International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.45, July 2024 – www.ijais.org

Ordered Frequent Itemsets Matrix based on FP-Tree Structure and Apriori Algorithm

Abdulkader M. Al-Badani
Faculty of Science and Engineering,Department of Computers,
Aljazeera University, Ibb, Yemen

Abdualmajed A. Al-Khulaidi
Faculty of Computer Science & Information Systems,
Sana'a University, Sana'a, Yemen

ABSTRACT
Apriori and fp-growth are two well-known association rule algorithms that are well-known to data mining researchers. Nevertheless, the association rule algorithm has certain drawbacks, such as the need for large memory, lengthy dataset scans to determine the frequency of the item set, and occasionally less-than-ideal rules. To examine the rule outcomes of the three algorithms, the authors of this research compared the fp-growth, Apriori, and OFIM algorithms.In this paper, the suggest alterations to the FP-Growth algorithm's operation. By using the proposed matrix OFIM instead of the tree employed in those methods, the recommended algorithm would lower the number of often formed items and the amount of time spent mining, resulting in a considerable reduction in the amount of decision-making in large datasets. In comparison to the conventional tree-based technique, the matrix OFIM enables effective storing and retrieval of frequently occurring itemsets, leading to quicker calculation and result extraction. Furthermore, our technique significantly improves its speed in handling large datasets by limiting the amount of items that are produced often, thereby optimizing memory use.

General Terms
Data Mining, Association Rule, Frequent Itemsets Mining.

Keywords
FP-Growth Algorithm, Aprioiri Algorithm, FP-tree, Support Count, OFIM

1. INTRODUCTION
Data is crucial in the modern world and may help with many different kinds of choices. It's a wise idea to base judgments on information gathered from field data, since this may enhance decision quality[1].Among the numerous branches is the algorithm for association rules. Mining frequently occurring itemsets yields Boolean association rules using the association rule algorithm[2]. It is termed an association rule because it is the outcome of applying the features of frequently occurring itemsets. After the Apriori method was proposed, other scholars have improved and studied this technique throughout time. Of course, after improvements, the Apriori algorithm's association analysis now operates much more efficiently.

There is also the well-known FP-Growth algorithm for association rules[3]. The FP growth algorithm has significantly reduced the method's association analysis time when compared to the original association rule algorithm. Instead of repeatedly scanning the database to produce itemsets, it employs a tree structure to address the issue of numerous scans[4]. Since the FP-Growth algorithm can mine common itemsets more quickly and effectively due to its tree structure, it is a well-liked option for association rule analysis. It greatly minimizes the amount of computing time and resources needed to generate frequent itemsets by doing away with the need for several database scans.

Preparing the data, choosing the best data mining techniques, carrying out the techniques, assessing the outcomes, and interpreting the findings are some of the phases that make up the data mining process. Classification, regression, clustering, association, and ranking models are a few often used data mining approaches. These methods may be used with various kinds of data and serve distinct purposes [5, 6].The FP-Growth and apriori algorithms are two data mining techniques that are often used in inventory analysis. To identify patterns of connection and high frequency between inventory items, both of these methods may be used to inventory management [7][8].

An method known as the apriori algorithm is used in data mining to identify patterns in a set of transaction data that often occur (frequent itemsets). In order for this technique to function, the dataset is divided into smaller subsets. Each subset's items that often occur are then searched for. The discovered itemsets will then be joined to create bigger itemsets, and their frequency will be adjusted after that. Iteratively going through this technique continues until no more frequently recurring itemsets are identified. Apriori algorithms are helpful in data analysis and marketing because they allow us to identify regularly occurring trends in customer purchasing behavior and develop more successful marketing campaigns [9].

An adaptation of the Apriori approach is the FP-Growth (Frequent Pattern Growth) method [10]. The FP Growth technique detects common item sets by constructing a tree, or FP-Tree [11]. FP-Growth is a more efficient procedure thanks to the FP-Tree concept.FP-Growth is the brand-new, very successful tree-based method for mining frequently occurring item sets [12]. A divide and conquer method is carefully considered and used to minimize the size of the resultant conditional FP-tree. This will need two scans of the datasets. Less information about the transactions is shown in the FP-tree. Because a compact representation does not reduce the potential combinatorial number of candidate item sets, FP-Growth is hindered [13].Moreover, the possibly enormous size of the resultant tree means that the main memory cannot support the big database structure [14]. Therefore, rather of using the tree utilized in previous methods, the suggested solution uses a new, two-dimensional array structure based on the FP-Growth algorithm called the Ordered Frequent Itemsets Matrix (OFIM). Compared to the conventional tree-based method, the matrix OFIM enables quicker calculation and result extraction by facilitating the effective storing and retrieval of frequent itemsets.

This is how the remainder of the paper is arranged: Relevant work is presented in Section 2. In Section 3, the Research Method is explained. Section 4 goes into great information about the OFIM. In Section 5, the suggested algorithm is shown. Section 6 describes the discussions and findings of the experiment. In Section 7, the conclusion.

2. RELATED WORK

Numerous FIM-related algorithms are provided in [15,16,17, 18]. In [19], a novel FP-Linked list technique for mining association rules is given. It has put out a novel frequent pattern mining technique that uses a bit matrix to extract frequent patterns and is based on the FP-Growth concept. The goal of this approach is to increase the effectiveness and precision of often occurring pattern mining in big datasets. It provides a unique method for finding useful patterns from data by leveraging a bit matrix and the FP-Growth concept.

