Frequent Itemset Mining

MANAN MAYUR POPAT, 2020A7PS0029H, BITS Pilani Hyderabad Campus, India

ABSTRACT

Data Mining is a field of data science specializing in extracting particular data patterns from a large dataset. Association rule mining is a rule-based technique used for uncovering relations between variables. In the truest sense, association rules work based on two concepts: usefulness and certainty, as it typically needs to satisfy the minimum user-specified usefulness and certainty. An association rule generation typically involves the application of a minimum support threshold to look for all the frequent items; if the minimum threshold criteria are satisfied, a rule is said to be composed. This technique is used in a variety of applications. One of them is Market basket analysis, which will be rigorously examined in this survey. Moreover, association rules in medical diagnosis can help physicians cure patients. Using relational association rule mining, we can identify the probability of the occurrence of an illness by monitoring various factors and symptoms. When applied to datasets of census information, efficient plans for public services and businesses can be made, supporting sound public policy and efficiently functioning democratic society. Knowledge and understanding of association rules are also necessary for the synthesis of artificial proteins.

Market basket analysis is a data mining technique to find the associations between the items in a store, using which we can analyze consumer behavior. It is based on the simple idea of how likely we are to buy or avoid other items if we have already purchased an itemset. It helps us examine the customer purchasing behavior and helps in boosting the sales and conserving inventory by analyzing historical sale transaction data. Market basket analysis increases customer satisfaction and sales. After determining which products are often purchased together, retailers can optimize product placement, offer special deals, and create new product bundles to encourage further sales. Also, decisions like which items to stock more, cross-selling, up-selling, and catalog design are determined.

The main goal of any algorithm designed for Frequent Itemset Mining (FIM) is to determine the set of items that are supposed to be associated with each other, i.e., frequently occurring together. This is done for the company selling these products to fine-tune the prices and cater better to the customers. These algorithms generally try to find associative patterns between a group of products or predict the likelihood of a given item to be bought along with the already purchased itemset. Some of the most commonly used algorithms are:

- **Apriori Algorithm:** It works on the principle that a subset of a frequently-bought-together set is also a viable frequently-bought-together set. This algorithm uses metrics like Support, Confidence, and Lift for interpreting the association between the items.
- **Frequent Pattern-Growth Algorithm:** This algorithm uses Frequent Pattern trees to represent the transaction database. FP-trees have null as their root, and items are inserted recursively as children in such a way that they capture the association between the items.
- **Other Algorithms:** EXTRACT, LP-Growth, P-Mine, TreeProjection, ECLAT, Can-Mining, PrePost, and several other modified and improved versions of the proposed algorithms.

Every data mining algorithm has its pros and cons. In this paper, we would analyze the strengths and weaknesses of many of these techniques. For example, the Apriori algorithm faces the issue of high computation time while working with huge datasets. It generates numerous uninteresting itemsets leading to many rules which are practically of no use. However, there have been many

Author's address: Manan Mayur Popat, 2020A7PS0029H, BITS Pilani Hyderabad Campus, India.

Manuscript submitted to ACM

improvements and modifications, such as the fast Apriori algorithm, which when combined with fuzzy logic, helps in better selection of association rules with increased accuracy. The survey also presents some aspects that future researchers can focus on, such as evaluating the association rules generated on changing the lift and confidence values and calculating the standard deviation. Research directions and the challenges that lay ahead are described henceforth, followed by a brief summary of the paper.

Additional Key Words and Phrases: Apriori, FP-Growth, association rule mining, market basket analysis, support, confidence, database, pattern, candidate itemset, trees, transaction

ACM Reference Format:
Manan Mayur Popat. 2022. Frequent Itemset Mining. 1, 1 (June 2022), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays, there is evidently an exponential rise in information and an increase in data. Data centers are erupting with details as the number of rows increases and the size of databases grows day-by-day [1]. The field of Data Mining comes to the rescue by combining traditional data analysis with sophisticated algorithms for processing such large amounts of data. Data mining may be defined as the process of discovering and extracting predictive and insightful information from massive datasets [2]. It is used to analyze features and trends from vast quantities of data. It is an interdisciplinary field that merges concepts from machine learning, statistics, information theory, database systems, and pattern recognition. Almost all databases contain a significant amount of implicit information, and many interesting association relationships may be obtained among sets of objects, which in turn reveal useful patterns for decision support, marketing policies, medical diagnosis, financial forecast, and many other applications [3]. Association rule mining plays an integral role in data mining, as its algorithm performance directly affects the efficiency, integrity, and effectiveness of the observations and results obtained by any data mining project [4]. Mining association rules often requires iterative scanning of large databases, which is costly in processing.

Frequent itemset mining (FIM) constitutes an essential part of association mining. In fact, it is the most crucial and expensive task for the industry today [5]. Association rules and patterns are derived from frequent itemsets satisfying a minimum threshold in a dataset. The second phase of association rule mining involves the production of strong association rules from the frequent itemsets; these rules must assure the minimum support and minimum confidence criteria. Many algorithms have been proposed for the same, and considerable improvements and modifications have been made in terms of their performance. The subject has been extensively researched due to its abundant applications in a range of data mining tasks, including clustering, classification, and outlier analysis [6].

Frequent itemsets are mined based on a minimum support count given by the user. Sometimes, these frequent itemsets may also contain some relations and interesting patterns [7]. However, applying FIM to large databases is often problematic. Primarily, vast databases may not fit into the main memory. This can be resolved using levelwise breadth-first search-based algorithms [8], such as the well-known Apriori algorithm. Here, the dataset is read repeatedly for every possible size of candidate itemsets for frequency counting. Unfortunately, the memory requirement for handling the complete set of candidate itemsets grows exponentially, rendering the algorithm inefficient to use on single machines. Moreover, keeping the output and runtime under control (by increasing the minimum frequency threshold) and thereby decreasing the number of candidate and frequent itemsets is highly recommended. But there have been conflicts regarding this. Some studies have shown that itemsets with lower frequencies are more interesting [9].

Formally, let I = $\{i_1, i_2, \ldots..i_n\}$ be a set of n binary attributes called items [10]. Let the set of transactions (database) be denoted by D = $\{t_1, t_2, \ldots..t_n\}$. A unique transaction ID is assigned to each transaction in D which contains a subset of

the items in I. A rule is defined as an implication of the form $X \longrightarrow Y$ where $X, Y \subseteq I$, and $X \cap Y = 0$. The sets of itemsets X and Y are called antecedent and consequent of the rule, respectively.

This paper aims to present a brief history of the research in the field of FIM, to review the development and working of algorithms, along with a comparison of the same, followed by a case study on its application in Market basket analysis, and future scope. Recently released papers on the major FIM algorithms were collected and critically analyzed during the work, and have been attempted to summarize here for the beginners in the field as well as for scholars to gain a better understanding and overview of the same.

