ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

An Improved Efficient FP-Growth Algorithm Using FP-TDA Algorithm

Abdulkader Mohammed Abdulla Al-Badani[1*];
Redwan Abbas Hussein Al-Dilami[2]

[1]Faculty of Science and Engineering, Department of Computers, Aljazeera University, Ibb, Yemen.
[2]Faculty of Computing and IT, University of Science and Technology, Sana'a, Yemen.

Corresponding Author: Abdulkader Mohammed Abdulla Al-Badani*

Publication Date: 2025/12/16

Abstract: Mining association rules is one of the most important tasks for uncovering hidden patterns in datasets to aid organizations in making efficient decisions based on customer behavior and preference. This association rule identifies other possible connections that may exist beyond customer preference and behavior. Some of the most prominent algorithms used in this regard are Apriori algorithms and FP-Growth algorithms. They help to obtain sufficient information based on huge datasets. However, association rule mining using existing algorithms has encountered several problems. They consume a huge memory space for data processing. They need data searching for finding out the frequency of each item. They also establish rules with low efficiency. In overcoming such problems engendered by data mining association rules, a better efficient Fp-TDA algorithm using advanced FP-Growth algorithms has recently been developed. This algorithm uses Matrix Tree instead of a tree concept used in existing algorithms. This largely cuts down data processing time. This concept largely cuts down frequently generated data. This concept largely addresses precision. This largely enhances data processing. This concept largely addresses efficiency. This largely addresses data quality. This largely addresses data time. This concept largely addresses memory space. The Fp-TDA largely addresses problems in association rule mining. This concept largely addresses data processing. This concept largely addresses data quality. This concept largely addresses data memory

Keywords: *FP-Growth Algorithm, Aprioiri Algorithm, FP-Tree, Support Count, TDA.*

How to Cite: Abdulkader Mohammed Abdulla Al-Badani; Redwan Abbas Hussein Al-Dilami (2025) An Improved Efficient FP-Growth Algorithm Using FP-TDA Algorithm. *International Journal of Innovative Science and Research Technology*, 10(12), 802-810. https://doi.org/10.38124/ijisrt/25dec443

I. INTRODUCTION

Frequent pattern mining lies at the juncture of data science and real-world impact [1]. Researchers and professionals in the database field have become interested in data mining because it can pull out hidden patterns and meaningful insights from massive datasets, thus enabling wiser decisions in areas like financial forecasting, market strategy, and overall decision support [2]. The buzz grows, and optimism does too, as organizations see these benefits, thus fueling optimism about even smarter algorithms and fresher processes on the horizon.

Indeed, the hub of this discipline lies in FPM, which is all about detecting repeated patterns in massive streams of data [3]. The original Apriori algorithm by Agrawal and Srikant in 1994 became a catalyst for mining market basket data. It greatly assisted business people in uncovering the hidden relationship between products, thus helping marketers and planners of inventory levels [4]. Essentially, however, FPM is not only confined to the retail industry but also extends to health, finance, and cybersecurity, where translating data into action involves make-or-break decisions [5].

FPM methods aim to find frequent item sets and derive association rules from them, thereby facilitating their applicability to market analysis, healthcare diagnostics, and predictions of users' behaviors [6]. Innovations like the FP-Growth algorithm have pushed efficiency and scalability forward, allowing organizations to handle complex datasets more quickly and accurately. Typical FPM use cases include refining marketing strategies, improving patient care, and spotting anomalies in network security [7].

Classic techniques like the Apriori and FP-Growth have been the mainstay of frequent pattern mining, with the former at its foundation and the latter offering a significantly more scalable alternative [8].

This broad search with candidate generation involves a lot of computing and multiple database scans, hurting scalability for large datasets. In contrast, FP-Growth

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

outperforms Apriori's candidate-generation-and-test approach by using a compact tree that will encode everything needed to mine frequent item sets[9]. This avoids the generation of candidate item sets and cuts down execution time considerably. According to various research, FP-Growth can be faster than Apriori by a factor ranging from 1 to 300 depending on the minimum support threshold. By switching multiple database scans with a tree-based structure, FP-Growth makes the mining process smoother[10], increasing efficiency without harming the quality of the association rules obtained. It is particularly useful for large datasets, keeping performance high while memory usage is low and computational complexity minimized[11].

But the tree-based approach has its own challenges: constructing and traversing conditional subtrees can get extremely memory-intensive with large datasets, and the combinatorial explosion in nodes hurts scalability when working with very large collections of data[12].