An Apriori Algorithm-based Analysis of Sales Patterns at University Industrial Corporation: A Step Toward Improved Customer Transaction Insights[20]. It has been suggested that they use association rules to conduct a thorough examination of product purchases. The study used the well-known Apriori algorithm, which is skilled at identifying frequently occurring objects in transactional databases, to achieve this goal. According to the research, important trends in the buying behavior of customers were discovered via the use of the Apriori algorithm. University Industrial Corporation was able to improve customer satisfaction and sales initiatives by using the insightful information this investigation offered.

Apriori and FP-Growth Algorithm Performance Analysis (Association Rule Mining) [21]. They compared two algorithms (Apriori and FP-growth) in their study using Weka, taking into account the characteristics of the database scan (number of instances, confidence, and support levels). The superiority of the FP-Growth algorithm over the apriori approach is evident. In terms of efficiency and execution time, the FP-Growth algorithm fared better than the Apriori method, particularly when taking into account characteristics related to database scans, such as the number of instances, confidence, and support levels. This implies that FP-Growth, as opposed to Apriori, would be a better option for association rule mining jobs.

A Better Apriori Algorithm for Association Rules [22]. It has suggested, This work proposes an improvement on Apriori by minimizing the spent time dependent on scanning just select transactions. Based on this method, it highlights the restriction of the original Apriori algorithm of spending time for scanning the full database looking on the frequent itemsets. [23] presents an improved apriori algorithm for mining association rules. The goal of this research is to create an association-rules-mining algorithm that is very efficient and simple to use. PGB (Pattern-Growth-Based) approach. The suggested technique, dubbed PGB-Apriori, mines association rules effectively by using a pattern-growth-based approach. PGB-Apriori helps save a lot of time by concentrating on a small number of transactions rather than searching the whole database for frequently occurring itemsets. FPtree is a tree-like data structure that is used in conjunction with a Depth-Frist search technique. PGB approach algorithms locate all frequent 1-itemsets iteratively in the conditional pattern base, which is effectively built with node link represented by Frequent-Pattern-tree (FP-tree), and thus identify the frequent itemsets. The FP-Growth algorithm [24] is the first algorithm based on the PGB approach. By using a small data structure, this technique mines common itemsets well and does not need candidate creation. It has been shown that the FP-Growth algorithm performs faster and uses less memory than Apriori and other conventional algorithms.

In [25], an enhanced version of the FP-Growth method for mining description-oriented rules is presented. Using an FP-Growth algorithm based on Gene Ontology (GO), they have presented a novel modification for the description of gene groups. The findings indicate that the new approach allows for the quicker generation of rules. In [26], a novel FP-Linked list technique for mining association rules is given. It has suggested a brand-new frequent pattern mining technique built on the FP-Growth concept, which extracts frequent patterns utilizing a bit matrix and a linked list structure. The goal of the suggested method in [26] is to increase mining association rule efficiency by combining bit matrices and linked lists. The results show that this method performs faster and more scalable than conventional FP-Growth algorithms.

In [27], effective techniques for data mining-based frequent itemset discovery are presented. Based on the frequent pattern growth method, these strategies are presented to provide privacy, usefulness, and efficiency in frequent itemsets mining. In [28], a better frequent itemset mining algorithm is created. It has suggested a non-recursive, more effective FPNR-growth method that enhances performance in both time and space. When compared to conventional methods, the FPNR-growth algorithm drastically lowers both computational complexity and memory utilization. This enhancement makes frequent itemset mining in big datasets quicker and more resource-efficient.

In, a brand-new hybrid frequent pattern-Apriori (FP-AP) high utility item set mining method is created [29]. The FP-AP algorithm, which combines the Apriori algorithm with frequent patterns, has been introduced. It is suggested that FP break lengthy transactions rather than terminate them and identify the most lucrative item while maintaining privacy for that item set and avoiding repetition. By combining the advantages of frequent pattern and Apriori algorithms, the FP-AP method seeks to increase the effectiveness and precision of high value item set mining. The approach improves the overall speed of item set mining while preserving privacy and minimizing duplication by using FP to partition lengthier transactions and prioritize highly valuable items.

In [30], an association rule mining survey on FP-Growth trees is given. A novel method that eliminates the need to create conditional FP-trees in order to extract all frequent itemsets was presented by the researchers. By not creating conditional FP-trees, this method seeks to increase the mining efficiency of frequent itemsets. The study sheds light on the advantages and restrictions of association rule mining using FP-Growth trees.

This paper presents the optimization of the FP-Growth algorithm against the background of big data and cloud computing [31]. In this article, a parallel mining method is explored. Every node computer uses the improved method to produce fragmented frequent itemsets via parallel mining. Then, summary is used to get all frequently occurring itemsets [32]. A matching projection database is created for every frequent item after the transaction databases associated with that item have been extracted. The goal of the suggested method is to use cloud computing's parallel processing power to mine frequent itemsets in large-scale datasets more efficiently. The algorithm can handle enormous volumes of data more efficiently by dividing the burden over numerous

nodes, leading to quicker and more accurate outputs.