2 TIMELINE OF FIM THROUGH 25 YEARS

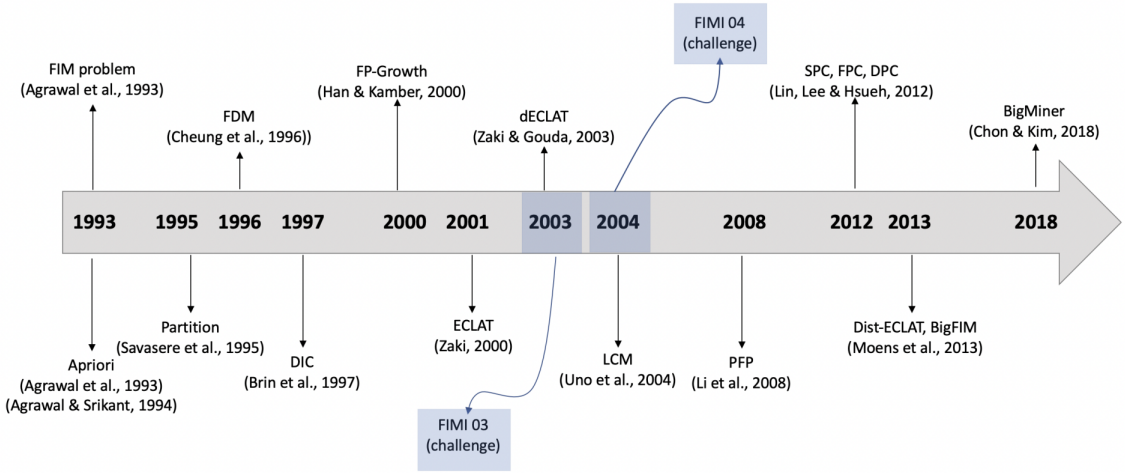

Fig. 1. Source: Reference [12]

It has been nearly 30 years since the first-ever algorithm of FIM was proposed, and now huge datasets can be analyzed within a few seconds. This was not just a matter of novel architectures and hardware progress but was also a consequence of the proposed algorithmic solutions [11]. Understanding how and why a recent algorithm reuses or adapts some algorithmic concepts of previous approaches may lead to new ideas flowing.

The very first approach was described at the beginning of the nineties, known as the Apriori algorithm [12]. In this approach, a levelwise breadth-first search (BFS) methodology helps produce candidate itemsets and multiple readings of the dataset (one for each size of itemset) assists in frequency counting. On the contrary, the algorithms proposed later (such as the FP-Growth algorithm) were majorly based on the depth-first search (DFS) procedure and considered either a compressed representation of the input dataset with main memory based on a prefix tree or utilized a vertical transposition of the dataset to work on transaction identifiers [13]. Moreover, due to the need to analyze large volumes of information, traditional FIM approaches have laid the foundations for novel methodologies based on parallel programming architectures considering both shared and distributed (non-shared) memory [8].

In 1996, the idea of mining patterns in a distributed environment was proposed [14]. In this approach, the concept of local and global patterns was also considered to offer a vast number of pruning strategies. Then in 2001, the ECLAT

(Equivalent CLAss Transformation) algorithm was proposed, which associated each pattern with the list of transactions in which it occurred [15]. Thus, the algorithm works well in obtaining the frequency of two patterns by simply intersecting their lists, but when a very high number of frequent patterns need to be intersected, its efficiency decreases. Immediately afterward, a similar algorithm based on the 'diffset' data structure (dECLAT) was proposed, which only keeps track of differences in the transactions of a candidate pattern from its generated frequent patterns [16].

Later research papers focussed on addressing the FIM problem using a heuristic search [17] instead of an exhaustive one as traditional approaches in this field do. Several sequential, multithread, and distributed computing solutions were also developed. After the transformation of all the data analytics subfields by the introduction of the term "Big Data" in 2005, the MapReduce programming framework was introduced to the industry enabling it to process data in parallel as small chunks in the form of distributed clusters [18]. The FP-Growth was modified using this framework to work on distributed machines, which came to be known as the Parallel FP algorithm (PFP) [19]. Some initial adaptations of the Apriori algorithm (SPC, FPC, DPC) were also designed [20] later. Then came the major breakthroughs in 2008 when two truly efficient MapReduce algorithms for FIM were proposed, namely DistECLAT and BigFIM [8]. The former was an improvement of the well-known ECLAT algorithm, and the latter combined principles from both Apriori and ECLAT.

Recent methods proposed in MapReduce for FIM seek to alleviate problems such as workload skewness, reducing the amount of intermediate data, and network communication overhead. Nevertheless, all these new approaches follow the same distributed methodology provided in 1995, in which the dataset is split into subsets for subsequent analysis [21].

3 RELATED WORK

3.1 Apriori Algorithm

The Apriori algorithm was proposed by R. Agrawal and R. Srikant in 1994 and is used to mine frequent itemsets for generating Boolean association rules. In principle, it is simple to understand and easy to execute [22]. Many searches are run on the database over all the candidate itemsets to find frequent itemsets by using frequent k-itemsets (satisfying minimum support threshold) for generating k+1-itemsets. Firstly, the frequency of each item in the database is checked to find frequent 1-itemsets, which are then used to capture frequent 2-itemsets, that in turn are used to generate frequent 3-itemsets, and so on until there are not any more k-itemsets. The algorithm utilizes the anti-monotonicity property that if an itemset is infrequent, then any subset of it is also infrequent. This helps in pruning itemsets from the search space in the database.

Formally, the core algorithm consists of two steps: the connecting step and the pruning step [23].

(1) **Connecting Step:** For identifying a frequent k-itemset L_k (frequent candidates are those whose count is not less than the minimum support count), a candidate k-itemset C_k is to be generated by L_{k-1} and its connections (the elements of L_{k-1} that can be connected).
(2) **Pruning Step:** The generated candidate k-itemset C_k is a superset of L_k (i.e., all the frequent k-itemsets are definitely included in C_k) whose members may be frequent or infrequent. While scanning the database, the frequency of each candidate in C_k as well as L_k can be determined. However, C_k may be large, thereby increasing computational complexity. For compression of C_k, the anti-monotonicity theorem comes to the rescue: any non-frequent (k-1)-itemset cannot be a subset of a frequent k-itemset. Therefore, if the (k-1)-itemset of a candidate k-itemset is not in L_k, then the candidate cannot be frequent; thus, it can be deleted from C_k.

Many improvements have been proposed since the core algorithm was developed. One such version [22] seeks to reduce the time consumed in scanning for candidate itemset generation. All transactions are scanned to generate L_1

items, as well as store their support count and transaction ID where the items are found. This is followed by using L_1 to generate L_2, L_3,..., L_k. For generating C_2, a self-join $L_1{}^*L_1$ is made. The transaction IDs and support count are maintained at each step and kept in mind while scanning for the next candidate itemset. The process is repeated until no unique frequent itemsets can be identified.

This version works well in terms of the time consumed (reduces by about 67%) to generate candidate support count when k of the k-itemset is very large.

Another better method is the incremental update of association rules, i.e., updating the derived association rules according to the new database [23]. This technique successfully encounters three issues: (i) how to generate the database from the given support and confidence thresholds, (ii) finding the updated database associations or link rules when either new datasets are added, or older ones are deleted from the original database, and (iii) given a database, how to generate the association rules when minimum support and confidence thresholds change. Based on this approach, two major algorithms have been suggested:

- **FUP (Fast UPdate) Algorithm:** This algorithm, put forward by D W Cheung, is one of the most typical, effective and practical improvements of the Apriori algorithm. It can effectively deal with obtaining the frequent itemsets in case of data increment. It basically uses mining results (i.e., the frequent items) of the original database and the multiple scans done over it to get the frequent items of the new database, thereby improving the efficiency. However, while renewing the database, the original database still needs to be repeatedly scanned for matching the candidates; thus, the time efficiency may often remain more or less the same.
- **Incremental Update Algorithm (IUA):** This algorithm uses a unique candidate frequent itemset to generate smaller frequent items before scanning the database each time, thereby enhancing the efficiency of the algorithm. It also utilizes the frequent items generated in previous scans to effectively adjust the minimum support and minimum confidence for gathering interest in the association rules. Thus, by using an effective candidate frequent item generation method and adopting a strategy for time against space, the necessary updation is successfully carried out. However, the algorithm suffers from the same limitations as the Apriori: multiple scans are required, and a large number of candidates are needed.