To overcome these limitations, recent methods using hybrid strategies include FP-Linked Lists and MFP-Growth. However, they still have to face the challenges of redundant generation of subtrees and/or suboptimal memory utilization[13]. This points to the continuous demand for a memory-efficient, scalable variant of the FP-tree that retains the advantages of pattern-growth techniques yet reduces redundancy and additional overhead[14].

In this paper, we propose FP-TDA, a novel two-dimensional, matrix-based approach to replace FP-Tree[15]. The structure organizes transactional data into a matrix format and traverses column by column to make the mining of frequent item sets efficient without the need for recursive subtree construction. Our contributions are threefold:

- *Design:*

We describe the architecture of FP-TDA using OFILs in order to fill a sparse matrix, reducing redundancy and memory usage.

- *Efficiency:*

FP-TDA performs up to 50% faster runtimes and produces up to 75% less frequent item sets than that of FP-Growth and Apriori at a low minimum support, based on rigorous experiments on benchmark datasets such as UCI's Single Elder Home Monitoring and Metro Interstate Traffic Volume [16].

- *Practical Relevance:*

We validate FP-TDA's applicability in real-world scenarios, such as traffic accident analysis and market basket optimization, where rapid pattern extraction is critical.

The remainder of this paper is organized as follows: Section 2 Methods and materials, and Experiments and Results, Section 4 Discussion, Section 5 Conclusion, and Section 6.

II. METHODS AND MATERIALS

- *The Evolution, Challenges, and Research Gaps of Study*

The frequent pattern mining has gone a long way since Apriori, the brainchild of Agrawal and Srikant in 1994, which kicked off candidate generation-and-test strategies. In Apriori, a breadth-first sweep is employed: candidate item sets are generated, checked, and the infrequent ones are pruned away. Yet, it requires lots of database scans and the number of candidates blows up exponentially[17]. It thus cannot be applied on very large datasets. Though it laid the basis for market basket analysis, its scalability woes pushed the researchers towards pattern-growth methods, with FP-Growth at the forefront[18].

FP-Growth introduces a completely different paradigm, as it abandons candidate generation. Instead, it uses a compact FP-Tree-a prefix-tree that stores transactions efficiently-reducing database scans and significantly speeding up the process, sometimes by orders of magnitude compared to Apriori[19]. However, FP-Trees are quite memory-consuming for either dense datasets or when lots of items are distinct, as their memory consumption is growing exponentially with respect to the number of different items. Works like Goethals and Zaki (2003) pointed out such disadvantages, claiming that recursive subtree traversals, which are conditional, further increase computational overhead and reduce scalability for big data.

Hybrid approaches have emerged to address these issues. The FP-Linked Lists enhance the traversal of the FP-Tree by embedding linked-list structures in order to reduce the number of redundant subtrees generated. MFP-Growth combines the strengths of pattern growth with the efficiency of simpler candidate generation and applies memory-saving vertical data formats[20].

The superiority of FP-Growth in terms of execution time and memory usage was confirmed by comparative analyses between Apriori and FP-Growth, as examined in [21]. Nevertheless, FP-Growth's reliance on tree structures continues to be a barrier. For example, research on traffic accident datasets showed how well FP-Growth handled massive transactional data, but it also revealed scalability issues when building conditional sub-trees [22]. FP-Growth's usefulness in market basket optimization was demonstrated in parallel research in retail analytics, such as [23], but its memory-intensive operations made it impractical for real-time applications.

Recent developments have attempted to strike a balance between speed and memory efficiency, such as hash-table integration [24] and adjacency table-based optimization [25]. Although these techniques performed better in dense datasets, they had trouble with dynamic data updates[26]. Furthermore, empirical tests with programs like Weka verified that FP-Growth outperformed Apriori in terms of execution time, but they also revealed unsolved issues with reducing redundant pattern generation [27].

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

➢ *Problem Formulation*

FIM aims at extracting recurrent combinations from large datasets. In a nutshell, an itemset is any nonempty set of distinct items from the data, and the support of that itemset is the percentage of transactions containing it. A major knob controlling FIM is the minimum support threshold, minsup: it decides what is considered frequent[28]. It is a crucial threshold for performance: increasing minsup removes potentially interesting patterns, while decreasing minsup tends to trigger the explosion in the number of frequent itemsets and with it, computing costs.

FIM research has made extensive use of conventional methods like Apriori and FP-Growth. Both approaches, however, have serious memory inefficiencies and scalability issues. Apriori depends on exhaustive candidate generation, which lengthens processing times and necessitates numerous database scans [29]. Although FP-Growth does away with candidate generation, it suffers from complicated recursive subtree traversal and high memory consumption, especially in large, dense datasets. In light of these constraints, a novel frequent itemset mining algorithm that does not rely on tree-based structures is proposed in this study [30]. By doing this, it hopes to improve computational efficiency, optimize memory usage, and increase scalability, which will increase the FIM's suitability for large-scale real-world datasets[31].