Finding Common Itemsets using a Signature-based Tree in [33].The authors of this paper propose a novel tree-based structure that prioritizes transactions over itemsets. Consequently, we avoid the problem of support values that negatively impact the generated tree. The fp-growth algorithm is the basis for frequent item sets mining, and several approaches have been put forward to achieve this goal while maintaining effectiveness, privacy, and value [34]. By effectively addressing the problem of support values, the suggested tree structure makes it possible to mine frequent itemsets with more accuracy and dependability. Through the integration of privacy, usefulness, and effectiveness factors, the authors provide a thorough method for frequent itemset mining that tackles a range of issues within the industry.

The most significant contribution of this study is a new algorithm that effectively solves complicated optimization problems in a fraction of the time needed compared to previous approaches by using the OFIM structure. , and then enhance the functionality of the algorithms running on FP-trees. This algorithm has been tested on a variety of datasets and has consistently surpassed previous algorithms in terms of accuracy and speed. This new method has the potential to completely change the optimization profession by solving complicated problems more quickly and accurately. Further investigation may also look at the applicability of this approach to optimization issues other than the ones that were examined in this paper.

3. RESEARCH METHOD

The database knowledge discovery process, also known as knowledge discovery in the database, or KDD for short, includes an analytical stage called data mining. bias information in the form of previously unknown data patterns or connections between reliable data. The process of finding new patterns from extremely big data sets using techniques slicked from artificial intelligence, machine learning, statistics, and database systems is known as data mining, which is a synthesis of numerous computer science fields. In order to find patterns and linkages that may be used to make wise judgments, it entails sifting through vast volumes of data. In today's data-driven world, data mining is an essential tool for firms seeking to remain competitive and obtain insights.Aiming to extract (take the essence) knowledge from a set of data so that a structure can be understood by humans, data mining encompasses post-processing of the structures found, online visualization, interest measures, model and inference considerations, database and data management, and data processing [35].

Data cleaning, data integration, data selection, data transformation, application of data mining methods, pattern evacuation, and knowledge presentation are some of the processes in data mining. Generally speaking, there are two types of data mining techniques: predictive and descriptive. Descriptive data mining refers to the process of searching for human-understandable patterns within the data that explain its features. Predictive, on the other hand, refers to the process of creating a knowledge model via data mining [36]. Classification, clustering, association, regression, forecasting, sequence, and deviation are some of the data mining techniques now in use.

3.1 Association rule mining

A data mining approach called association analysis may be used to uncover intriguing connections between a group of hidden elements in a database. Association norms are one way to express this connection [37]. The goal of association analysis is to identify the connection between two or more characteristics. Association rules often have the format IF antecedent THEN consequent. Support and confidence are two metrics that may be used to gauge how strong an association rule is. The proportion of these things combined in the database that is called support (support value) and confidence (certainty value). Specifically, the associative rules' strong relationships between things.

Group of items item = {it1, it2,... itn} and transaction of data items in the database db = {tr1, tr2,... trn} are included in the association rule. G → H is the association rule's consequence. G and H are itemsets of {fi1 = val1, fi2 = val2, fi3 = val3,..., fin = valn}, where f is a field name or item name and val is an item value. This represents network-based intrusion detection. Assuming that G is a part of item and H is a part of item, the association rule G → H satisfies the minimal confidence and support [38]. In network-based intrusion detection, association rules are employed to find behavioral patterns that can point to a security risk. For these rules to be deemed reliable for identifying possible intrusions, they must fulfill a number of requirements related to trust and support.

The rule of association In both G and H, support is calculated as a percentage of transactions. The connection between the relevance of items discovered together as itemsets is measured. The frequency of the item in the database serves as the primary indicator of support. The more often the goods appear together in transactions, the greater the support value. Using this data to spot trends and base business choices on the behavior of your customers is essential.

3.1.1 Apriori Algorithm

This algorithm elaborates to determine subsets which are common to at least a minimum number of the item sets. The demonstrate frequent pattern mining based on support and confidence measures produced desired output in various fields.

L_1 {large1-itemsets} \\count item frequency
for (K=2; Lk_{-1} # {}; k++)
do begin
C_k=Apriori-gen (L_{k-1}); \\ new conditions
for all transactions t Є D
do begin
C_t=subset (C_k,t); \\candidates in transaction
for all Candidates' c Є C, do
C.count $^{++}$; \\determine support
end
L_k={c Є C_k| c. count ≥ min sup} \\create new set
end
Result= U_k L_k;

The Apriori Algorithm's Function: Generally speaking, the Apriori Algorithm is a two-step procedure [39]:

- All item sets that have a support factor more than or equal to the minimum support amount entered by the user are created.

- All produced rules have a confidence factor that is either higher than or equal to the minimal confidence that the user has selected.

3.1.2 FP-GROWTH ALGORITHM

The first column lists the names of the feature items, which are sorted in descending order of their support while the second column stores a chain table that connects the nodes of the same items in the FP-tree. The FP-Growth algorithm is mainly divided into two steps: building the FP-tree and mining the frequent itemsets based on the FP-tree recursion.The feature item names are listed in the first column in decreasing order of support, and a chain table connecting the nodes of the same items in the FP-tree is stored in the second column. Building the FP-tree and mining the frequent itemsets using the FP-tree recursion are the two primary phases of the FP-Growth algorithm. The FP-tree is built in the construction stage by the algorithm putting transactions into the tree after repeatedly scanning the dataset. After the FP-tree is constructed, the mining stage generates conditional FP-trees and traverses the tree to recursively extract frequent itemsets.

