International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Developing an Efficient FP-Growth Algorithm using Ordered Frequent Itemsets Matrix for Big Data

Abdulkader Mohammed Abdulla Al-Badani*[1], Abdualmajed Ahmed Ghaleb Al-Khulaidi[1]

Submitted: 26/01/2024 **Revised**: 04/03/2024 **Accepted**: 12/03/2024

Abstract: Mining big data is difficult. Problems require an efficient algorithm and software computer for computation in big datasets The FP Growth Algorithm needs a lot of memory and requires a long time for computation and extract result. In this work, we propose modifications to the workings of the FP-Growth algorithm. The suggested algorithm will reduce the time in mining and decrease the number of frequently created items, yielding a significant reduction in decision-making in big datasets through our use of the proposed matrix OFIM instead of the tree used in those algorithms. The matrix OFIM allows for efficient storage and retrieval of frequent itemsets, resulting in faster computation and extraction of results compared to the traditional tree-based approach. Additionally, our algorithm optimizes memory usage by minimizing the number of frequently created items, further enhancing its performance in handling big datasets.

Keywords: *Big Data, Aprioiri Algorithm, FP-Growth Algorithm, Support Count.*

1. Introduction

Methods that can facilitate the collection of extremely large data sets are necessary as a result of the development of information technology in the modern era, which has generated significant advancements that greatly improve the amount of data that can be gathered and stored in a very large database [1]. Programs for big data analytics have significantly changed how individuals live their daily lives. As interest in data-driven decision making has grown, an extensive array of big data analytics solutions has been developed [2].Data mining is a technology that can be applied. The process of searching through selected data for intriguing patterns or information is known as data mining[3][4]. Data mining is also referred to as knowledge discovery in databases (KDD). KDD [5] encompasses both the collection of data and its utilization to identify relationships or patterns within vast data volumes.

Data from the physical world often comprises sets of objects, such as assemblages of products that were purchased together at a supermarket. For example, a frequent itemset refers to a collection of items that manifest a recurring pattern within a transaction dataset. The following definitions apply to the frequent itemset:-

Let L=L1, L2, L3,.In represent a collection of items. Since D is the collection of database transactions, and since each transaction T is made up of a set of objects, L contains T. For each transaction A contained in L, A can be referred to be the item set if and only if the transaction T completes A. The support count of an item set refers to the frequency with which the item set A appears in the transactions of database D. The itemset A is commonly known as the frequent itemset, and the provided support count is considered as the minimum support count (minsup) if the support count of itemset A is equal to or exceeds the specified support count. The support count is a crucial metric in association rule mining as it helps identify frequent itemsets. By setting a minimum support count (minsup), we can filter out infrequent itemsets and focus on those that occur frequently in the database transactions. This allows us to uncover meaningful associations and patterns among the items in the dataset.

The FP-Growth (Frequent Pattern Growth) method was created from the Apriori method [6]. The idea of creating a tree, known as the FP-Tree, is how the FP Growth approach finds common item sets [7]. The FP-Tree idea makes the FP-Growth process more efficient.The first effective tree-based approach for mining the frequent item sets is called FP-Growth. [8]. In order to reduce the size of the resulting conditional FP-tree, a divide and conquer strategy is implemented with careful consideration. For this, the datasets must be scanned twice. The FP-tree is a condensed representation of the transactions. FP-Growth is impeded by the prospective combinatorial quantity of candidate item sets, which a compact representation does not diminish [9]. Furthermore, the main memory is incapable of accommodating the large database structure as a consequence of the potentially immense size of the resulting tree [10]. As a result, the proposed method makes use of a novel, two-dimensional array structure based on the FP-Growth algorithm termed the Ordered Frequent Itemsets Matrix (OFIM). With the help of in this novel structure, a transactional database is compressed to produce an environment that is favorable for efficient mining of

[1]*Faculty of Computer Science & Information Systems, Sana'a University, Sana'a, Yemen.*
ORCID ID : 0009-0005-6772-370X
** Corresponding Author Email: Abdulkaderbadani@su.edu.ye*

frequent itemsets.

