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

Developing an Efficient Mining of Frequent Itemsets using OFIM for Big Data

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
Big data mining is challenging.An effective algorithm and computer software are needed to solve problems while working with large datasets.The FP Growth Algorithm takes a long time to compute and extract results, and it demands a lot of memory.Right now, the FP_Growth algorithm is among the finest methods for mining frequent itemsets.The transaction dataset is used to create a tree structure, which is then recursively traversed to extract frequently occurring itemsets using a depth first search strategy.Additionally, creating an FP_tree requires time and suffers from growing larger FP_trees and producing a high number of frequent itemsets.In this paper, the suggest alterations to the FP_Growth algorithm's operation.With our usage of the proposed matrix OFIM to build a very compact FP-tree, the recommended approach would cut mining time and the number of regularly generated items, giving a considerable reduction in decision_making in large datasets.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, Ordered Frequent Itemset Matrix.

1. INTRODUCTION
Large-scale improvements in the amount of data that can be collected and stored in a database have been made possible by the development of information technology in the modern era, which has made the creation of techniques that make the gathering of extraordinarily big data sets easier necessary [1]. Utilizing big data analytics is one such technique that makes it possible to handle and analyze enormous volumes of data in order to uncover important patterns and insights. Due to the ability to make better decisions based on data-driven information, this has completely changed a number of sectors, including marketing, banking, and healthcare. Big data analytics programs have fundamentally altered how people live their everyday lives.

A wide range of big data analytics solutions have been created in response to the increasing interest in data_driven decision making [2].One tool that may be used is data mining.Data mining is the process of looking through certain data to find interesting patterns or information[3][4].A key element of big data analytics is data mining, which enables businesses to glean insightful information from enormous volumes of data.In today's data_driven world, organizations may gain a competitive advantage and make better choices by using data mining tools.

Sets of items are a common component of data from the physical world, such as collections of goods that were bought together in a supermarket. A frequent itemset, for instance, is a group of items in a transaction dataset that exhibit a recurrent trend.For the frequent itemset, the following definitions are applicable:-

Let L be equal to L1, L2, and L3.represent an assortment of things.L contains T since D is the collection of database transactions and T is made up of a set of objects per transaction.If and only if the transaction T completes A, then A for each transaction A in L may be referred to as the item set.If a frequent itemset satisfies a minimal support criterion, which indicates how often it occurs in the dataset, it is deemed important.Frequently used for rapidly identifying frequently occurring itemsets, the Apriori algorithm generates candidate itemsets and eliminates those that fall below the support criterion.Itemset A is also referred to as the frequent itemset. If the support count of itemset A equals or surpasses the supplied support count, the provided support count is regarded as the minimum support count (minsup).A key indicator in association rule mining is the support count, which aids in the identification of frequently occurring itemsets.

The may eliminate uncommon itemsets and concentrate on those that show up often in database operations by establishing a minimum support count, or minsup.This enables us to find significant relationships and trends between the dataset's components. Setting a higher minimum support count will result in fewer itemsets being considered frequent, while a lower minimum support count will result in more itemsets being classified as frequent. Adjusting the minimum support count can help tailor the analysis to specific needs and goals.

The Apriori approach served as the model for the FP_Growth (Frequent Pattern Growth) method [6].The FP Growth technique detects common item sets by building a tree, called the FP-Tree [7].The FP_Growth procedure is more effective because to the FP-Tree concept.FP-Growth is the first efficient tree-based method for mining the frequently occurring item sets. [8].A well thought_out divide and conquer method is used to reduce the size of the resultant conditional FP_tree.The datasets need to be scanned twice for this.A simplified depiction of the transactions is the FP_tree.The potential combinatorial number of candidate item sets hinders FP_Growth and is not reduced by a compact representation [9].Moreover, because of the potentially enormous size of the resultant tree, the main memory is unable to accommodate the massive database structure [10].Consequently, the Ordered Frequent Itemsets Matrix (OFIM), a unique two_dimensional

array structure based on the FP_Growth algorithm, is used in the suggested technique.This innovative structure compresses a transactional database to provide an environment that is ideal for effective mining of frequently occurring itemsets.

The remaining content is arranged as follows:Related work is included in Section 2.In Section 3, the original FP_Growth algorithm is provided.Section 4 presents the Methodology. Section 5 provides a description of the experiments and discussions. Section 6 contains the conclusions.

2. RELATED WORK

Some of the current algorithms for mining frequent itemsets are presented in this section. There are numerous frequent itemsets mining algorithms given in [8][11][12].