Below [42] is the pseudo-code for a transaction database's FP-Growth algorithm:

Dataset D as input; support threshold min_sup

FP-tree as the output

1. To get the frequent 1 item set L1, first traverse the dataset, compute the support of each feature item, sort in decreasing order, then filter out the infrequent items compared with min_sup.

2. Establish the FP-tree's root node, designate it as T, and designate "null" for the root node's content. Make the table of frequently used objects and make the connection null. Proceed with the second iteration of the dataset as follows.

3. for the D do transaction.

4. Sort the items in the transactions based on the feature item order in L1, filter the frequently occurring items based on L1, and record the items as P;

5. Place P inside T. If T has a P prefix, increase the count of each prefix node by 1, and create a new node that only has a count of 1 for the item that comes after the prefix;

6. Modify the relevant links in the table of frequently used objects.

There is a header table connected to the FP-tree. In the header table, single items and their counts are kept in decreasing order of frequency. An example of a transactional dataset is provided in Table 1, and the FP-tree produced by the FP-Growth algorithm from this dataset is shown in Figure 1. Every node in the FP-tree denotes an item and its frequency count. Without creating candidate sets, the tree structure enables effective mining of frequently occurring itemsets.

Table 1: A dataset with nine transactions.

TID	List of items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

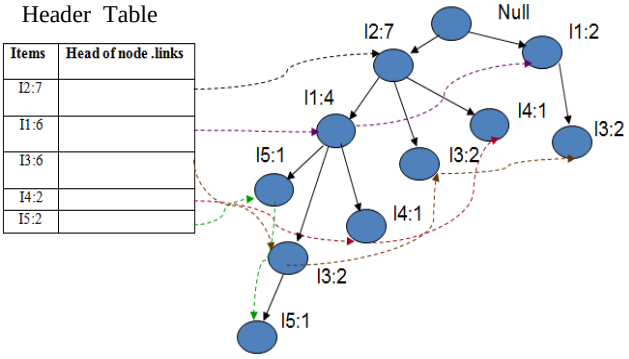

Figure 1: An Example of FP-tree(*minsup*=50%).

Table 2 displays the frequent itemsets that were generated.

Table 2: The discovered frequent itemsets by FP-Growth algorithm.

TID	Conditional FP-tree	Frequent itemsets
I5	<I2:2,I1:2>	{I2,I5:2}, {I1,I5:2}, {I2,I1,I5:2}
I4	<I2:2>	{I2,I4:2}
I3	<I2:4,I1:2>,<I1:2>	{I2,I3:4},{I1,I3:4}, {I2,I1,I3:2}
I1	<I2:4>	{I2,I1:4}

4. ORDERED FREQUENT ITEMSET MATRIX

All frequent itemsets are included in OFIM, a two-dimensional array that is used to summarize transactional databases. The itemsets are arranged in support descending order. The OFIM is N×M, where M is the longest number of often ordered goods and N is the number of transactions. Each itemset's support value is kept in the associated array cell. This makes it possible to quickly and effectively get the most crucial data about frequently used itemsets from the database.

The suggested method looks through the transactional information to provide a list of often occurring items that are arranged in decreasing order of frequency. This ordering is significant since it will dictate how OFIM is constructed. Ordered Frequent Itemsets Lists (OFILs) are lists of often occurring itemsets that are added to the list of candidate itemsets for every transaction whose occurrence frequencies are higher than or equal to the minsup threshold. The frequent itemsets are then combined using the OFILs in a manner that preserves the frequency decreasing order to create the OFIM. By concentrating on the most significant items first, this method narrows the search area and enables effective mining of frequently occurring itemsets. The remaining non-frequent candidate itemsets are thrown away. The rightmost column of Table 3 lists the frequently occurring elements in each transaction in this sequence. For the purpose of finding relationships and patterns in the data, these frequently occurring itemsets are essential.

Table 3: Transactional dataset with OFILs.

TID	List of items	OFILs.
T1	I1,I2,I5	I2,I1,I5
T2	I2,I4	I2,I4
T3	I2,I3	I2,I3
T4	I1,I2,I4	I2,I1,I4
T5	I1,I3	I1.I3
T6	I2,I3	I2,I3
T7	I1,I3	I1,I3
T8	I1,I2,I3,I5	I2,I1,I3,I5
T9	I1,I2,I3	I2,I1,I3

Observe how the transaction's frequent items are arranged in accordance with their listing in the frequent things list. Table 3 shows that transaction I1,I2,I5 has an OFIL of I2,I1,I5. "0" values are used to initialize an empty OFIM with N×M. The OFILs are scanned list by list throughout the matrix generating process. Items are taken out of each list by the procedure. One by one, it then adds the objects to the matrix's rows and matching columns. The procedure is iterated for every list inside the OFILs. After reviewing each OFIL listed in Table 3, Table 4 provides a comprehensive description of the OFIM.The OFIM matrix will be started with "0" values and have dimensions of N x M. After reading each OFIL list, entries are taken out and inserted to the matrix as needed. After processing each of the OFILs from Table 3, the final OFIM is described in full in Table 4.

Table 4. The OFIM.