In particular, this work aims to reduce memory overhead by doing away with tree structures, replace recursive FP-Trees with a dynamic two-dimensional array representation, and improve scalability to manage large high-dimensional datasets with low computational complexity[32]. Furthermore, the suggested algorithm is thoroughly tested using real-world datasets to compare it with well-known methods like Apriori and FP-Growth. Lastly, the study will show the practical applications of the algorithm[33].

➢ *FP-TDA Design*

To efficiently summarize transactional databases, all frequently occurring itemsets were included in a two-dimensional array (TDA). The support values of these item sets are arranged in descending order. The TDA is organized as a N×M matrix, where N is the total number of transactions and M is the maximum number of regularly ordered products. Important information about frequently used itemsets can be quickly and effectively extracted from the database because each cell in the array contains the support value for the corresponding itemset.

The decision-making processes in marketing and inventory management are greatly improved by this methodical approach to data organization, which enables quick information retrieval and analysis. Businesses can optimize their marketing strategies and inventory control by focusing on the most frequently purchased items and customizing their offerings to match customer preferences.

➢ *Mathematical Model of TDA and OFIL:*

- *Step 1: Identifying Frequent Itemsets*

For each itemset I in the database, compute its support(1):

Support (I) = Frequency of I in the datasett/ (Total number of transactions) (1)

Where:
I is the itemset.
Frequency of I is the number of transactions that contain all items in I.
Total number of transactions is the total count of transactions in the dataset.

If support (I) ≥minsup, then I is added to the list of frequent itemsets.

- *Step 2: Sorting Frequent Itemsets*

Sort the frequent itemsets by their support values in decreasing order(2):

$I_1, I_2,\ldots,I_k$

Where

$\text{supp}((I_1) \geq \text{supp}(I_2) \geq \cdots \geq \text{supp}(I_k)$ (2)

These sorted itemsets form the Ordered Frequent Itemset List (OFIL).

- *Step 3: Constructing the Two-Dimensional Array (TDA)*

Construct the matrix M with dimensions N×k, where N is the number of transactions and k is the number of frequent itemsets in OFIL. Each entry Mij represents the occurrence of the itemset Ij in transaction Ti. The matrix is filled as follows (3):

Mij=∑_0^1▒ _otherwise^(if itemset I_(j) is present in transaction T_(j.)) (3)

The columns of the matrix are ordered by the decreasing support of the itemsets.

- *Step 4: "Right-Most" Column Representation*

The "Right-Most" column in Table 3 refers to the sequence of itemsets that are most frequently occurring in each transaction. This can be represented as (4):

$R_i=\{I_j \backslash M_{ij}=1 \text{ for } j=1,2,\ldots,k\}$ (4)

Where Ri is the list of frequent itemsets (from the OFIL) that appear in transaction Ti.

➢ *The Purposed Algorithm*

In this Section, a new algorithm based on FP and the TDA In this Section, a new algorithm based on FP and the TDA structure is presented, which is called FP-TDA.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

- *Input:*
  Candidate itemset c

- *Output:*
  The support of candidate itemset c

The inputs are a minsup threshold and a database of transactions (3).

$$\text{Let } D=\{T\_1,T\_2,\ldots,T\_N\} \quad (5)$$

be a transaction database where each $Tk \subseteq I$ (set of all items).

The symbol D represents number transactions in database and the symbol N represents number transaction in database.

- *Frequent 1-Itemsets (F1):*
  Compute support for each item(6):

$$\text{Support}(i)=|\{T \in D \mid i \in T\}| \quad (6)$$

Extract frequent items (7):

$$f\_1=\{\{i\} \mid i \in I, \text{supp}(i) \geq \text{minsup}\} \quad (7)$$

- *Ordered Frequent Items List (OFIL):*
  Sort F1 in descending support order(8):

$$\text{OFIL}=|i\_1,i\_2,\ldots,i\_M| \text{ where } \text{supp}(i\_1) \geq \text{supp}(i\_2) \ldots \geq \text{supp}(i\_M) \quad (8)$$

- *Supporters (Tid Lists):*
  For each item $i \in$ OFIL, define its supporters (transaction IDs) (9):

$$T(i)=\{t \mid T\_t \in D, i \in T\_t \quad (9)$$

- *Two-Dimensional Array (TDA):*
  Construct a matrix where rows and columns correspond to items in OFIL. Each entry TDA[i][j] (for $i \neq j$) represents the intersection $T(i) \cap T(j)$, computed dynamically.