3.2 FP-Growth Algorithm

The FP-Growth algorithm is currently one of the fastest and most popular approaches to FIM. The algorithm revolves around a prefix tree representation (called an FP-tree) of the given database of transactions [24], and follows a divide-and-conquer strategy or a recursive elimination scheme. Firstly, all infrequent individual items are deleted from the transactions in the preprocessing step. Then all the transactions containing the least frequent item (least frequent among those that are frequent) are selected, and this item is deleted. This is followed by recursing the procedure on the reduced (or "projected") database since the itemsets found in the recursion share the deleted item as a prefix. Then the processed item is removed in return from the database, and the process is started over (i.e., the second least frequent item is processed, and so on). The prefix tree is enhanced by the links between the branches in these processing steps. It is exploited to find the transactions containing a given item as quickly as possible and remove this item from the transactions after being processed.

The algorithm can be illustrated as follows. Let's consider the transaction database in the below table and take the support count as 2 [25]. We first scan the database to obtain a set of frequent items and their support count in decreasing order and write the resulting set in L. Thus, we obtain L=[C:4,D:3, E:3,A:2, B:2]. This is followed by building the FP-tree.

A root node with the tag "null" is created. Then the database is scanned for the second time. A branch is created for each transaction with items ordered by the sequence of L. For example, the five items C,D,E,A,B constitute the first transaction "001:A,B,C,D,E" and generate the first branch ⟨(C:1),(D:1),(E:1),(A:1),(B:1)⟩ of the FP-tree. Essentially, the branch has five nodes with C as the child of the root, followed by D linking to C, E linking to D, and so on. Similarly, the items C,E,B constitute the second transaction "002:B,C,E" and generate another branch with C linking to the root, E linking to C and B linking to E. This branch shares the prefix ⟨C⟩ with the existing path of transaction "001". Therefore, the count of node C is increased by 1, and two new nodes ⟨(E:1),(B:1)⟩ are created as a link of (C:2).

In this way, the algorithm adds a branch for each transaction, and the count for a node is increased by one and appropriately linked when it follows a common prefix. Moreover, an additional item header table can be created for the convenience of tree traversal, wherein each item through node-link points to itself in the FP-tree. Thus, once all transactions are scanned, we obtain the FP-tree as shown below.

Transaction database.

TID	Items
001	A, B, C, D, E
002	B, C, E
003	C, E, D
004	A, C, D

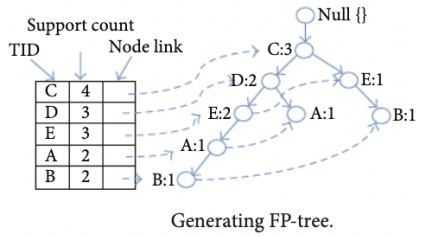

Fig. 2. Source: Reference [25]

The next step is mining the FP-tree. Using the initial suffix pattern, all the frequent patterns of length 1 are found. Then a conditional pattern base (a "sub-database," consisting of the prefix path set which appears with the suffix pattern) is built for each such pattern, followed by building a conditional FP-tree for the conditional pattern base. Hence, the algorithm recursively digs the tree through the links between frequent patterns thus obtained, and a pattern growth is achieved. The mining of FP-tree is summarized in the table shown below.

Dig FP-tree through creating conditional subpattern base.

Item	Conditional pattern base	Conditional FP-tree	Frequent pattern
B	{(C D E A:1), (C E:1)}	⟨C:2, E:2⟩	C B:2, E B:2, C E B:2
A	{(C D E:1), (C D:1)}	⟨C:2, D:2⟩	C A:2, D A:2, C D A:2
E	{(C D:2), (C:1)}	⟨C:2, D:2⟩	C E:2, D E:2, C D E:2
D	{(C:2)}	⟨C:2⟩	C D:2

Fig. 3. Source: Reference [25]

FP-Growth algorithm offers a wide number of advantages over the Apriori. Primarily, the database is scanned only twice for processing [26], and is thus faster. Moreover, the dataset is "compressed" during the passes [27], and no candidate itemset generation is involved. The only issue with the FP-Growth algorithm is that it requires the generation of a huge number of conditional FP-trees; thus, it is computationally expensive to build and may not even fit in memory often.

There exist many improved algorithms based on the FP-Growth algorithm. Two of the most famous ones are the Painting Growth Algorithm and the N-Painting Growth Algorithm [25], which scan the database only once to obtain the results. The former technique obtains two-item permutation sets of all transactions and 'paints' the set of all different items across all transactions (known as the "peak set"). This is followed by painting the association picture according to the two-item permutation sets and the peak set by using the appropriate links and maintaining the count. Then the support count is exploited to eliminate infrequent associations. This is applied recursively to obtain frequent itemsets of all possible lengths. On the contrary, the N-Painting Growth Algorithm removes the painting steps, and instead counts each permutation in two-item permutation sets to get all item association sets, followed by removing infrequent associations according to the support count. Hence, frequent item association sets are obtained and accordingly we get all frequent itemsets. While these improvements are definitely better in terms of simplicity and time efficiency, they lack behind the standard version in the increased memory overhead.

Other optimizations include the construction and usage of the D-trees [26] to reduce the number of conditional pattern bases generated and remove the generation of the conditional FP-trees. A D-tree is fundamentally the replica of the database under consideration, which is then used for further processing by scanning it (scanning the tree instead of the database) to obtain the frequency of each item. This is followed by constructing a "new improved FP-tree" and a node table and using the support count to get the frequent itemsets.

3.3 ECLAT Algorithm

The ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) algorithm was developed with the aim of removing the limitations of the Apriori algorithm. It uses a vertical database to facilitate for scanning of the database only once and finds the elements starting from bottom to top similar to the depth-first search method [28]. Then the supports of all the items are checked against the minimum support. This is followed by putting all those frequent items in a new database, and the bottom-up approach is again applied on the same.

The algorithm is also used in the MapReduce framework and is implemented in the Java programming language. The speed can be further increased by dividing the database into different processing units and combining the results of those units. The ECLAT algorithm is commonly used in network services. A common application is seen in online markets where the algorithm facilitates the customer in displaying the most frequently bought products, and the shopkeeper also gets to know the interest of people.

A major limitation of the algorithm lies in the large number of iterations required for processing the items and the large escape time needed for finding the frequent itemsets [29]. An advanced algorithm exists that instead uses the top-down approach and transposes the original database. In transpose operations, all the items are given to all the transactions at the same time. Thus, a new dataset is initialized to contain the information about the rows and columns of the transposed dataset. Then the support threshold is checked against all transactions, and the satisfying ones are taken. In this way, the processing time (to create the final association rules) and complexity of the algorithm are improved.

Yet another parallel ECLAT optimization algorithm BPEclat (Balanced Parallel ECLAT) is suggested to solve the performance shortcoming of the ECLAT algorithm in the serial processing of large-scale data [30]. It achieves the optimization by combining the depth pre-pruning and post-pruning strategies to reduce the computation of infrequent or irrelevant itemsets (thereby compressing the candidate set size) and using the range portioning idea to balance the node load. Moreover, prefix terms are used in order to divide the dataset, and the idea thus works well in improving the parallel computing power of the algorithm. Therefore, massive amounts of data can be processed more efficiently and reliably, and a considerable improvement in scalability and universality is achieved.

The algorithm is implemented using the Spark framework, one of the Hadoop ecosystem-based widely used distributed parallel computing frameworks. Spark effectively solves the defects in the MapReduce framework, namely excessive I/O load, poor fault tolerance, and non-conducive nature to iterative calculation due to the acyclic data flow model. The only drawback of the BPEclat algorithm implemented using Spark that remains is that load balancing is not considered while dividing different computing nodes, leading to skewed data and slightly reducing the performance of parallel computing.