The structure of the remainder of the document is as follows: There is related material in Section 2. The FP-Growth algorithm is described in its original form in Section 3. In Section 4, an algorithm proposal is presented. In Section 5, an account of the experimental results and interrelated discussions is presented. Section 6 contains the concluding remarks.

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][13].

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 [14].

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 [15]. 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 [16].

Distributed Frequent Pattern Analysis In Big Data is proposed in [2]. 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 [18]. 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 [19]. 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 [20]. In this study, the authors suggest a brand-new tree-based structure that places a stronger emphasis on transactions than itemsets. As a result, we 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 [21].

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 [22]. 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 [23]. The proposed framework is specifically engineered to incorporate incremental data, which typically arrives at varying intervals.

3. FP-Growth Algorithm

FP-Growth is an alternative method for determining the most prevalent itemset in a data collection. FP-growth adopts a different paradigm from the one used by the Apriori algorithm. FP-Growth is an alternative method for determining the most frequent collection of data sets in a data set. The FP-Growth algorithm was created by modifying the Apriori algorithm. FP-growth is one method that frequently results in a mining itemset without a candidate Generation. It creates an extremely dense data structure (FP-tree) to condense the initial transaction database. This dense data structure allows for efficient and fast mining of frequent itemsets. By compressing the transaction database into the FP-tree, FP-growth eliminates the need for candidate generation, which can be computationally expensive in large datasets. Additionally, the FP-tree structure enables quick and effective pattern matching, making it a popular choice for frequent itemset mining tasks.

Mining all frequent itemsets just requires two dataset scans with the FP-Growth algorithm. The first scan counts how many times each item appears. The first FP-tree, which contains all of the frequency data from the original dataset, is built during the second scan. The FP-tree is then mined instead of the dataset. By mining the FP-tree, the algorithm can efficiently identify and extract all frequent itemsets without having to repeatedly scan the original dataset. This approach significantly reduces the computational time and resources required for frequent itemset mining. Additionally, the FP-Growth algorithm is particularly effective in handling large datasets with a high number of transactions, making it a popular choice in data mining tasks. The FP-Growth method for a transaction database's pseudo code is shown below [11].

An input consists of a minsup threshold and a transaction

database (DB).

Results: FP-tree.

Procedure:-

step 1: Extract the support count for each item from the transactional DB.

Step 2: Discard the item if item_id is less than the support. Step3: generate an I-list header table to store the frequently occurring item sets. The item sets should be arranged in a decreasing order according to support and node link.

Step 4:Create the FP-Growth tree initially. The algorithm begins by constructing an FP-Growth tree and assigns it the value "null" in the initial phase. Additionally, create a branch for each item in each transaction after reading it. If a prefix was previously shared by each node, increment the value by one; otherwise, create a new node.

Step 5: Each item in the header table is linked to its corresponding instance in the tree via a single link list, represented by dashed lines.

Step 6: Create the mine FP tree, also referred to as the FP-Growth tree.

The fp-tree is coupled with a header table. In decreasing order of frequency, the header table stores single objects and their counts.

An example of a transactional dataset is shown in Table 1, and the FP-tree produced from this dataset using the FP-Growth algorithm is shown in Figure 1.

Table 1. A dataset containing nine transactions is shown.

Header Table	
TID	List of items
D1	K1,K2,K5
D2	K2,K4
D3	K2,K3
D4	K1,K2,K4
D5	K1,K3
D6	K2,K3
D7	K1,K3
D8	K1,K2,K3,K5
D9	K1,K2,K3