T1	I2	I1	I5	0
T2	I2	I4	0	0
T3	I2	I3	0	0
T4	I2	I1	I4	0
T5	I1	I3	0	0
T6	I2	I3	0	0
T7	I1	I3	0	0
T8	I2	I1	I3	I5
T9	I2	I1	I3	0

5. THE PROPOSED ALGORITHM

Small data sets may benefit from the basic FP-Growth method, but large data sets cannot be served by it due to the time-consuming nature of creating an FP-tree and locating many frequent itemsets. As a result, and may not fit in the main memory due to the expanding FP-tree. Finding often occurring itemsets involves using the OFIM and a minsup threshold as inputs. The memory restriction is addressed by the One-Itemset-at-a-Time Mining (OFIM) technique, which uses a two-dimensional array that is updated when new itemsets are found. When working with big datasets, this method enables scalability and effective memory management. The technique can handle higher data amounts without experiencing memory limits since it only focuses on one itemset at a time.

With the exception of the sets of items that did not get support, the suggested method starts scanning from the last column and computes the support for each item in each column in OFIM. Then, in order to decrease the search area and increase efficiency, the algorithm prunes infrequent itemsets. In big datasets, this method makes it possible to identify common itemsets more quickly. We will eliminate them, and for every item that received support in the final column, the will calculate the frequency of the prior itemsets associated with it and those that are comparable in its records. The will then remove these itemsets from OFIM.By eliminating the tree and cutting down on the time and effort required to generate the frequent itemsets, we are able to expedite the process of finding patterns in the data.

The algorithm may concentrate on examining just the most relevant and meaningful patterns by removing uncommon itemsets early on, which produces more effective data mining outcomes. Thus, performance is much better than using the FP-Growth method.Non-frequent itemsets are effectively eliminated by the OFIM algorithm by being discarded early on. Because only the most relevant itemsets are taken into consideration, this makes it possible to find patterns in the data in a more streamlined and effective manner.Consequently, the OFIM algorithm's total performance is markedly enhanced in contrast to other techniques such as FP-Growth. It minimizes processing time and computational load by only taking into account itemsets that have obtained support, hence producing more frequent itemsets. It is a more effective substitute for the FP-Growth algorithm due to its enhanced performance. Furthermore, the streamlined methodology of the OFIM algorithm allows it to easily manage bigger datasets. Its superiority for frequent itemset mining jobs is further reinforced by its scalability. The following are extensive descriptions of the suggested algorithm:-

A DB of transactions and a minsup threshold constitute the inputs.

Output: identified recurring item sets.

1.Perform a database scan after each transaction. Compile F, the frequent item sets of F, and F's supporters. Sort F in descending order as OFIL, the list of frequently sorted items. During this phase, every set of infrequent items is eliminated.

2.Create the OFIM. Every item ordered frequently used in the OFIL is individually entered into the corresponding columns for each row associated with the OFIL.

3- Generation of frequent itemsets.

3.1- Assume that the OFIM column number is c.

3.2- For (c= M; c>=1; c--)

{

If c=1 Then Do

{

The current column (c) and the columns that came before it compare the collection of frequently occurring items and compile their supporters. Let the output be [r, f: n | OFIL] where f is the current frequent item in column (c) and r is the parent frequent item of the preceding columns.

rent frequent item of the preceding columns.

We take the matching rows of item r from the collection of frequent items and compile the previous columns of item f from the current column's (c) supporters. Remove these rows from the OFIM as well.

}

Else Do

{

Go to column (c) before, compare the collection of frequently used items, and gather the supporters who support each item. Let the output be [r, f: n | OFIL] where f is the current frequent item in column (c) and r is the parent frequent item of the preceding columns.

Extract these rows for the recurring parent items. For the repeated item, f, it is processed according to its order. And delete these rows from OFIM.

}

}

The suggested algorithm creation of recurring itemsets in this manner: begin by calculating the support for each item in the OFIM's last column. and the items in the current column are distinguished using the other (prior) column.The previous column is moved from the current column after the support for the various items in the current column is calculated. The items that obtain the support are identified as the frequent itemsets, and these rows are removed from the OFIM. For every column, repeat the preceding procedures. This procedure is carried out by the algorithm until no more frequent itemsets can be produced. Through a methodical analysis of each OFIM column, this approach effectively finds patterns in the data.

The generated frequent item sets are shown in Table 5. The item sets are arranged in the table according to their support values, with the most often occurring sets at the top. The data's patterns and trends may be found using this information.

Table 5. Displays the created frequent item sets

Frequent itemsets
{I2,I3:2}, {I1,I3:2}, {I2,I1,I5:2}
{I2,I1,I3:2}

6. RESULTS AND DISCUSSIONS

The real-world datasets used to verify the suggested algorithm's performance are available from the UCI Machine Learning Repository, a repository of widely used benchmark and real-world datasets for the data mining and KDD communities [43]. These datasets enable academics to test their algorithms on a variety of data sources, spanning a broad range of fields including biology, economics, and social sciences. These datasets are freely accessible to researchers, who may use them to evaluate the effectiveness of various machine learning strategies. The suggested method's performance is assessed and contrasted with the well-known FP Growth algorithm in terms of the amount of time needed to locate frequent itemsets from the given datasets as well as the quantity of frequent itemsets that are found. The findings demonstrate that the suggested approach works better than the FP-Growth algorithm in terms of time efficiency and the quantity of frequent itemsets discovered. This illustrates the new algorithm's efficacy and promise for mining frequent itemsets in big datasets. Overall, the research emphasizes how important it is to create good frequent itemset mining algorithms in order to efficiently manage enormous datasets. Subsequent investigations may concentrate on refining the suggested algorithm to achieve even more efficiency. Every experiment is run on a laptop running 32-bit Windows 10 with Python, 16GB of RAM, and an Intel(R) Core(TM) i3-2375M CPU @ 1.50GHz 1.50 GHz. A statistical breakdown of the datasets utilized in this comparison analysis is shown in Table 6. The datasets range from tiny to large-scale data sets in terms of both size and complexity. This data is essential for comprehending how well the algorithms work with various kinds of data.