3.4 TreeProjection Algorithm

The TreeProjection algorithm mines frequent itemsets by constructing a lexicographic tree through breadth-first search or depth-first search techniques (or often a mix of the two). For every frequent itemset in the transaction, the support is maintained and projected onto a lexicographic tree in the form of a node. This helps significantly in improving the performance in mining for a particular frequent itemset [31]. Moreover, nodes in this tree assume a lexicographic ordering among items in the database, and a node on depth (or level) k corresponds to an itemset of size k (while the root node has no itemset).

The projection techniques are based on prefix tree and perform the mining recursively on progressively smaller databases. This algorithm has been shown to outperform Apriori and many of its variant algorithms, being an order of magnitude faster in several scenarios. However, the projection-based algorithms have received relatively less attention than its peer candidate generation-based algorithms. Moreover, to the best of our knowledge, there does not exist a projection-based FIM algorithm for GPUs [32].

The advantage of visualizing itemset generation in terms of a lexicographic tree is that it provides us with the ability of picking the correct strategy during the tree generation phase as well as transaction projection phase [33]. By combining various strategies, a variety of algorithms are possible to provide very high performance in most situations.

3.5 EXTRACT Algorithm

The approach essentially mines frequent itemsets using the mathematical concept of Galois lattice [34]. The architecture of the algorithm is partitioned into four functions, one each for calculating the support count, combining the itemsets, eliminating the itemsets that are repeated, and extracting association rules from the frequent itemsets.

Firstly, all the single items are checked for satisfying the minimum support to generate the frequent 1-itemsets and remove the infrequent ones. Then the itemsets are combined in order to discover all possible combinations of frequent itemsets. This is followed by eliminating those combinations of frequent itemsets that are redundant. Once all the unique frequent itemsets are mined, the association rules which fulfill the minimum confidence will be generated; whereas those rules which do not satisfy the confidence threshold are removed from the rule discovery process.

In this way, the EXTRACT algorithm outperforms the Apriori in terms of speed and computational efficiency. However, since the frequent itemsets that have been mined are not stored in any disk or database, the algorithm must be executed again to mine the new set of frequent itemsets if there is a change in the dataset.

3.6 LP-Growth Algorithm

In this algorithm, frequent itemsets are mined using arrays in a linear structure known as Linear Prefix Growth tree (LP-tree). The tree comprises of arrays instead of pointers [35], thus the memory usage is reduced since the information of various connections between the nodes is decreased considerably. Each LP-tree is composed of multiple Linear Prefix Nodes (LPNs) in a linear fashion, which are further constituted of multiple arrays to store the frequent itemsets. In the case of the LPN being the first node to be inserted in the LP-tree, the root of the tree is indicated by the header of that LPN.

The LP-tree is generated much faster by the algorithm as compared to the FP-Growth algorithm since a series of array operations are used here to create multiple nodes simultaneously as opposed to creating nodes one at a time for generating FP-trees. Since the nodes are stored in the form of arrays, while searching through the LP-tree, any parent or child node can be accessed without using any pointers. Moreover, the LP-tree can be traversed quickly due to the fact that there is direct access to the corresponding memory locations in the array structure. In addition, since pointers are not used for linking up the nodes, the memory consumption reduces significantly. However, the process of inserting nodes is tedious in this approach, because the items from a transaction may be saved in various LPNs, and thus the memory needs to be freed continuously.

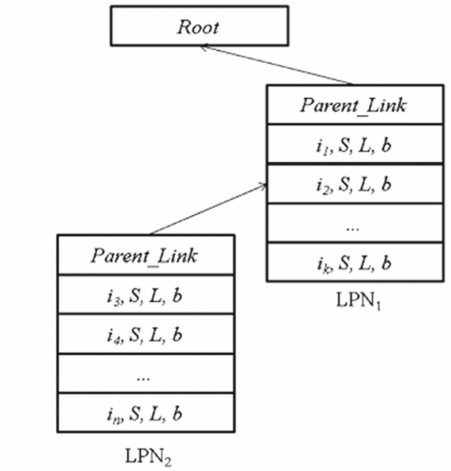

Fig. 4. Source: Reference [35]

3.7 P-Mine Algorithm

A parallel disk-based approach is adopted to mine frequent items on a multi-core processor [36]. Using a data structure known as the VLDBMine, the time required to produce a dense version of the dataset on disk decreases considerably. The data structure consists of a Hybrid-tree (HY-tree) to store the entire database and other information required to support the data retrieval process. A pre-fetching technique further enhances the efficiency of accessing the disk.

Multiple projections of the dataset are loaded into different processor cores for the purpose of mining the frequent itemsets. The results from each processor are eventually gathered and merged for constructing the entire frequent itemsets.

The overall mining is performed in two steps. Firstly, a persistent and lossy representation of the dataset items is stored in secondary memory using compact data structures. This is followed by mining on relevant portions of these data structures. Only reduced portions of data are selectively loaded into the main memory to be processed by the current mining task. Similar to database indices, the disk-based data representation is built once and then exploited for many different mining sessions.

The VLDBMine data structure essentially helps improve the scalability and performance of mining the frequent itemsets from the dataset, because the data can be selectively accessed using the HY-tree. Thus, it effectively supports the data-intensive loading process at a minimized cost. Moreover, the performance on each node is optimized locally during the mining process being executed across the various processor cores in parallel. However, this optimization can be achieved to the maximum extent only when multiple cores are available in the processor.

The proposed approach may easily lend itself to a distributed implementation, such as in a cloud computing environment. In particular, it may be seen as a building block to developing a SaaS (Software-as-a-Service) service offering a data analytics facility to cloud users.

3.8 PrePost Algorithm

This algorithm is also based on a vertical data representation called the N-list to store all the crucial information about frequent itemsets [37]. The idea originates from the FP-tree, and a similar coding prefix tree called the PPC-tree is used. The PrePost algorithm is based on this N-list data structure for efficient mining of frequent itemsets. The efficiency of the approach is achieved due to the following three reasons. Primarily, the data structure is compact because transactions share the same nodes of the PPC-tree with common prefixes. Secondly, support counting is performed by the intersection of N-lists whose complexity can be efficiently reduced to O(m+n), where m and n are the respective cardinalities of the two N-lists. Moreover, the algorithm enables direct finding of frequent itemsets without generating candidate itemsets by making use of the single path property of N-list.

The PrePost algorithm has been experimentally evaluated and proven to be faster than the ECLAT and FP-Growth algorithms, especially under the condition of low support counts. It also has an added advantage in terms of less memory consumption and better scalability. However, in the case of sparse datasets, the efficiency falls significantly.

3.9 Can-Mining Algorithm

Frequent itemsets are mined in an incremental manner from a Canonical-Order tree (Can-tree) [38]. The architecture of the algorithm is depicted below. A header table containing all the information about the database items, including the frequencies and pointers to the first and last nodes containing the items, is used in this algorithm similar to the FP-Growth algorithm. A list of frequent items is required in order to extract frequent patterns from the Can-tree and perform the mining operations. Since only frequent items are appended to the trees in a predefined order, the time of mining in the case of nested Can-trees is reduced significantly. This approach easily outperforms the FP-Growth algorithm when the minimum support threshold is set very high, while in the case of a low threshold, the efficiency of the FP-Growth algorithm is higher.