On the basis of a linear table, they have presented a novel frequent itemset mining algorithm. The linear table has the capacity to store more shared data while requiring fewer scans of the original dataset [13].

A divide-and-conquer technique based on an FP-Growth Tree to build a node-tree structure that is first sorted so that the most prominent patterns are accessible throughout the tree development process [14]. It has put out a novel frequent pattern mining algorithm based on the FP- Growth idea that pulls out frequent patterns using bit matrices and linked list structures [15].

Distributed Frequent Pattern Analysis In Big Data is proposed in [16]. This study uses the FP growth algorithm to find common item sets in a database without the need for candidate generation, and incremental FP-Growth analysis is suggested to create the least redundant tree structure possible. As a result, the database will undergo fewer scans, which will lower latency. A brand-new mining algorithm for accurate pattern determination in massive amounts of data is proposed [17][18]. A combination of calculations based on Map, Map Reduce Frame, and Hadoop opensource implementation are suggested for this usage.

This study introduces FP-Growth algorithm optimization against a backdrop of cloud computing and computer big data [19]. A parallel mining algorithm is discussed in this work. The enhanced technique is utilized by each node machine to generate fragmentary frequent itemsets via parallel mining. Subsequently, all frequent itemsets are retrieved through summarization [20]. Following the extraction of transaction databases in accordance with each frequent item, a corresponding projection database is generated for each such item.

Signature-based Tree for Finding Frequent Itemsets in [21]. In this study, the authors suggest a brand-new tree-based structure that places a stronger emphasis on transactions than itemsets. As a result, the steer clear of the issue of support values that have an adverse effect on the tree that is produced. Numerous strategies have been proposed to attain frequent item sets mining, which is founded on the fp-growth algorithm, while also ensuring privacy, utility, and efficacy [22].

A framework for an intriguing association rule mining technique for big data that is incrementally parallel is provided. During the mining process, the suggested framework combines interestingness measures [23]. The suggested framework processes incremental data, which typically arrives at various intervals, allowing the user's critical knowledge to be explored solely through the processing of new data, rather than starting over from scratch.

In the present study, they propose incremental maximal frequent itemset mining techniques that, throughout the mining stage, take into account the subjective interestingness requirement [24]. The proposed framework is specifically engineered to incorporate incremental data, which typically arrives at varying intervals.

The main contribution of this study is a new method that enhances the performance of algorithms that operate on FP-trees by using OFIM. This new approach efficiently lowers the computational cost while increasing the mining efficiency of regular patterns. It is a significant development in data mining research since it has shown encouraging outcomes in a number of real-world applications.

3. FP-GROWTH ALGORITHM

Han, Pei, et al. proposed the FP_tree (frequent pattern tree) data structure in [9].A very condensed representation of all pertinent frequency information in the data set is the FP-tree. Each FP_tree route denotes a frequent itemset, and the nodes inside the path are arranged in decreasing order of the respective items' frequency. Overlapping itemsets in an FP-tree share the same prefix route, which is a tremendous benefit. As a result, the data set's information is heavily compressed. There is no need for candidate itemsets, and the data set just has to be scanned twice.Compared to conventional approaches, this effective structure enables quicker mining of common patterns, which makes it especially helpful for huge datasets. Additionally, FP-tree has been shown to outperform other algorithms in terms of both memory usage and runtime.

There is header table in an FP-tree.The nodes in its FP-tree are linked to by the nodes in the header table. In the header table, single items and their counts are arranged in decreasing order.Retrieving frequently occurring objects and their accompanying pathways in the FP_tree is made efficient by the header table. The process of mining frequently occurring itemsets in big datasets is accelerated by this structure. A sample data set is shown in Fig. 1a, and the FPtree created by that data set with minsup = 30% is shown in table 1.

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

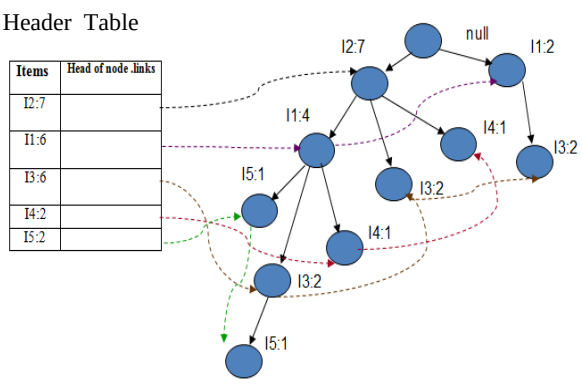

Figure 1 :An Example of FP-tree(*minsup*=50%).