Table 6: Characteristics of the test datasets

Datasets	Size	#Transactions
Grocery	0.56MB	10800
Mushroom	0.12MB	8124
Car	0.055MB	1728

Using these three datasets, the comparative performance of the suggested algorithm and the FP-Growth algorithm is shown below:-

6.1 Experiment One

The Grocery dataset was used to conduct this experiment. It includes data from actual point-of-sale transactions from a regular neighborhood grocery store for a month. To effectively assess the effectiveness of the suggested method in comparison to the original FP-Growth algorithm, many tests have been carried out using varying values of minsup and compared. The Grocery dataset offered a realistic and useful setting for evaluating the performance of the algorithm. Researchers were able to evaluate how the suggested method worked better under various circumstances than the original FP-Growth algorithm by adjusting the minsup settings. Table displays the findings derived from the execution time needed to identify the frequent itemsets and the quantity of frequent itemsets with different minsup values, such as 10%, 20%, 30%, and 50%.

Table 7: Comparison results for the Grocery dataset with various *minsup* thresholds.

No.	*minsup*	Execution time per milliseconds (s)			# Discovered Frequent itemsets		
		Apriori	FP-Growth	New algorithm	Apriori	FP-Growth	New algorithm
1	10%	0.40	0.17	0.10	19	11	8
2	20%	0.23	0.15	0.08	19	9	7
3	30%	0.21	0.13	0.07	17	8	7
4	50%	0.73	0.1	0.03	13	5	3

Both algorithms' execution times and the quantity of frequently found itemsets often decrease as minsup values rise.Although the suggested approach modifies the minsup threshold value, it is found that its execution time is less than that of the FP-Growth algorithm. The performance of two algorithms for four distinct minsup thresholds is shown in Figure 3 based on their execution times. It demonstrates unequivocally how much better the suggested approach is over the FP-Growth algorithm. The FP-Growth algorithm requires a significant amount of time and memory to create several conditional sub-trees before producing a big number of frequent itemsets.

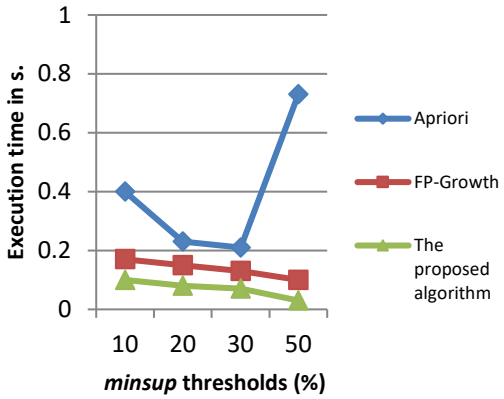

Figure 3: Comparing the results of the execution time and the *minsup* thresholds for the Grocery dataset.

6.2 Experiment two

This experiment made use of the mushroom dataset. There are 8124 entries in this collection, and there are 23 characteristics totaling 119 items. The collection contains data on a number of mushroom attributes, including habitat, odor, and cap shape. Based on these characteristics, machine learning projects often utilize it to categorize mushrooms as edible or toxic. Table 8 displays the execution duration, the number of FP-Growth frequent item sets that were found, and the suggested method with different minimum criteria, including 5%, 8%, 12%, and 20%.

Table 8: Comparison results for the Mushroom dataset with various *minsup* thresholds

No.	*minsup*	Execution time per milliseconds (s)			# Discovered Frequent itemsets		
		Apriori	FP-Growth	New algorithm	Apriori	FP-Growth	New algorithm
1	5%	1969.95	106.48	2.13	89855	10172	8
2	8%	1049.60	61.45	1.86	53167	3451	6
3	12%	676.47	36.70	1.77	34335	1209	5
4	20%	326.57	18.44	1.36	14703	287	2

Figure 4 compares the three algorithms' results based on their execution times using four distinct minsup thresholds.

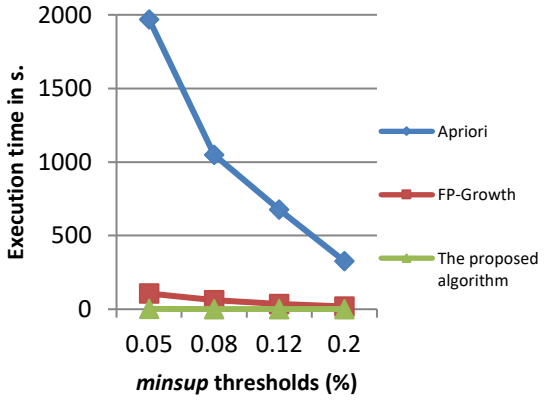

Figure 4: Comparing the results of the execution time and the *minsup* thresholds for the Mushroom dataset.

This chart shows that the number of created frequent itemsets and the execution time of three algorithms reduce considerably with increasing minsup threshold. This implies that by lowering the amount of itemsets that must be taken into account, raising the minsup threshold may greatly increase the algorithms' efficiency. It also suggests that for data mining jobs, a higher minsup threshold may result in quicker execution durations and more controllable outcomes.