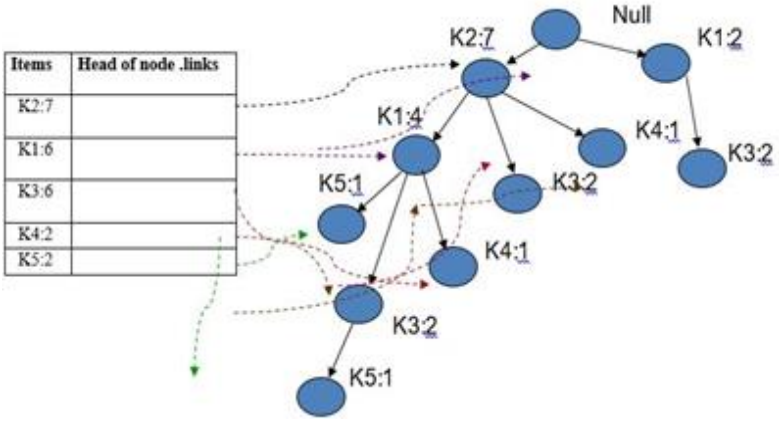

Fig 1. A minsup=50% FP-tree example.

Table 2 displays the derived frequent itemsets..

Table 2. lists the frequent item sets that the FP-Growth algorithm identified.

TID	Conditional FP-tree	Frequent itemsets
K5	<K2:2,K1:2>	{K2,K5:2}, {K1,K5:2}, {K2,K1,K5:2}
K4	<K:2>	{K2,K4:2}
K3	<K2:4,K1:2>,<K1:2>	{K2,K3:4},{K1,K3:4},{K2,K1,K3:2}
K1	<K2:4>	{K2,Kl:4}

4. The Proposed Algorithm

The basic FP-Growth algorithm can be applied to small data sets, but not to enormous data because building an FP-tree and finding many frequent itemsets take a lot of time. Therefore, and is affected by the growing FP-tree, which might not fit in the main memory. The OFIM and a minsup threshold are inputs used in the process of finding frequently occurring itemsets. The OFIM, or One-Itemset-at-a-Time Mining, approach addresses the memory limitation by two-dimensional array and updating it as new itemsets are discovered. This allows for efficient processing of large datasets without requiring excessive memory usage. Additionally, the minsup threshold determines the minimum support level that an itemset must meet to be considered frequent, allowing for customization based on specific data mining goals and requirements.

The transactional database is summarized using a two-dimensional array called the Ordered Frequent Itemset Matrix (OFIM), which contains all frequent itemsets and is sorted in support of decreasing order. The OFIM is composed of the longest number of often ordered commodities (Y) and the number of transactions (X). The OFIM allows for efficient and easy retrieval of frequent itemsets based on their support values. It provides a concise representation of the transactional database, making it easier to analyze and identify patterns in the data. Additionally, the OFIM can be used to generate association rules and make predictions about future transactions based on past patterns. The suggested algorithm searches the transactional dataset to provide a list of often occurring items, with the items listed in decreasing order of frequency. This arrangement is crucial since it will determine how OFIM is built. Ordered frequent itemsets lists (OFILs) are collections of candidate item sets for a transaction whose occurrence frequencies surpass the minsup threshold and are subsequently added to the list of frequent itemsets. OFILs play a significant role in constructing OFIM as they provide valuable insights into the most frequently occurring itemsets. By organizing the items in decreasing order of frequency, the algorithm ensures that the most relevant and important itemsets are prioritized for further analysis and decision-making. The final few candidate item sets are taken away. The sequence of frequently occurring items in each transaction is presented in the column labelled "Right-Most" of Table 3.

The "Right-Most" column in Table 3 provides valuable insights into the patterns and trends of frequently occurring items in each transaction. By analyzing this sequence, researchers can identify the most popular and commonly purchased items, helping businesses make informed decisions about product placement and marketing strategies. Additionally, this column allows for a better understanding of customer preferences and can be used to optimize inventory management systems. This listing of frequent items allows for easy identification of the most popular and frequently occurring items in each transaction. Analyzing this information can provide valuable insights into customer preferences and purchasing patterns, aiding in strategic decision-making and targeted marketing efforts.

Table 3. OFILs-based transactional dataset.

