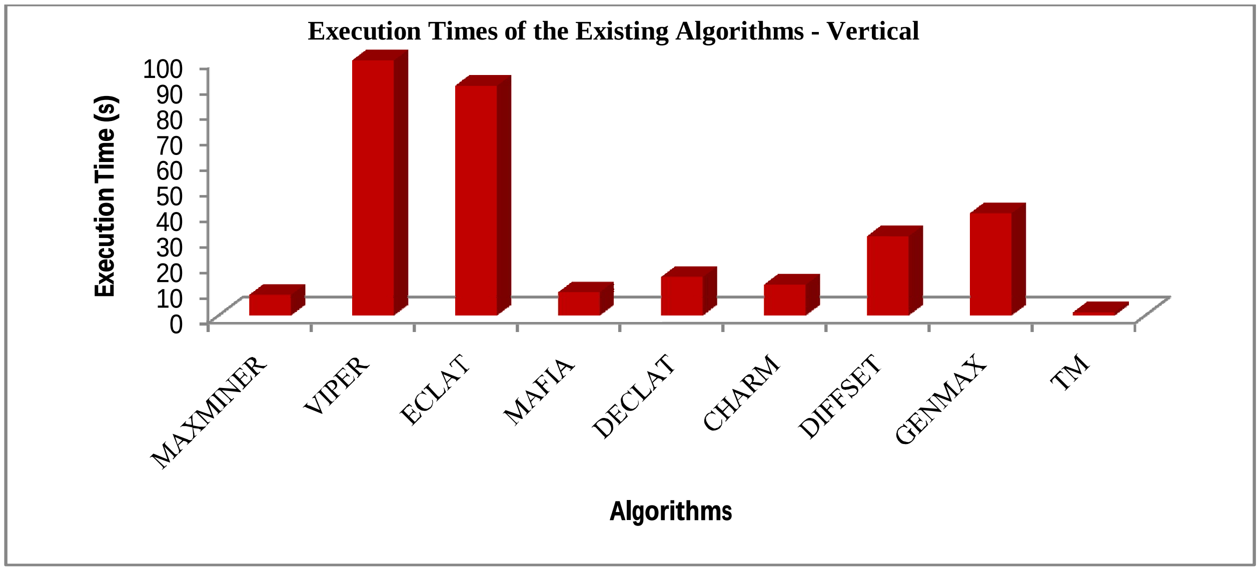
Table 5: Visualization of Execution times of the Existing Vertical Data Layout Algorithms- Performance Emphasis

Algorithms	Average Transaction Size	Threshold	Execution Time (%s)	Year
MAXMINER	30	1.2	8	1998
`VIPER	10	1.5	100	2000
ECLAT	40	1.4	90	2000
MAFIA	10	0.14	9	2001
DECLAT	40	1.4	15	2001
CHARM	30	1	12	2002
DIFFSET	20	0.1	31	2003
GENMAX	40	1.5	40	2005
TM	25	2	1.109	2006

Performance Execution Graph for the different Vertical Layout of Algorithms with runtime.

The Table 5 and Figure 2 show the execution times of the different algorithms in vertical data layout. The X-axis represents algorithms and Y-axis represents the memory usage of each algorithm.

Figure 2. Graphical Representation of Execution Times of the Existing Algorithms.



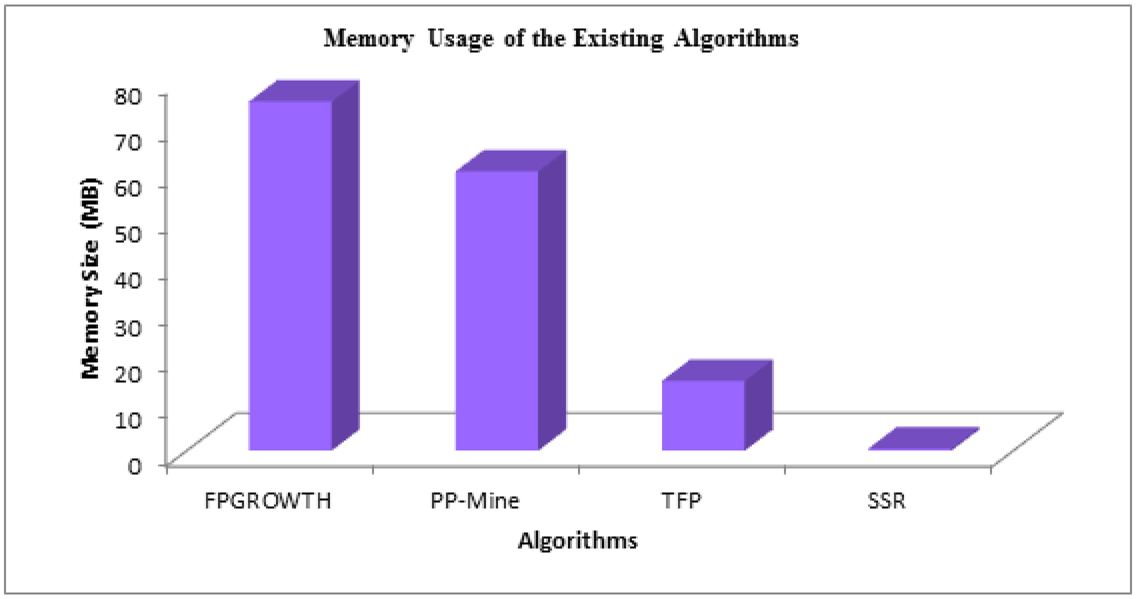
3.2 Memory Usage of the Existing Algorithms