- *Frequent Itemset Generation:*
  Loop Over Columns (Descending Order):

For c=M downto 1:
Let f=c (current item in column c).
If c=1:
Output {f} (already in F1).

Else (c>1):

For each $r \in \{i\_1,i\_2,\ldots,i\_(c-1)\}$:

Compute supp({r,f})= $|T(r) \cap T(f)|$.

If supp({r,f}) $\geq$ minsup:

Add {r,f} to frequent itemsets.
Recursion: Treat {r,f} as a new "item" with supporters $T(r) \cap T(f)$, append to OFIL, and repeat the process for larger itemsets.

- *Output:*
  All frequent itemsets $F=\bigcup k \geq 1 Fk$, where Fk is the set of frequent k-itemsets.

The basic FP-Growth technique may work well for small data sets, but it is not suitable for large data sets because it takes a long time to generate an FP-tree and identify several frequently recurring item sets. Because of the growing FP-tree, it might not be able to fit in the main memory. The Fp-Tda approach can handle large data volumes without experiencing memory limitations by focusing on one itemset at a time. As a result, this method maximizes overall computational efficiency while also improving the ability to handle large datasets.

III. EXPERIMENTS AND RESULTS

➢ *Experimental Setup*

We used datasets from the UCI Machine Learning Repository, a reputable source of benchmark and real-world datasets commonly used by the data mining and knowledge discovery in databases (KDD) communities, to thoroughly assess the performance of the suggested technique [35]. These datasets are an essential tool that allows practitioners and scholars to methodically test and compare their approaches to established benchmarks.

This not only demonstrates the algorithm's resilience but also points out possible areas for improvement, opening the door for further study and advancement in the area. A laptop running 64-bit Windows 10 with Python, 32GB of RAM, and an Intel(R) Core(TM) i7-10850H CPU operating at 2.70GHz or 2.71GHz is used for each experiment.

Table 1 Test Dataset Characteristics

Datasets	Size	#Transactions
Single Elder Home Monitoring	42MB	75512
Metro Interstate Traffic Volume	3.472MB	48205
Metro Interstate Traffic Volume	3.472MB	48205

- *Experiment One*

The dataset monitored the living activities of an elderly person living alone with infrared sensors for movement, temperature sensors, and gas sensors. The collected data spanned from November 6, 2019, to February 13, 2020. Several experiments with different minsup values were

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

executed and compared with the original FP-Growth algorithm for performance evaluation, which demonstrated how well the proposed approach can perform as compared to the original one. The Single Elder Home Monitoring dataset was provided as a realistic setting to check the effectiveness of this algorithm in practice. These results showed improvement regarding accuracy and speed, therefore revealing the advantages of the modified strategy compared to traditional ones. Insights into the behavior and preferences of consumers could come about by analyzing the grocery dataset patterns, adding pragmatic value to this study. Table 2 shows the execution time for finding frequent itemsets and number of frequent itemsets by minsup values 0.003%, 0.005%, 0.007%, and 0.009%. The findings indicate a strong connection between the minimum support threshold and runtime: the smaller the minsup value, the more frequent itemsets are generated, and thus the analysis would take more time. This underlines the crucial issue of choosing the appropriate minsup that balances the processing performance with item granularity.

Table 2 Results of A Comparison Using Different Minsup Single Elder Home Monitoring Dataset are Shown

# Discovered Frequent itemsets	Execution time per seconds (s)	minsup	No.		
New Algorithm	FP-Growth	New Algorithm	FP-Growth		
74930	603798	20.986	23.390	0.003%	1
21076	318010	9.019	10.088	0.005%	2
3627	139023	3.451	6.561	0.007%	3
1866	104598	2.735	5.932	0.009%	4

As you raise the minsup threshold, you'll see fewer and fewer frequent itemsets pop up; both algorithms complete faster. Set the minsup cutoff wherever you like, and the new method is faster every time. Figure 1 compares their run times across four levels of minsup. The tendency is obvious: the higher the minsup, the fewer the number of itemsets and the shorter the execution times. This efficiency shows how well the suggested approach squeezed the processing power without compromising the integrity of the mining process. It clearly differentiates the suggested method from FP-Growth, particularly with large datasets, where FP-Growth tends to drag because of its lower speeds and memory limits. The new approach will enable practitioners to do quicker data analysis and draw valuable insights out of it with lesser processing overhead. FP-Growth, however, often requires considerable amounts of memory and time in building a number of conditional subtrees, thus blowing up the frequency count of the itemsets.