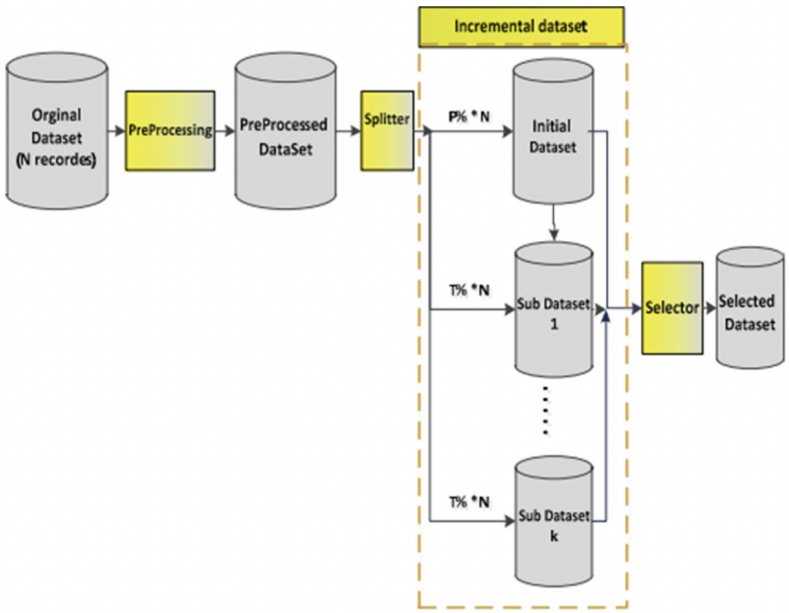

Fig. 5. Source: Reference [38]

3.10 TM Algorithm

Similar to the ECLAT algorithm, this algorithm also uses vertical data representation for mining frequent itemsets [39]. Here, the transaction ID of each itemset is transformed and mapped into a list of transaction intervals at another location. This is followed by performing an intersection between the transaction intervals using the depth-first search approach throughout the lexicographic tree hence obtained for counting the frequency of itemsets.

In the case of high minimum support, the transaction IDs are significantly compressed into continuous intervals using this mapping technique. The intersection time is also considerably reduced during the compression of itemsets into a list of transaction intervals. In terms of performance, the TM algorithm is proved to be better than the FP-Growth and dEclat algorithms, especially when the dataset contains short frequent patterns. However, the algorithm is still lower in terms of processing speed as compared to the latter algorithms.

4 MARKET BASKET ANALYSIS: A CASE STUDY

FIM has a wide variety of applications in marketing, bioinformatics, education field, nuclear science, etc.; the most common relevance being in MBA. The main aim of MBA in marketing is to provide the information to the retailer to understand the buyer's purchase behavior, which can help the retailer in correct decision making [40]. In order to understand the customer purchase patterns, knowing what products are often purchased simultaneously can be used and FIM techniques may be applied on them. Here we present a case study by Yusuf Kurnia et al. [41] using the Apriori algorithm to identify the sales pattern at the O! Fish Restaurant. Using the results obtained after this analysis, the restaurant managers were effectively able to create more potential promotional strategies to boost sales by referring to items (menus) that are often purchased together. Later the results of this study were deployed in the form of a

website-based application to analyze purchasing patterns (item association rules) by consumers, where the purchase patterns are used as recommendations in determining the promotion development strategy for similar restaurants.

Support is the percentage of item combinations that appear simultaneously in the dataset. Confidence is a percentage of the strength of relationships between items. TThe lift value is a measure of the importance of a rule. The following formulas obtain the support, confidence and lift measures:

$$Support(A) = \frac{Number\ of\ transactions\ in\ which\ A\ appears}{Total\ number\ of\ transactions}$$

$$Confidence(A \rightarrow B) = \frac{Number\ of\ transactions\ in\ which\ A\ and\ B\ appear}{Number\ of\ transactions\ in\ which\ A\ appears} = \frac{Support(A \cup B)}{Support(A)}$$

$$Lift(A \rightarrow B) = \frac{Confidence(A \rightarrow B)}{Support(B)} = \frac{Support(A \cup B)}{Support(A) * Support(B)}$$

The paper reviews the following methods for generating the right menu association rules:

- **Business Understanding:** This phase involves the determination of promotional strategies for boosting the sales and the procurement of raw products, which are generally difficult to predict.
- **Data Understanding:** The collection of initial data is done in this phase. That data will be used or processed later, and contains information of consumer purchase transactions within a given period of the restaurant operations. Using the Apriori algorithm, association rule analysis would be performed on this data, especially in terms of whether an item is sold or not, instead of the quantity sold.
- **Data Preparation:** The transaction data obtained is moved into an Excel file for facilitating data preprocessing, thereby also eliminating some attributes which are not used. This research analyzed 38 types of menus, 23 food menus, and 15 types of drinks to identify the frequent combinations often ordered by consumers.
- **Modeling:** This phase involves the study and conceptualization of the survey starting from the preparation of data to be used, the selection of mining techniques, to combining the parameters of the results obtained with optimal values.
- **Evaluation:** The sample data which has been processed and analyzed is begun to be prepared in this stage. An in-depth evaluation of the built model is carried out with the aim of adjusting the results at the modeling stage to suit the business objectives.
  (i) Evaluation Result - The association rules which can be used as new sources of information and meet the minimum support and confidence criteria are found out and declared as "valid" association rules.
  (ii) Review Process - The rules obtained are then tested against future transactions, and their utility (when implemented as promotional strategies) is examined.

The results of applying the Apriori algorithm over a database of 150 transactions are shown below. The analysis was performed on a data science software platform called 'RapidMiner', and the set minimum support value and confidence value were 4% and 60%, respectively. The figure depicts the association rules generated by the MBA.

The result of the association rule is stated as "if x, then y". For example, the third association rule is read as "O!fish Ice Coffee implies Cobia Pansear", i.e., if a customer buys the O!fish Ice Coffee, he/she is likely to buy Cobia Pansear with 4% support and 66.7% confidence. The retailers can use this knowledge to recognize the layout of a supermarket

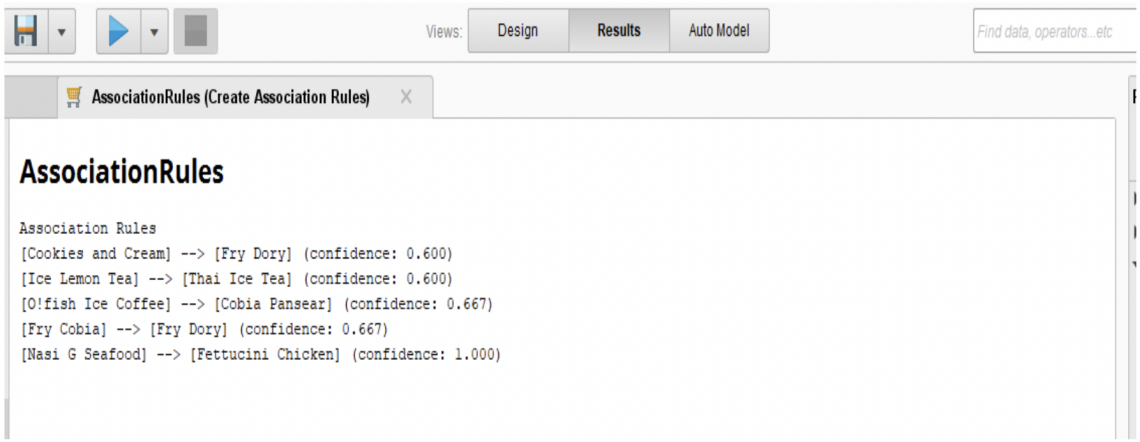

Fig. 6. Source: Reference [41]

[42], by taking frequently bought together products and locating them in close proximity. Moreover, the results can also be used to improve the effectiveness of a promotional campaign by keeping in mind that the products which are associated should not be put on promotion simultaneously. Instead, promoting just one of the associated products at a time helps increase the sales of that product and also gets an accompanying sales increase for the associated products.