The following table 6 and Fig. 3 show a clear idea about the results of the different authors and their memory usage of their algorithms for mining the frequent itemsets from large databases.

Table 6: Memory Usage of the Existing Algorithms

Algorithms	Average Transaction Size	Threshold	Memory Size (MB)	Year
FPGROWTH	20	3	75	2000
PP-Mine	10	3.11	60	2002
TFP	20	3	15	2006
SSR	10	1	0.5	2012

Figure 3 Visualization of Memory Space of Existing Algorithms for Vertical Layout.



CONCLUSION

Most of the previously proposed methods adopted Apriori like candidate-generation, frequent pattern-tree approach, test approaches, prolific and/or long patterns. The discovered patterns suffer from the serious challenges such as generalized share-prefix items, hypes/uncertainties in incorporating new algorithms, low support thresholds/large data sets leading to repeated scans, and a large number of frequent itemsets in recursive sub-trees work either in horizontal or vertical layouts. In this paper, a brief discussion about the existing algorithms of frequent pattern mining in both data layouts such as horizontal and vertical is attempted . Also a comparative analysis is given based on the performance of each algorithm and memory space occupied by the different algorithms is presented in a graphical manner.

REFERENCES

[1]. Agrawal.R and Srikant. R, 1994 ,"Fast Algorithms for Mining Association Rules", Proceedings of 20th International Conference on Very Large Databases, Santiago ,Chile ,San Mateo, USA, pp. 487-499.

[2]. Agrawal.R, T. Imielinski and Swami.A.N., 1993, "Mining Association Rules between Sets of Items in Large Databases," proceedings of 1993 ACM SIGMOD Intl. Conf. Management of Data, , San Jose, CA ,vol. 22 , pp. 207-216.

[3]. Agrawal.R and R. Srikant. Quest Synthetic Data Generator, IBM Almaden Research Center, SanJose, California, http://www.almaden.ibm.com/cs/syndata.html.

[4]. Burdick D, Calimlim M, Gehrke J ,2001 , MAFIA: a maximal frequent itemset algorithm for transactional databases, In Proceeding of the 2001 international conference on data engineering (ICDE'01), Heidelberg, Germany, pp 443–452.

[5]. Borgelt.C and X. Wang, 2009, SAM: A Split and Merge Algorithm for Fuzzy Frequent Item Set Mining, Proceeding 13th International Fuzzy Systems Association World Congress and 6th Conference of the European Society for Fuzzy Logic and Technology, IFSA/EUSFLAT Organization Committee, Lisbon,Portugal.

[6]. Calders, T. and Goethals .B, 2005, Depth-first non-derivable itemset mining. Proc. SIAM International Conference of Data Mining, vol. 119, pp. 250-261.

[7]. Chen et. al., 2001, Data Mining and Uncertain Reasoning: Integrated Approach, Wiley-Interscience Publications.

[8]. Chin-Feng Lee et. al. 2006, A data mining approach to database compression, Information System Frontier , Springer Science and Business Media, LLC.

[9]. Eliot.T.S., 1972, "The Sacred Wood: Essays on poetry and Criticism ", Methuen and Co. Ltd. London.

[10]. El-Hajj.M. and Zaiane.O.R ., 2004,COFI approach for mining frequent itemsets revisited, proceedings of ACM SIGMOD Workshop Res. Issues Data Mining Knowledge Discovery, New York, pp. 70–75.

[11]. Ellis Horowitz et. al. 1996, Computer Algorithms Galgotia Publications.