Table 2 displays the generated frequent itemsets.

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. METHODOLOGY

4.1 The OFIM

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.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 remaining non_frequent candidate itemsets are thrown away.

In Table 3, the first column on the right lists the frequently occurring items in each transaction.

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,I5

You'll see that the transaction's frequent items are arranged in the same order as they appear in the list of frequent things. With reference to Table 1, I2,I1,I5 is the OFIL for transaction I1,I2,I5.An empty OFIM with N×M values is initially set to "0".List by list, the OFILs are read throughout the matrix creation process. Every list has elements extracted by the procedure. Subsequently, each item is added one at a time to the rows and matching columns of the matrix.For every list in the OFILs, the procedure is repeated.

After reading each of the OFILs, which are listed in Table 1, Table 2 provides a description of the whole OFIM.

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	I5	0

4.2 The Proposed Algorithm

The conventional FP_Growth method has many drawbacks, including slow FP_tree construction, a high number of frequent itemset discoveries, and FP_tree size increases that may not fit in main memory [25,26,].

The OFIM and a minsup threshold are inputs used in the process of finding frequently occurring itemsets.The suggested approach looks over each column in OFIM to determine the support of each unique item, and it uses the other (prior) columns to determine which node is the parent node of the current column.

In order to store the data about the current nodes and their parent nodes, we are unable to calculate frequent items one at a time using this method, which allows us to insert nodes into the FP-tree one level at a time.

Furthermore, no other item in the same row is frequent if an infrequent item, let's say x, is discovered in any column. In order to minimize execution time, the suggested method eliminates this row, significantly reducing the search space. The suggested approach saves money on frequent item scans by including OFIM into the tree building process.

Furthermore, things that appear more often are positioned higher on the FP-tree and are hence more likely to be shared. This suggests that the structure of an FP_tree is often rather compact.This reduces the size of the FP_tree and speeds up tree creation; as a result, performance is much better than that of the FP-Growth method. The detailed descriptions of the proposed algorithm are as follows :

inputs : A transaction dataset and a minsup threshold.

output :FP-tree .

1. Perform a single transaction database scan.Gather th e group of F often occurring things and their support ing.
   The list of ordered frequent items, or OFIL, is sorted F in descending order for support.
   In this stage, all infrequent itemsets are removed.

2. Establish OFIM.All sorted frequent items in the OF IL are input, item by item, into the respective colum ns for each row that corresponds to the OFIL.
3. Construct the T FP-tree root and designate it as "null". Let j be the column number in the OFIM.
4. For (j=1; j<=M; j++)

{
If j=1 Then Do
{
Gather the group of often occurring objects and thei r supports, then arrange the items in decreasing orde r of support count.Let the outcome be [f: n | OFIL], where n is the count and f is the first frequently occ urring item in OFIL.
Put these nodes into the FP_tree as the child nodes o f the root.Regular objects, f, are handled in the order that they appear.

}
Else Do
{

Make a comparison between the set of often occurring items a nd the columns that came before it, as well as the current colu mn (j) and its supporters.
The output should be [p, f: n | OFIL], where f is the current fre quent item in column (j) and p is the parent frequent item of th e preceding columns.
Using the node_link structure, connect the nodes that have the same item name.This entails adding [f: n] to the FP-tree as the child nodes of p and allowing their node-link to be connected to other nodes that have an item name via the node-link structure.
}

}

The proposed algorithm mined the FP_tree in the following w ay: build its conditional FP_tree and perform mining recursive ly on such a tree. The conditional pattern base is made up of t he set of prefix paths in the FP_tree that co_occur with the suf fix pattern, starting from each frequent length 1 pattern as an i nitial suffix pattern.The construction of the pattern is made po ssible by concatenating the suffix pattern with the common pa tterns generated by a conditional FP tree [14].

The FP_tree created by the recommended technique with the t ransactional data in Table 1 is shown in Figure 2.

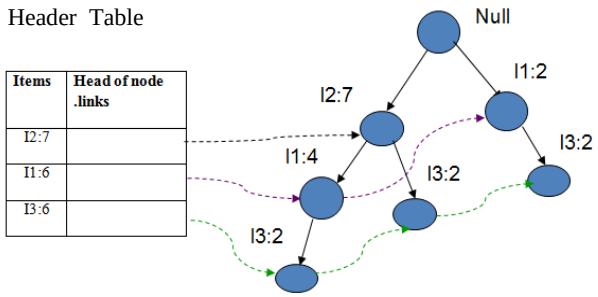

Figure 2: Construction of FP-tree using the proposed algorithm (*minsup*=50%).