Other methodologies such as market share analysis and time series analysis also exist which assist companies in making decisions for improving the production and marketing [43]. Yet another innovative approach has been proposed in [44] using the automatic word segmentation technology for obtaining the association rules on the items' internal characteristics. This method seeks to provide more scientific decision support for retail market and consumers' buying behavior at a lesser time cost than traditional analysis methods that cannot obtain enough association rules that policymakers are interested in.

5 COMPARISON OF FIM ALGORITHMS

In general, the algorithms for Frequent Itemset Mining (FIM) can be classified into three main categories, namely Join-Based, Tree-Based, and Pattern Growth algorithms. The Join-Based algorithms apply a bottom-up approach to identify frequent items in a dataset and expand them into larger itemsets as long as those itemsets satisfy a minimum threshold value defined by the user in the database. Tree-Based algorithms use set-enumeration concepts to solve the problem of frequent itemset generation by constructing a lexicographic tree that enables the items to be mined through a variety of ways like the breadth-first or depth-first search order. Finally, the Pattern Growth algorithms implement a divide-and-conquer method to partition and project databases depending on the presently identified frequent patterns and expand them into longer ones in the projected databases.

The two most commonly used algorithms, Apriori and FP-Growth, use the horizontal approach, and owing to the need of discovering all the candidate itemsets for each level, the number of candidates generated increases with the length of the frequent itemsets. The tree-projection method helps improve the efficiency in terms of speed, but it utilizes more space [45]. These drawbacks can be resolved by techniques such as hashing, partitioning, etc. Based on the same, multiple approaches have been proposed, and a comparative analysis is provided in the above table [46].

FPM algorithm	Advantages	Disadvantages
Apriori (Agrawal and Srikant 1994)	Uses an iterative level-wise search technique to discover (k+1)-itemsets from k-itemsets	Has to produce a lot of candidate sets if k-itemsets is more in numbers Has to scan the database repeatedly to determine the support count of the itemsets
FP-Growth (Han and Pei 2000)	Preserves the association information of all itemsets Shrinks the amount of data to be searched	Constructing the FP-Tree is time consuming if the data set is very large
EClaT (Zaki 2000)	Scanning the database to find the support count of (k+1)-itemsets is not required	More memory space and processing time are required for intersecting long TID sets
TreeProjection (Agarwal et al. 2001)	Identifies the frequent itemsets in a fast manner because only the subset of transactions that can probably hold the frequent itemsets is searched by the algorithm	Different representations of the lexicographic tree present different limitations in terms of efficiency for memory consumption
COFI (El-Hajj and Zaiane 2003)	Uses a pruning method to reduce the use of memory space significantly by constructing smaller COFI-Trees while mining for the frequent itemsets	The performance of the algorithm degrades in a sparse database if the threshold value of the minimum support is low
TM (Song and Rajasekaran 2006)	Compresses the itemsets into a list of transaction intervals in order to greatly save the intersection time for mining the frequent itemsets	Still slower in terms of processing speed compared to the FP-Growth* algorithm
P-Mine (Baralis et al. 2013)	Optimizes performance and scalability by executing the mining of frequent itemsets in parallel with multiple processor cores	The algorithm can only be optimized to the maximum level when multiple cores are available in the processor
LP-Growth (Pyun et al. 2014)	Generates the LP-Tree in a faster manner as a series of array operations are used to create multiple nodes together	Memory needs to be freed continuously as the items from a transaction may be saved in various LPNs
Can-Mining (Hoseini et al. 2015)	Outperforms the FP-Growth algorithm when the minimum support has a high threshold value	Mining time is longer if the threshold value of the minimum support is much lower
EXTRACT (Feddaoui et al. 2016)	Mines more than 300 objects and 10 attributes with an execution time that does not exceed 1200 ms	The algorithm needs to be executed again in order to mine the new set of frequent itemsets if there is a change in the data set

Fig. 7. Source: Compiled

Amongst the existing Pattern Growth algorithms, most of them are evolved from the FP-Growth algorithm. This is because FP-Growth generates all the frequent patterns using only two scans of the data set, representing the entire dataset with a compressed tree structure, and decreases the execution time by removing the need to generate the candidate itemsets [47]. Although the existing FIM algorithms are able to mine the frequent itemsets in a dataset by identifying the association between different data items, a lengthy processing time and large consumption of memory space are still the two major problems faced by FIM, especially when the amount of data increases in a dataset. Therefore, a more robust algorithm needs to be developed for identifying the significant frequent itemsets of an incremental dataset that performs more efficiently.

6 FUTURE WORK

FIM has been a focused theme of research in the data mining field for over a decade. Recent times have witnessed abundant literature being dedicated towards its research and significant progress being made [48]. Apart from efficient and scalable algorithms being developed, numerous research frontiers have also emerged, including structured pattern mining, sequential pattern mining, correlation mining, frequent pattern-based clustering, associative classification, etc. Application-based research studies have also proven to broaden the scope of data analysis substantially, and will have a profound impact on future technologies. However, some challenging research issues still need to be solved before it can claim a cornerstone approach in data mining applications.

Scalability is an important issue that needs to be resolved in existing algorithms and to be addressed in future techniques. In the current scenario, we are able to derive the complete set of frequent itemsets under certain constraints efficiently, but the existing algorithms fail to derive a compact but high-quality set of patterns that are most useful in applications. The frequent itemsets derived by most of the current methods are too large for practical usage. There are proposals such as closed and maximal patterns, condensed pattern bases, and discriminative frequent patterns for the

reduction. Nevertheless, it is still not clear which patterns are satisfactory in terms of compactness and quality for a particular application, and whether it is possible to mine such patterns directly and efficiently.

Secondly, mining of approximate frequent patterns is often the best choice in many applications instead of a precise and complete set of frequent patterns. For example, one would like to find long sequence patterns that approximately match the sequences in biological entities in the analysis of DNA or protein sequences. Much research is still needed to make such mining more effective than the currently available tools in bioinformatics.

Moreover, we still need better mechanisms for deep understanding and interpretation of patterns, such as contextual analysis and semantic annotation of frequent patterns. The focus should be more on finding the meaning of the pattern, finding the synonym patterns, and the typical transactions that the pattern resides in. In most of the cases, frequent itemsets are mined from datasets containing only structural information. A contextual analysis of this information can help respond to questions like why the pattern is frequent, and hence improve the interpretability and usability of the frequent patterns.

Some aspiring works have already begun. One of the pioneering research aims at constructing a more efficient FIM algorithm [49]. The algorithm will be designed to mine the data from a data warehouse in order to identify the patterns that exist frequently and are hidden from the normal view of users. All the frequent patterns that have been mined from the data warehouse will be stored in a Frequent Pattern Database (FP-DB) using the technology of Not-Only Structure Query Language (NoSQL). The FP-DB will be updated continuously so that the hidden patterns of data can be mined within a shorter run-time using less memory consumption even when the amount of data increases over time.

Additionally, FIM algorithms can also be incorporated in the field of bioinformatics for biological data analysis. Typical applications include the interpretation of gene expression data, protein interaction networks, annotations, biomolecular localization prediction, etc. For example, DNA sequences contain important information, which when extracted using frequent pattern mining may help us study the evolution, function and variation of genes. It also assists in the analysis of disease causes and treatment, and genetic and mutation analysis [50]. The incorporation of a two-level nested hash table data structure and set operations is found to obtain the exact positions in DNA sequences by only one scan. The same method can also be employed in discovering frequent patterns in proteins, RNA or other biological sequences. However, one of the major factors that hampers here is not the extraction of patterns itself, but rather how they are subsequently ranked and filtered [51]. Moreover, biological problems often require domain knowledge to be formalized for defining special task-specific interestingness metrics.