[12]. Goethals .B, 2003, "Survey on Frequent Pattern Mining" manuscript.

[13]. Gwangbum Pyun, 2014, Efficient frequent pattern mining based on Linear Prefix tree Knowledge-Based Systems vol. 55 , pp. 125–139.

[14]. Karam Gouda and Mohammed J. Zaki, 2005. "Gen Max: An Efficient Algorithm for Mining Maximal Frequent Itemsets, Data Mining and Knowledge Discovery, Vol. 11, No. 3, pp. 223-242.

[15]. Ke-Chung Lin et. al., 2011, An improved frequent pattern growth method for mining association rules, Expert Systems with Applications vol. 38 ,pp. 5154–5161.

[16]. Han, J. and Pei, J, 2000, Mining frequent patterns by pattern-growth: methodology and implications. ACM SIGKDD Explorations Newsletter2, pp. 14-20.

[17]. Hsiao-Wei Hu et. al. 2013, An Approach of Frequent Pattern Mining In Cloud-Based Environment, Business and Information.

[18]. Jaiwei Han et. al.,2001, Data Mining: concepts and Techniques, Morgan Kaugmann publishers.

[19]. Jiawei Han, Jianpei, Yiwen Yin, Runying Mao, 2004 , "Mining frequent patterns without candidate Generation: A frequent pattern tree Approach" Data Mining and Knowledge Discovery, Vol. 8, Issue 1, pp. 53-57.

[20]. Jaiwei Han et. al., 2007, Frequent pattern mining: current status and future directions, Data Mining Knowledge Discovery.

[21]. Kalpana. B, Nadarajan. R.,2007, Optimizing Search Space Pruning in Frequent Itemset Mining with Hybrid Traversal strategies - A comparative performance on different data organizations, IAENG.

[22]. Mohammed J. Zaki and Karam Gouda, 2003, Fast Vertical Mining using Diffsets, In proceedings of the ninth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining, Washington, D.C,pp. 326-335.

[23]. Mannila, H., H. Toivonen and A. Inkeri Verkamo, 1994, Efficient algorithms for discovering association rules, Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, (KDD-94), IEEE, pp: 181-192.

[24]. Nittaya Kerdprasop and Kittisak kerdprasop, 2007, Mining frequent patterns with functional programming, International Journal of computer and Information science and Engineering.

[25]. PradeepShenoyet. al., 2000 ,Turbo-Charging Vertical Mining of large databases, IISC, Technical report, DSL.

[26]. Piatetsky-Shapiro, G., 1991, Notes of AAAI`91 Workshop Knowledge Discovery in Databases (KDD`91). AAAI/MIT Press, Anaheim, CA.

[27]. Pujari.K. , 2001 ,Data Mining Techniques ,Universities Press.

[28]. Sarawagi. S, S. Thomas and R. Agrawal, 1998, Integrating Association Rule Mining with Databases: Alternatives and Implications, ACM SIGMOD International Conference of Management of Data..

[29]. Song. M and Rajasekaran , 2006 , A Transaction Mapping Algorithm for Frequent Itemsets Mining, IEEE Transactions on knowledge and Data Engineering, vol. 18, No. 4.

[30]. Sotiris Kotsiantis et. al., 2006, Association Rules Mining: A Recent Overview, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), pp. 71-82.

[31]. Tarek.et.al., 2007, "A New Paradigm for the computational complexity Analysis of Algorithms and Functions",International journal of applied mathematics and informatics, volume 1,issue 1.

[32]. Tiwari, A et. al., 2010, A survey on frequent pattern mining: Current status and challenging issues. Information Technology Journal.

[33]. Toivonen , 2010 , Frequent Pattern, Encyclopedia of Machine Learning, Chapter No: 00403.

[34]. Xu.Y, et. al., 2002, From path tree to frequent patterns: A framework for mining frequent patterns, in Proceedings of IEEE International Conference on Data Mining pp.514-521.

[35]. Yen. S. J et. al, 2009 ,The studies of mining frequent patterns based on frequent pattern tree, in proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Lecture Notes of Artificial Intelligence vol. 5476 , pp.232-241.

[36]. Yen.S.J. et. al,, 2012, A Search Space Reduced Algorithm for Mining Frequent Patterns, Journal of Information Science and Engineering , pp.177-191.

[37]. Zaki.M.J. 2000, Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering, vol. 12(3), pp. 372-390.

[38]. Zaki.M.J.,2002 , Efficiently Mining Frequent Trees in a Forest, ACM SIGKDD International conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp.71-80.