7. CONCLUSION

One component of the association rule algorithm using an associative method is the FP-Growth and Apriori algorithms. The Apriori method has the benefit of being able to generate ideal rule combinations and is simple to use. The very lengthy dataset scan time is the problem, however, since FP-Growth must build a sizable number of conditional sub-trees in order to generate a sizable number of frequent item sets.This is a memory-intensive and time-consuming operation. This study's primary goal is to compare three algorithms: OFIM, fp-growth, and Apriori. This work proposes an improved FP-Growth approach for efficient mining of frequent itemsets. By leveraging OFILs to create the OFIM, the proposed method improves mining efficiency in the big data environment. Therefore, after extracting the set of frequent items using OFIM, the proposed method produces less frequent itemsets.

The three algorithms' execution times for different minsup values to assess each one's effectiveness. It clearly shows how much better the suggested method performs than the Apriori and FP-Growth algorithms. The FP-Growth method must build a sizable number of conditional sub-trees before producing a sizable number of frequent item sets. This is a memory-intensive and time-consuming operation. However, the suggested technique produces frequent item sets more effectively and quickly since it does not need building conditional sub-trees. This demonstrates how the suggested approach performs faster and more efficiently than the Apriori and FP-Growth algorithms.

On the other hand, the suggested approach effectively creates frequent item sets without requiring the creation of conditional sub-trees. This is a more effective and scalable approach than the FP-Growth method as it cuts down on both execution time and memory utilization. Furthermore, when the minsup values rise, the suggested algorithm's performance boost becomes more noticeable, underscoring its superiority over FP-Growth and Apriori.

.8. REFERENCES

[1] Riadi, I., Herman, H., Fitriah, F., Suprihatin, S., Muis, A., & Yunus, M. 2023. Implementation of association rule using apriori algorithm and frequent pattern growth for inventory control. Jurnal Infotel, 15(4),pp. 369-378.

[2] Dunham M, Naughton J, Chen W D, et al. 2010..Proceedings of the 2000 ACM SIGMOD international conference on Management of data[J]. Water International, 26(4),pp. 607-609.

[3] Singh R, Bhala A, Salunkhe J, et al.2015. Optimized Apriori Algorithm Using Matrix Data Structure[J]. International Journal of Research in Engineering and AppSciences,9(5),pp. 2249-3905.

[4] Yu, C., Liang, Y., & Zhang, X. 2023. Research on Apriori algorithm based on compression processing and hash table. In Third International Conference on Machine Learning and Computer Application (ICMLCA

2022) (Vol. 12636, pp. 606-611). SPIE.

[5] A. S. Hoong Lee, L. S. Yap, H. N. Chua, Y. C. Low, and M. A. Ismail. 2021. "A data mining approach to analyse crash injury severity level," J. Eng. Sci. Technol., vol. 16, pp. 1–14.

[6] S. Wang, J. Cao, and P. S. Yu. 2019 . "Deep learning for spatiotemporal data mining: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 8, pp. 1–21.

[7] Elisa, E. 2018. Market Basket Analysis Pada Mini Market Ayu Dengan Algoritma Apriori. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 2(2),pp. 472-478.

[8] Salam, A., Zeniarja, J., Wicaksono, W., & Kharisma, L. 2018. Pencarian Pola Asosiasi Untuk Penataan Barang Dengan Menggunakan Perbandingan Algoritma Apriori Dan Fp-Growth (Study Kasus Distro Epo Store Pemalang). Dinamik, 23(2),pp. 57-65.

[9] M. D. Febrianto and A. Supriyanto. , 2022. "Implementasi algoritma apriori untuk menentukan pola pembelian produk," Jurikom, vol. 9, no. 6, pp. 2010–2020.

[10] M. M. Hasan and S. Z. Mishu..2018. "An adaptive method for mining frequent itemsets based on apriori and fp growth algorithm," in 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2). IEEE, pp. 1–4.

[11] A. Almira, S. Suendri, and A. Ikhwan, 2021."Implementasi data mining menggunakan algoritma fp-growth pada analisis pola pencurian daya listrik," Jurnal Informatika Universitas Pamulang, pp. 442–448.

[12] J. Han, J. Pei, and Y. Yin. .2000. "Mining frequent patterns without candidate generation," ACM sigmod record, no. 2, pp. 1– 12.

[13] F. Wei and L. Xiang. 2015. "Improved frequent pattern mining algorithm based on fp-tree," in Proceedings of The Fourth International Conference on Information Science and Cloud Computing (ISCC2015), pp. 18–19.

[14] R. Krupali, D. Garg, and K. Kotecha. 2017. "An improved approach of fp-growth tree for frequent itemset mining using partition projection and parallel projection techniques," International Recent and Innovation Trends in Computing and Communication, pp. 929–934.

[15] AGRAWAL, Rakesh, et al. 1994. Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. pp. 487-499.

[16] HAN, Jiawei; PEI, Jian; YIN, Yiwen. 2000. Mining frequent patterns without candidate generation. ACM sigmod record, 29.2: 1-12.

[17] SHRIDHAR, M.; PARMAR. 2017. Mahesh. Survey on association rule mining and its approaches. Int J Comput Sci Eng, 5.3: 129-135.

[18] KHANALI, Hoda; VAZIRI, Babak. 2017. A survey on improved algorithms for mining association rules. Int. J. Comput. A,pp. 165: 8887.