The generated frequent itemsets are shown in Table 5.

Table 5: The generated frequent itemsets by the proposed algorithm.

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

5. RESULTS AND DISCUSSIONS

The put the suggested strategy into practice using datasets that included numbers.We assess the effectiveness of the propose d technique using real_world datasets obtained from the UCI Machine Learning Repository.The domains of KDD and data mining often make use of this collection of benchmark and rea l-world datasets [27].These datasets have been thoroughly exam ined by scholars and span a variety of topics.The evaluation's findings show how adaptable and reliable the suggested appro ach is in a variety of contexts.By testing the approach under v arious settings, the utilization of multiple datasets guarantees a thorough evaluation of its performance.The efficiency of the proposed method is evaluated and compared with the well-known FP_Growth algorithm in terms of the amount of freque nt item sets retrieved from the provided datasets and the time r equired to find frequent item sets.
In addition, we ran tests on other datasets with different featur es to make sure our conclusions were reliable.Our assessment provides a thorough examination of the recommended approa ch's efficacy by accounting for the effects of many factors and settings on its performance.The overall goal of our work is to provide a detailed efficiency and scalability comparison betwe en the proposed method and the FP-Growth algorithm.
We want to provide important insights into the advantages an d disadvantages of both algorithms in mining frequent item se ts by taking into account a number of variables and carrying o ut a number of in_depth trials.Every experiment is conducted on a laptop with an Intel(R) Core(TM) i3_2375M CPU operat ing at 1.50 GHz, 64GB of RAM, Windows 10 64_bit, and C+ + installed.Table 6 presents the statistical classification of the datasets used in this side-by-side analysis.
The study's datasets were carefully chosen to reflect a wide va riety of real_world events.They cover a range of sectors, popu lations, and regions to guarantee that our results are broadly a pplicable.

The statistical categorization in Table 6 provides a clear overview of the dataset characteristics, such as size, complexity, and distribution, allowing for a comprehensive

understanding of the experimental setup. This information is crucial for researchers to make informed decisions on data analysis methods and to interpret the results accurately. It also helps in identifying any potential biases or limitations in the study design.

Table 6: Characteristics of the test datasets

Datasets	Size	#Transactions
Data827	77.7MB	100004
QtyT40I10 D100K	44.9MB	105111

Comparative results between the suggested algorithm and the FP-Growth algorithm on the given datasets are shown in the following illustrations:-

5.1 First Experiment one

The dataset utilized in this experiment was Data8277.
It contains the modified transactions used as big data for the R C, TA, SA2, and DHB census night population counts in 2006 , 2013, and 2018.To pinpoint exactly which of the offered tec hniques performs better than the original FP-
Growth algorithm, a plethora of tests were carried out using v arious values of minsup.
Table 7 displays the results, which were sorted by the number of frequently occurring itemsets and the execution time neede d to find them for different minsup values (10%, 20%, 30%, a nd 50%).The findings demonstrate that, for all evaluated mins up values, the proposed method consistently beat the original FP-Growth algorithm in terms of accuracy and efficiency.
Table 7 presents a clear picture of how various minsup values affect the effectiveness of the recommended strategy.
It is evident that as the execution duration grows, less frequent itemsets are found as the minsup value rises.This offers insig htful information that will be useful in choosing a suitable min sup value for next implementations.Moreover, the results indi cate that when selecting a minsup value, a compromise betwe en the quantity of frequent itemsets and the execution time has to be made.
Achieving the best possible technique performance in practice requires making this trade-off.

Both algorithms' execution times and the quantity of frequently found itemsets generally decrease as minsup values rise.

Despite changing the minsup threshold number, it is found tha t the suggested technique takes less time to execute than the F P-Growth algorithm.
The performance of two algorithms for four distinct minsup th resholds 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 reason for this is because FP Growth requires a lot of con ditional sub_trees to be built before it can produce a lot of freq uent itemsets, which takes a lot of time and memory.
Thus, the suggested algorithm's effectiveness in cutting down on execution time may be linked to its efficient method of pro ducing frequent itemsets.Compared to FP_Growth, this impro vement enables quicker processing and less memory utilizatio n.