TID	List of items	OFILs.
D1	K1,K2,K5	K2,K1,K5
D2	K2,K4	K2,K4
D3	K2,K3	K2,K3
D4	K1,K2,K4	K2,K1,K4
D5	K1,K3	K1.K3
D6	K2,K3	K2,K3
D7	K1,K3	K1,K3

D8	K1,K2,K3,K5	K2,K1,K3,K5
D9	K1,K2,K3	K2,K1,K3

It is important to note that the frequent items of the transaction are presented in the same order that they appear in the inventory of frequent items. According to Table 3, the OFIL for transaction K1,K2,K5 is K2,K1,K5. For X and Y, an empty OFIM initialization value of "0" is present. Throughout the process of constructing the matrix, each OFIL is read individually. The method extracts items from each individual list. The elements are subsequently appended in a sequential manner to the rows and corresponding columns of the matrix. Each enumeration within the OFILs is subjected to this process once more. Upon perusing the entirety of the OFILs, as presented in Table 3, Table 4 delineates the OFIM in its entirety.

Table 4 provides a comprehensive representation of the OFIM after extracting and appending elements from each individual list in the OFILs. The resulting matrix showcases the sequential arrangement of items across rows and columns, with an initial empty initialization value of "0" for X and Y.

Table 4. The OFIM..

D1	K2	K1	K5	0
D2	K2	K4	0	0
D3	K2	K3	0	0
D4	K2	K1	K4	0
D5	K1	K3	0	0
D6	K2	K3	0	0
D7	K1	K3	0	0
D8	K2	K1	K3	K5
D9	K2	K1	K3	0

The proposed technique begins scanning from the final column and computes the support for each item in each column in OFIM, except for the sets of items that did not achieve support., we will discard them, and each item that has achieved support in the last column, we will compute the number of occurrences of the previous itemsets linked with it and similar ones in its records, and we consider them frequent itemsets, and we delete them from OFIM, In this way, we get rid of the tree and reduce the time and operations of generating the frequent itemsets, So, compared to the FP-Growth algorithm, performance is substantially better. The OFIM algorithm efficiently eliminates non-frequent itemsets by discarding them early on. By only considering itemsets that have achieved support, it reduces the computational burden and processing time required to generate frequent itemsets. This improved performance makes it a more efficient alternative to the FP-Growth algorithm. 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.

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 proposed algorithm Generation of frequent itemsets as follows : start from compute the support for each item from the last column of the OFIM. and the other (previous) column is used to distinguish the items of current column.The support is compute for the different items in the current column that achieves the support is taken as the frequent itemsets,and these rows are deleted from the OFIM, Then the previous column is moved from the current column. Repeat the previous steps for each column...

Table 5 displays the created frequent item sets.

Table 5. Displays the created frequent item sets.

Frequent itemsets	
{K2,K3:2}, {K2,K1,K3:2}	{K1,K3:2},

The FP-Growth method produces more frequent item sets than the suggested algorithm, according to Tables 1 and 2.

5. Results and Discussions

We implemented the method provided on databases containing numbers. The efficacy of the suggested methodology is evaluated by employing real-world datasets acquired from the UCI Machine Learning Repository. This compilation of benchmark and real-world datasets is frequently employed in the fields of KDD and data mining [24]. These datasets cover a wide range of domains and have been extensively studied by researchers. By using these datasets, we can ensure the generalizability and reliability of our results. Additionally, the UCI Machine Learning Repository provides detailed documentation and preprocessing guidelines for these datasets, which further enhances the validity of our evaluation. The effectiveness of the suggested approach is assessed and contrasted with the widely recognized FP-Growth algorithm with regards to the time needed to identify frequent item sets and the quantity of frequent item sets extracted from the given datasets. Furthermore, we conducted experiments on multiple datasets with varying characteristics to ensure the robustness of our findings. Our evaluation also takes into account the impact of different parameters and settings on the performance of the suggested approach, providing a comprehensive analysis of its effectiveness. Each experiment is executed on a laptop equipped with 64GB of RAM, Windows 10 64-bit, C++, and a 3.20 GHz Intel (R) coreTM i7-8700 processor. The datasets utilized for this side-by-side comparison are categorized statistically in Table 6. The datasets used in this study are carefully selected to represent a diverse range of real-world scenarios. They encompass different industries, demographics, and geographical locations to ensure the generalizability of our findings. 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.