One of the common goals in the field is to attribute individuals to populations, for which the strategy of comparing the genotype of the individual with the genetic profiles of defined source populations is adopted. A set of specially selected genetic markers constitute a genotype, which highlight the distinctions between individuals. For example, highly variable genetic markers such as genes or microsatellites (also called Short Tandem Repeats or STRs), and more recently Single Nucleotide Polymorphisms (SNPs), have been used for source attribution. The issue that persists is on deciding which approach is most appropriate for the particular problem in terms of assignment accuracy and computation time.

Most of the traditional approaches in the field tend to generate mutliple projected databses and short candidate patterns, leading to increased computation time and memory requirement. A fast and efficient frequent pattern mining algorithm has been proposed for multiple biological sequences (MSPM) [52]. This method works by first detecting frequent primary patterns, which are then extended to form larger patterns in the sequence. A prefix tree is constructed for the same, based on which a pattern-extending technique is employed to mine frequent patterns without producing a large number of irrelevant candidate patterns. It works on the principle that sequence similarity is the basis of genome

sequence data mining. This mining can be followed by sequence alignment and further classification and clustering using distributed or parallel computing measures.

While many algorithms aim at searching for patterns in the sequence which exceed a certain threshold, i.e., mining alternative patterns, the major drawback they face is that these methods can only mine alternating patterns in a single sequence. Perhaps future research should focus more on simultaneous analysis of repeated sequence patterns in biological sequence sets, which can work well even with massive datasets. Furthermore, since DNA sequence pattern mining typically generates an explosion of candidate sequence patterns, thereby increasing the computational time and memory requirements. Thus, designing a suitable search strategy and a method for eliminating redundant sequence patterns would be an important direction for future work.

7 CONCLUSION

Based on our survey, it is evident that most FIM approaches suffer from the problem of scanning the database more than once, which incurs high process costs and generation of candidate itemsets that require more memory space and can become complex to handle when the database is large [53]. So, to overcome the problem of candidate set generation, tree-based approaches (such as the FP-Growth algorithm) have been proposed for mining frequent itemsets. However, these algorithms also generate a massive number of conditional FP-trees. To overcome this problem, another algorithm has been put forward where the tree is constructed as a directed acyclic graph. Moreover, for effective and decisive mining fuzzy transformation can be applied to the quantitative database resulting in optimal patterns.

In this survey, we reviewed the strengths and weaknesses of the important and recent algorithms in FIM so that a more efficient algorithm can be developed. In summary, two major problems have been identified. First, the hidden patterns that frequently exist in a dataset become more time-consuming to be mined when the amount of data increases. It causes large memory consumption as a result of heavy computation by the mining algorithm. In order to solve these problems, the next stage of the research aims to: (1) formulate an FIM algorithm that efficiently mines the hidden combinations of frequent itemsets within a shorter run-time; (2) formulate the algorithm to consume less memory in mining the hidden patterns and association rules; (3) evaluate the proposed algorithm with some existing algorithms in order to ensure that it is able to mine an increased dataset within a shorter run-time with less memory consumption [54]. By implementing the proposed algorithm, users will be able to reduce the time of decision making, improve the performance and overall operation, and increase the profit of their organizations.

REFERENCES

[1] Hou, Shujun. (2021). Research on the Application of Data Mining Technology in the Analysis of College Students' Sports Psychology. Mobile Information Systems. 2021. 1-7. 10.1155/2021/6529174.

[2] Sadiku, Matthew & Shadare, Adebowale & Musa, Sarhan. (2015). DATA MINING: A BRIEF INTRODUCTION. European Scientific Journal.

[3] Győrödi, Cornelia & Gyorodi, Robert & Dr, Prof & Ing, Stefan & Stefan, Holban. (2004). A Comparative Study of Association Rules Mining Algorithms. 10.13140/2.1.1450.3365.

[4] Song, Changxin. (2016). Research of association rule algorithm based on data mining. 1-4. 10.1109/ICBDA.2016.7509789.

[5] Sharma, Sachin & Bhatia, Shaveta. (2017). A study of frequent itemset mining techniques. International Journal of Engineering & Technology. 6. 141. 10.14419/ijet.v6i4.8300.

[6] Aggarwal, Charu. (2014). Frequent Pattern Mining. 10.1007/978-3-319-07821-2_1.

[7] Yuan, J., Ding, S. (2012). Research and Improvement on Association Rule Algorithm Based on FP-Growth. In: Wang, F.L., Lei, J., Gong, Z., Luo, X. (eds) Web Information Systems and Mining. WISM 2012. Lecture Notes in Computer Science, vol 7529. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33469-6_41.

[8] Moens, Sandy & Aksehirli, Emin & Goethals, Bart. (2013). Frequent Itemset Mining for Big Data. Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013. 111-118. 10.1109/BigData.2013.6691742.

[9] Wu JM-T, Zhan J, Chobe S (2018) Mining Association rules for Low-Frequency itemsets. PLoS ONE 13(7): e0198066. https://doi.org/10.1371/journal.pone.0198066.

[10] Ziauddin, Zia & Kamal, Shahid & Ijaz, Muhammad. (2012). Research on Association Rule Mining. Advances in Computational Mathematics and its Applications (ACMA). 2. 226-236.

[11] Xia, Qiangfei & Berggren, Karl & Likharev, Konstantin & Strukov, Dmitri & Jiang, Hao & Mikolajick, Thomas & Querlioz, Damien & Salinga, Martin & Erickson, John & Pi, Shuang & Xiong, Feng & Lin, Peng & Li, Can & Xiong, Shisheng & Hoskins, Brian & Daniels, Matthew & Madhavan, Advait & Liddle, James & Mcclelland, Jabez & Raychowdhury, Arijit. (2020). Roadmap on emerging hardware and technology for machine learning. Nanotechnology. 32. 10.1088/1361-6528/aba70f.

[12] Luna, José María & Fournier Viger, Philippe & Ventura, Sebastian. (2019). Frequent Itemset Mining: a 25 Years Review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 10.1002/widm.1329.

[13] Han, Jiawei & Pei, Jian & Yin, Yiwen. (2000). Mining Frequent Patterns Without Candidate Generation. Sigmod Record. 29. 1-12. 10.1145/342009.335372.

[14] Cheung, David & Han, Jiawei & Ng, Vincent & Fu, Ada & Fu, Yongjian. (1997). A Fast Distributed Algorithm for Mining Association Rules. Parallel and Distributed Information Systems - Proceedings of the International Conference.

[15] Zaki, Mohammed. (2000). Scalable algorithms for association mining. IEEE Trans Knowl Data Eng. Knowledge and Data Engineering, IEEE Transactions on. 12. 372 - 390. 10.1109/69.846291.

[16] Zaki, Mohammed & Gouda, Karam. (2003). Fast vertical mining using diffsets. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 326-335. 10.1145/956750.956788.

[17] Mata Vázquez, Jacinto & Álvarez, José & Riquelme, José. (2001). Mining Numeric Association Rules with Genetic Algorithms. 264-267. 10.1007/978-3-7091-6230-9_65.

[18] Dean, J., & Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. Communications of the ACM , 51 (1), 107–113.

[19] Li, Haoyuan & Wang, Yi & Zhang, Dong & Zhang, Ming & Chang, Edward. (2008). Pfp: parallel fp-growth for query recommendation. 107-114. 10.1145/1454008.1454027.

[20] Lin, M., Lee, P., & Hsueh, S. (2012). Apriori-based frequent itemset mining algorithms on MapReduce. ICUIMC '12.

[21] Shorman, Hussam. (2014). An Efficient Algorithm for Mining Association Rules for Large Itemsets in Large Databases. International Journal of Engineering and Innovative Technology. 3. 237-240.