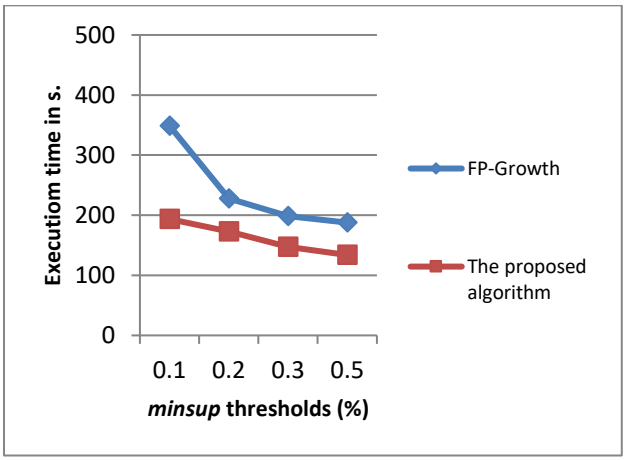

Figure 3: Comparing the results of the execution time and the *minsup* thresholds for the Data827 dataset.

5.2 Second Experiment two

For this experiment, QtyT40I10D100K dataset was used.
This collection has 105111 records with 3 properties that may be manipulated as big data.Table 8 displays the execution dur ation, the number of FP_Growth frequent item sets that were f ound, and the suggested method with different minsup criteria , including 10%, 20%, 30%, and 50%.

Table 8: Comparison results for the QtyT40I10D100K dataset with various *minsup* thresholds.

No.	*minsup*	Execution time per milliseconds (s)		# Discovered Frequent itemsets	
		FP-Growth	Proposed algorithm	FP-Growth	Proposed algorithm
1	10%	129.507	91.74	430	376
2	20%	122.979	78.55	225	197
3	30%	79.584	62.26	146	86
4	50%	68.787	56.38	106	72

Table 8 shows that although the suggested technique modifies the QtyT40I10D100K dataset's minsup threshold, it still exec utes almost equally long and produces less frequent itemsets a nd lessmining time than the FP-Growth approach.

Table 7: Comparison results for the Data827 dataset with various *minsup* thresholds.

No.	*minsup*	Execution time per milliseconds (s)		# Discovered Frequent itemsets	
		FP-Growth	Proposed algorithm	FP-Growth	Proposed algorithm
1	10%	348.699	193.56	216	146
2	20%	227.806	173.1	155	132
3	30%	198.36	147.23	134	106
4	50%	187.72	133.82	126	87

Compared to FP_Growth, the suggested approach scales much better.This is due to the fact that the frequency of frequent ite msets increases considerably when the minsup threshold decre

ases.Large candidate sets that the FP_Growth must manage re sult in highly costly pattern matching when many candidates a re found by searching the FP-tree.

The results of comparing the two algorithms' execution times using four different minsup thresholds are shown in Figure 4.

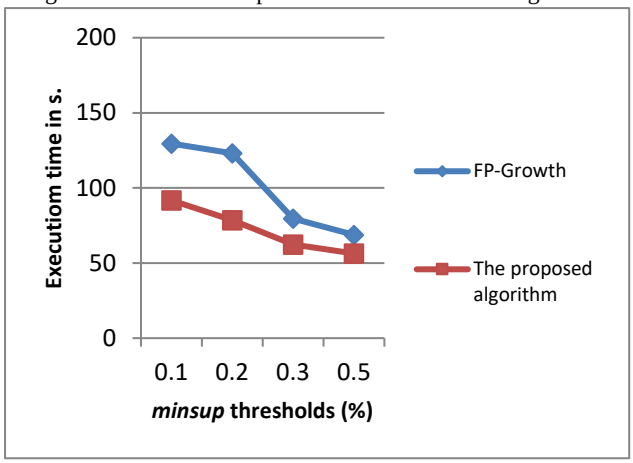

Figure 4: Comparing the results of the execution time and the *minsup* thresholds for the QtyT40I10D100K dataset.

6. CONCLUSION

In order to increase big data mining efficiency,An enhanced FP-Growth method for effective mining of the frequent itemsets is presented in this study. By building the OFIM using OFILs, the suggested approach increases the mining efficiency in the large data environment. As a result, the suggested technique shortens the time required to build an FP-tree by using OFIM to create a very compact FP-tree and then saving the expensive dataset scans for later mining operations. By using a pattern growth approach, it does not need expensive candidate generation. OFIM allows the FP-tree to be initialized immediately to the next level and saves traversal time for all elements.