Table 6. Test dataset characteristics .

Datasets	Size	#Transactions
Data8277	77.7MB	5029922
QtyT40I10D100K	44.9MB	3960457

The subsequent illustrations illustrate the comparative performance of the proposed algorithm and the FP-Growth algorithm on the specified datasets:-

5.1. *First Experiment one*

Data8277 was used for this experiment's dataset. It includes

transactions that are manipulated as big data for the 2006, 2013, and 2018 Censuses (RC, TA, SA2, DHB) census night population counts. A multitude of experiments were conducted utilizing different values of minsup and compared to the original FP-Growth algorithm in order to precisely determine the superior performance of the suggested technique. The obtained outcomes, categorized by the quantity of frequent itemsets and the execution time required to locate them for various minsup values (10%, 20%, 30%, and 50%), are presented in Table 7. The results in Table 7 clearly demonstrate the impact of different minsup values on the performance of the suggested technique. It can be observed that as the minsup value increases, the number of frequent itemsets decreases while the execution time increases. This provides valuable insights for selecting an appropriate minsup value in future implementations.

Table 7. Results of a comparison using different minsup criteria for the Data8277 dataset are shown.

# Discovered Frequent itemsets		Execution time per seconds (s)		minsup	No.
New algorithm	FP-Growth	New algorithm	FP-Growth		
14	97	100.918	194.184	10%	1
11	88	92.167	178.649	20%	2
10	84	91.26	178.022	30%	3
8	78	90.638	159.288	50%	4

As the value of minsup increases, the execution time of both methods and the quantity of identified frequent item sets tend to decrease.

The point at which the proposed method surpasses the FP-Growth algorithm is unequivocally demonstrated through the execution time comparisons of two algorithms for various minsup values (10%, 20%, and 30%) as shown in Figure 2. Constructing a substantial quantity of conditional sub-trees and consequential frequent item sets requires considerable time and memory. This is particularly evident when the minsup value is set to a higher percentage, such as 30%. The proposed method, on the other hand, efficiently handles the construction of conditional sub-trees and frequent item sets, resulting in significantly reduced execution time and memory usage.

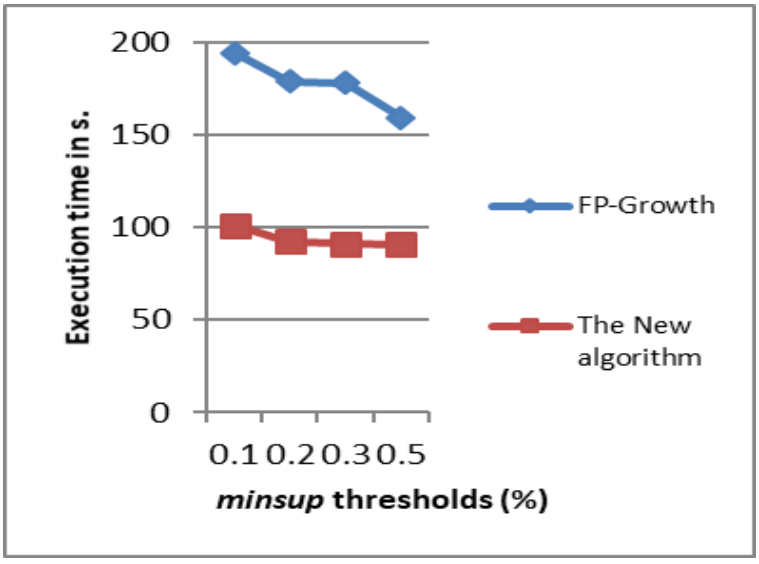

Fig. 2. A comparison of the Data8277 dataset's execution time and minsup threshold results.