[22] Al-Maolegi, Mohammed & Arkok, Bassam. (2014). An Improved Apriori Algorithm For Association Rules. International Journal on Natural Language Computing. 3.10.5121/ijnlc.2014.3103.

[23] Chengyu, & Ying, Xiong. (2010). Research and Improvement of Apriori Algorithm for Association Rules. 1 - 4. 10.1109/IWISA.2010.5473473.

[24] Borgelt, Christian. (2010). An Implementation of the FP-growth Algorithm. Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations. 10.1145/1133905.1133907.

[25] Zeng, Yi & Yin, Shiqun & Liu, Jiangyue & Zhang, Miao. (2015). Research of Improved FP-Growth Algorithm in Association Rules Mining. Scientific Programming. 2015. 1-6. 10.1155/2015/910281.

[26] Narvekar, Meera & Syed, Shafaque. (2015). An Optimized Algorithm for Association Rule Mining Using FP Tree. Procedia Computer Science. 45. 10.1016/j.procs.2015.03.097.

[27] Nguyen, T.T.. (2013). A compact FP-tree for fast frequent pattern retrieval. 27th Pacific Asia Conference on Language, Information, and Computation, PACLIC 27. 430-439.

[28] Bansal, Urvashi. (2014). ECLAT Algorithm for Frequent Itemsets Generation. International Journal of Computer System.

[29] Kaur, M. & Bansal, Urvashi & Kaur, S.. (2015). Advanced eclat algorithm for frequent itemsets generation. International Journal of Applied Engineering Research. 10. 23263-23279.

[30] Singh, Pankaj & Singh, Sudhakar & Mishra, Kaushala & Garg, Rakhi. (2020). RDD-Eclat: Approaches to Parallelize Eclat Algorithm on Spark RDD Framework. 10.1007/978-3-030-37051-0_85.

[31] Agarwal, Ramesh & Aggarwal, Charu & Prasad, V.V.V.. (2002). A Tree Projection Algorithm for Generation of Frequent Item Sets. Journal of Parallel and Distributed Computing. 61. 350-371. 10.1006/jpdc.2000.1693.

[32] Teodoro, George & Mariano, Nathan & Meira Jr, Wagner & Ferreira, Renato. (2010). Tree Projection-Based Frequent Itemset Mining on Multicore CPUs and GPUs. Proceedings - 22nd International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2010.

[33] Rak, Rafal & Kurgan, Lukasz & Reformat, Marek. (2008). A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation. Data Knowl. Eng.. 64. 171-197. 10.1016/j.datak.2007.05.006.

[34] Feddaoui, Ilhem & Felhi, Faîçal & Akaichi, Jalel. (2016). EXTRACT: New extraction algorithm of association rules from frequent itemsets. 752-756. 10.1109/ASONAM.2016.7752322.

[35] M., Sinthuja & Puviarasan, N. & Aruna, P.. (2019). Frequent Itemset Mining Using LP-Growth Algorithm Based on Multiple Minimum Support Threshold Value (Multiple Item Support Frequent Pattern Growth). Journal of Computational and Theoretical Nanoscience. 16. 1365-1372. 10.1166/jctn.2019.8046.

[36] Baralis, Elena & Cerquitelli, Tania & Chiusano, S. & Grand, Alberto. (2013). P-Mine: Parallel itemset mining on large datasets. Proceedings - International Conference on Data Engineering. 266-271. 10.1109/ICDEW.2013.6547461.

[37] Deng, Zhi-Hong & JiaJian, Wang. (2012). A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists. Science China. Information Sciences. 55. 2008-2030. 10.1007/s11432-012-4638-z.

[38] Sadat, Masome & Nadimi-Shahraki, Mohammad H. & Soleimani Neysiani, Behzad. (2015). A New Algorithm for Mining Frequent Patterns in CAN tree CAN Growth Mining. 10.1109/KBEI.2015.7436153.

[39] Song, Mingjun & Rajasekaran, Sanguthevar. (2006). A transaction mapping algorithm for frequent itemsets mining. Knowledge and Data Engineering, IEEE Transactions on. 18. 472- 481. 10.1109/TKDE.2006.1599386.

[40] Kaur, Manpreet & Kang, Shivani. (2016). Market Basket Analysis: Identify the Changing Trends of Market Data Using Association Rule Mining. Procedia Computer Science. 85. 78-85. 10.1016/j.procs.2016.05.180.

[41] Kurnia, Yusuf & Isharianto, Yohanes & Giap, Yo & Hermawan, Aditiya & Riki, Riki. (2019). Study of application of data mining market basket analysis for knowing sales pattern (association of items) at the O! Fish restaurant using apriori algorithm. 10.1088/1742-6596/1175/1/012047.

[42] Trnka, Andrej. (2010). Market Basket Analysis with Data Mining methods. 446 - 450. 10.1109/ICNIT.

[43] Chen, Jiangping & Yang, Xiaoxian & Chen, Lihua & Dong, Li & Fu, Yuanyan. (2010). An analysis of shopping basket in a supermarket. 10.1109/ICIME.2010.5478116.

[44] Wen-xiu, Xie & Heng-nian, Qi & Mei-li, Huang. (2010). Market Basket Analysis Based on Text Segmentation and Association Rule Mining. 10.1109/ICNDC.2010.67.

[45] Garg, Ritu & Gulia, Preeti. (2015). Comparative Study of Frequent Itemset Mining Algorithms Apriori and FP Growth. International Journal of Computer Applications. 126. 8-12. 10.5120.

[46] Chee, Chin-Hoong & Jaafar, Jafreezal & Izzatdin, Abdul & Hasan, Mohd Hilmi & Yeoh, William. (2019). Algorithms for frequent itemset mining: a literature review. Artificial Intelligence Review. 52. 10.1007/s10462-018-9629-z.

[47] Mittal, Amit & Nagar, Ashutosh & Gupta, Kartik & Nahar, Rishi. (2015). Comparative Study of Various Frequent Pattern Mining Algorithms. IJARCCE. 4. 550-553. 10.17148/IJARCCE.2015.44127.

[48] Han, Jiawei & Cheng, Hong & Xin, Dong & Yan, Xifeng. (2007). Frequent pattern mining: Current status and future directions. Data Min. Knowl. Discov.. 15. 55-86. 10.1007/s10618-006-0059-1.

[49] Gupta, Adity & Tyagi, Swati & Panwar, Nupur & Sachdeva, Shelly & Saxena, Upaang. (2017). NoSQL databases: Critical analysis and comparison. 293-299. 10.1109/IC3TSN.2017.8284494.

[50] Deng, Na & Chen, Xu & Li, Desheng & Xiong, Caiquan. (2019). Frequent Patterns Mining in DNA Sequence. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2933044.

[51] Naulaerts, Stefan & Meysman, Pieter & Bittremieux, Wout & Vu, Trung Nghia & Berghe, Wim & Goethals, Bart & Laukens, Kris. (2013). A primer to frequent itemset mining for bioinformatics. Briefings in bioinformatics. 16. 10.1093/bib/bbt074.

[52] Chen, Ling & Liu, Wei. (2013). Frequent patterns mining in multiple biological sequences. Computers in biology and medicine. 43. 1444-52. 10.1016/j.compbiomed.2013.07.009.

[53] Solanki, Surbhi & Patel, Jalpa. (2015). A Survey on Association Rule Mining. 212-216. 10.1109/ACCT.2015.69.

[54] Muliono, Rizki & Muhathir, Muhathir & Khairina, Nurul & Harahap, Muhammad. (2019). Analysis of Frequent Itemsets Mining Algorithm Against Models of Different Datasets. Journal of Physics: Conference Series. 1361. 012036. 10.1088/1742-6596/1361/1/012036.