Because the suggested method correctly removes infrequent items, the succeeding processes finish their duties more quickly and don't have to waste time analyzing unnecessary data. The relatively compact structure of the FP_tree, which employs OFIM to store just the often occurring elements in a frequency_descending sequence, is primarily responsible for the performance boost that the suggested approach achieves. The experimental findings demonstrate that, in terms of both mining time and the quantity of frequently found itemsets, the suggested approach outperforms the original FP_Growth algorithm.

7. REFERENCES

[1] S. P. Tamba, M. Sitanggang, B. C. Situmorang, G. L. Panjaitan, and M. Nababan. 2022. “Application of data mining to determine the level of fish sales in pt. trans retail with fp-growth method,” INFOKUM, pp. 905–913.

[2] T. Patil, R. Rana, and P. Singh. 2022. “Distributed frequent pattern analysis in big data.”International Research Journal of Modernization in Engineering Technology and Science ,pp.1-3.

[3] A. Ayu, A. P. Windarto, and D. Suhendro, 2021 .“Implementasi data mining dengan metode fp-growth terhadap data penjualan barang sebagai strategi penjualan pada cv. a & a copier,” Resolusi: Rekayasa Teknik Informatika dan Informasi, pp. 67–75.

[4] A. Siswandi, A. S. Sunge, and R. Y. Wulandari. 2018.“Analisa data mining dengan metode klasifikasi untuk produk cacat pada pt. shuangying international indonesia,” Jurnal SIGMA, pp. 153–156.

[5] B. Anwar, A. Ambiyar, and F. Fadhilah, 2023.“Application of the fp-growth method to determine drug sales patterns,” Sinkron: jurnal dan penelitian teknik informatika, pp. 405–414.

[6] 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.

[7] Almira, A., Suendri, S., & Ikhwan, A.2021. Implementasi Data Mining Menggunakan Algoritma Fp-Growth pada Analisis Pola Pencurian Daya Listrik. Jurnal Informatika Universitas Pamulang, 6(2), pp.442-448.

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

[9] 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.

[10] R. Krupali, D. Garg, and K. 2017. Kotecha, “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.

[11] M. Shridhar and M. 2017. Parmar, “Survey on association rule mining and its approaches,” Int J Comput Sci Eng, no. 3, pp. 129–135.

[12] H. Khanali and B. Vaziri. 2017. “A survey on improved algorithms for mining association rules,” Int. J. Comput. Appl, p. 8887.

[13] J. Lu, W. Xu, K. Zhou, and Z. Guo. 2023. “Frequent itemset mining algorithm based on linear table,” Journal of Database Management (JDM), pp. 1–21.

[14] M. Barkhan, R. Ramazani, and A. Kabani. 2022. “An algorithm to create sorted fp-growth tree for extracting association rules,” Research square, pp. 1–9.

[15] M. K. Sohrabi and M. H. HASANNEJAD. 2016. “Association rule mining using new fp-linked list algorithm,” Journal of Advances in Computer Research , pp. 23–34.

[16] T. Patil, R. Rana, and P. Singh. 2022. “Distributed frequent pattern analysis in big data.”International Research Journal of Modernization in Engineering Technology and Science ,pp.1-3.

[17] K. BHARATHI and D. B. DEVENDER. 2020. “Frequent itemset mining from big data using fp-growth algorithm,” Complexity International Journal (CIJ) , pp. 582–591.

[18] 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.

[19] 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.

[20] M. El Hadi Benelhadj, M. M. Deye, and Y. Slimani. 2023. “Signaturebased tree for finding frequent itemsets,” Journal of Communications Software and Systems, pp. 70–80.

[21] S. Bhise and S. Kale. 2017. “Effieient algorithms to find frequent itemset using data mining,” Int. Res. J. Eng. Technol., pp. 2645–2648.

[22] A. S. Alhegami and H. A. Alsaeedi. 2020. “A framework for incremental parallel mining of interesting association patterns for big data,” International Journal of Computing, pp. 106–117.

[23] H. A. Alsaeedi and A. S. Alhegami. 2022. “An incremental interesting maximal frequent itemset mining based on fp-growth algorithm,” Complexity.

[24] J.Han, J.Pei, & Y.Yin. 2000. "Mining frequent patterns without candidate generation." ACM sigmod record , pp. 1-12.

[25] F.Wei, & 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.

[26] R.Krupali, D.Garg , & 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.

[27] C. J. Merz, “Uci repository of machine learning databases,” URL: http://www. ics. uci. edu/˜ mlearn/MLRepository. html, 1998.