[19] Sohrabi, M. K., & HASANNEJAD, M. H. 2016. Association rule mining using new FP-linked list algorithm.

[20] Huaman Llanos, A. A., Huatangari, L. Q., Yalta Meza, J. R., Monteza, A. H., Adrianzen Guerrero, O. D., & Rodriguez Estacio, J. S. 2024. Toward Enhanced Customer Transaction Insights: An Apriori Algorithm-based Analysis of Sales Patterns at University Industrial Corporation. International Journal of Advanced Computer Science & Applications, 15(2).

[21] BALA, Alhassan, et al. 2016. Performance analysis of apriori and fp-growth algorithms (association rule mining). Int. J. Computer Technology &Applications, 7.2,pp. 279-293.

[22] Al-Maolegi, M., & Arkok, B. 2014. An improved Apriori algorithm for association rules. arXiv preprint arXiv:1403.3948.

[23] Yuan, X. 2017. An improved Apriori algorithm for mining association rules. In AIP conference proceedings (Vol. 1820, No. 1). AIP Publishing.

[24] Han J, Pei J, Yin Y. 2000. Mining frequent patterns without candidate generation (Acm Sigmod Record, 29(2)), pp.1-12.

[25] Gruca, A. 2014. Improvement of FP-Growth algorithm for mining description-oriented rules. In Man-Machine Interactions, Part of Advances in Intelligent Systems and Computing, (AISC), Springer, vol. 242, pp. 183-192.

[26] Sohrabi, M. K., and Marzooni, H. H. 2016. Association rule mining using new FP-Linked list algorithm. Journal of Advances in Computer Research (JACR), 7(1), pp. 23-34.

[27] Sagar, B. P., and Kale, S. 2017. Efficient algorithms to find frequent itemsets using data mining. International Research Journal of Engineering and Technology (IRJET), 4(6), pp. 2645- 2648.

[28] Hao, J., and Xu, H. 2017. An improved algorithm for frequent itemsets mining. In 5th International Conference on Advanced Cloud and Big Data (CBD), IEEE Computer Society , pp. 314-317.

[29] Devi, R. S., and Shanthi, D. 2016. A new hybrid frequent Pattern-Apriori (FP-AP) algorithm for high utility item set mining. Middle East Journal of Scientific Research (MEJSR), 24(3), pp. 986-991.

[30] Princy. S, Ankita, H., Babita, P., and Shiv, K. 2017. A survey on FP (Growth) tree using association rule mining. International Research Journal of Engineering and Technology(IRJET), vol. 4, Issue 7, pp. 1637-1640.

[31] B. Zhang. 2021 ."Optimization of fp-growth algorithm based on cloud computing and computer big data," International Journal of System Assurance Engineering and Management, pp. 853–863.

[32] S. X. Le Zhang, X. Li, X. Wu, and P.-C. Chang. 2019. "An improved fp-growth algorithm based on projection database mining in big data," Journal of Information Hiding and Multimedia Signal Processing, pp. 81–90.

[33] M. El Hadi Benelhadj, M. M. Deye, and Y. Sliman. 2023. "Signaturebased tree for finding frequent itemsets," Journal of Communications Software and Systems, pp. 70–80.

[34] S. Bhise and S. Kale. 2017. "Effieient algorithms to find

frequent itemset using data mining," Int. Res. J. Eng. Technol., pp. 2645–2648.

[35] AL-ZAWAIDAH, Farah Hanna; JBARA, Yosef Hasan; MARWAN, A. L. 2011." An Improved Algorithm For Mining Association Rules In Large Databases",World Of Computer Science And Information Technology Journal, 1.7,pp. 311-316.

[36] Dr. Suyanto, S. M. 2017."Data Mining Untuk Klasifikasi Dan Klasterisasi Data", Bandung: Informatika.

[37] WINARTI, Titin; INDRIYAWATI, Henny. 2023. Data Mining Modeling Feasibility Patterns of Graduates Ability With Stakeholder Needs Using Apriori Algorithm. International Journal of Information Technology and Business, 4.2,pp. 55-60.

[38] Sivanantham, S., Mohanraj, V., Suresh, Y., & Senthilkumar, J. 2023. Association Rule Mining Frequent-Pattern-Based Intrusion Detection in Network. Computer Systems Science & Engineering, 44(2).

[39] Maragatham, M. Lakshmi, and G. Maragatham, , 2012. "A Recent Review on Association Rule Mining," in Indian Journal of Computer Science. vol. 2.

[40] Han, J., Pei, J., & Yin, Y. 2000. Mining frequent patterns without candidate generation. ACM sigmod record, 29(2),pp. 1-12.

[41] ZENG Xia, ZHANG Fuqiang, SHAO Shujun, DU Chao. , 2022. Fault Feature Analysis for CNC Machine Tools Based on FP-Growth Algorithm [J]. MACHINE TOOL & HYDRAULICS50(16),pp. 174-180.

[42] GU, Juan; JIANG, Tianyuan; SHEN, Lei. 2023. Equipment maintenance data mining based on FP-growth algorithm. In: International Conference on Electronic Information Engineering and Data Processing (EIEDP 2023). SPIE,. pp. 216-222.

[43] Blake, C. L., and Merz., M. J, UCI Repository of Machine Learning Databases [http://www. ics. uci. edu/~ mlearn/ MLRepository. html]. Irvine, CA: University of California, Department of Information and Computer Science.