5.2. Second Experiment two

The dataset QtyT40I10D100K was utilized for this experiment. This collection has 4 attributes and 3960457 records that can be manipulated as big data. Table 8 displays the FP-Growth and the proposed algorithm's execution times as well as the number of frequent itemsets that were discovered for various minsup criteria, including 10%, 20%, 30%, and 50%.

Table 8. Results of comparisons using different minsup thresholds for the QtyT40I10D100K dataset are shown.

# Discovered Frequent itemsets		Execution time per seconds (s)		minsup	
New algorithm	FP-Growth	New algorithm	FP-Growth		No.
147	434	302.164	980.964	10%	1
130	223	267.354	566.875	20%	2
126	145	151.296	386.94	30%	3
103	105	93.176	287.364	50%	4

According to Table 8, the suggested approach's execution time and the quantity of frequent item sets discovered are both less than those of the FP-Growth algorithm. Figure 3 displays the comparison findings for the QtyT40I10D100K dataset's minsup thresholds and execution times for the two algorithms.Figure 3 compares the performance of the two algorithms using 4 different minsup thresholds and displays the results.

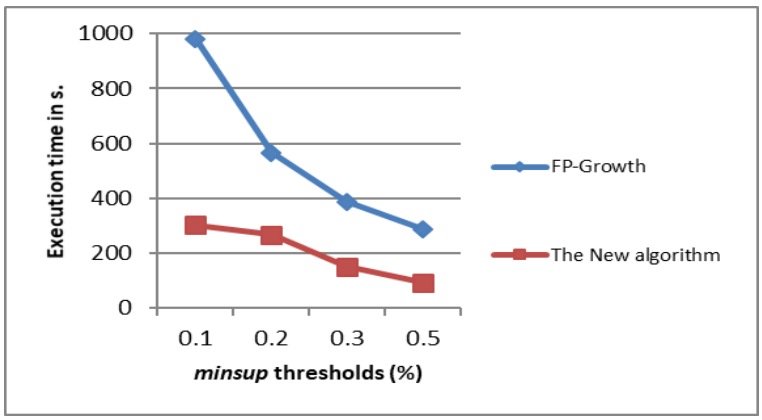

Fig. 3. : A comparison of the QtyT40I10D100K dataset's execution time and minsup threshold results.

6. Conclusion

To increase the effectiveness of big data mining,An enhanced FP-Growth method is suggested in this study for effective mining of frequent itemsets. The suggested approach increases the efficiency of mining in the big data environment by using OFILs to build the OFIM. As a result, the suggested technique creates fewer frequent itemsets after using OFIM to extract the set of frequent items.

In order to reduce execution time and memory consumption, the proposed method accurately deletes infrequently accessed objects. This deletion process is based on a thorough analysis of the object's usage patterns and relevance to the current system requirements. By removing these infrequently accessed objects, the method optimizes the overall performance and resource utilization of the system. The execution time of two algorithms for various minsup values in order to determine their efficacy. It unequivocally demonstrates how significantly the proposed algorithm outperforms the FP-Growth algorithm. Prior to generating a substantial quantity of frequent item sets, the FP-Growth algorithm is required to construct a considerable number of conditional sub-trees. This process is time-consuming and memory intensive.

In contrast, the proposed algorithm efficiently generates frequent item sets without the need for constructing conditional sub-trees. This not only reduces the execution time but also minimizes the memory usage, making it a more efficient and scalable solution compared to the FP-Growth algorithm. Additionally, the performance improvement of the proposed algorithm becomes more pronounced as the minsup values increase, further highlighting its superiority over FP-Growth.

References

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

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

[3] A. Ayu, A. P. Windarto, and D. Suhendro, “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, 2021, pp. 67–75.

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

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

[6] M. M. Hasan and S. Z. Mishu, "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, 2018, pp. 1–4.

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

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

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

[10] R. Krupali, D. Garg, and K. 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, 2017, pp. 929–934.

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

[12] R. Agrawal, R. Srikant et al., "Fast algorithms for mining association rules," in Proc. 20th int. conf. very large data bases, 1994, pp. 487–499.

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

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

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

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

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

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

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

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

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

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

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

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