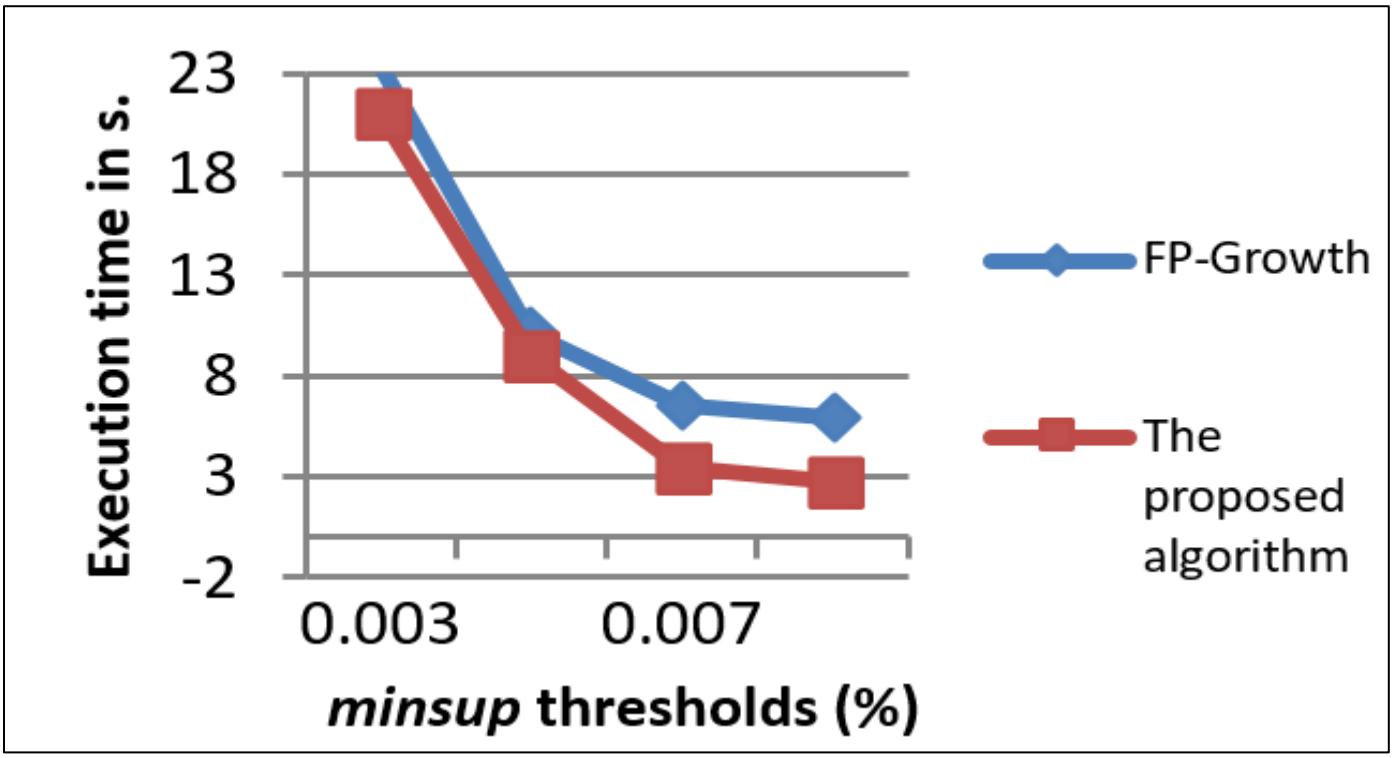

Fig 1 Comparing the Single Elder Home Monitoring Dataset's Execution Time and Minsup Threshold Results

- *Experiment Two*

The Metro Interstate Traffic Volume dataset was utilized in this experiment. This collection contains 48204 entries and 8 items. Among other aspects of Metro Interstate Traffic Volume, the collection contains data on habitat, odor, and cap shape. These features are frequently used in machine learning applications to classify mushrooms as dangerous or edible. These divisions are essential for maintaining security and disseminating information about mushroom collecting. Researchers can create algorithms that reliably forecast the

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

edibility of different mushroom species using the trends found in the dataset. The execution time, the number of FP-Growth frequent item sets discovered, and the recommended strategy with various minimal criteria (0.004%, 0.005%, 0.007%, and 0.009%) are shown in Table 3.

Table 3 Results of a Comparison Using Different Minsup Criteria for the Metro Interstate Traffic Volume Dataset are Shown

No.	minsup	Execution time per seconds (s)		# Discovered Frequent itemsets	
		FP-Growth	New algorithm	FP-Growth	New Algorithm
1	0.004%	10.296	4.496	313294	59073
2	0.005%	2.168	1.834	91937	2952
3	0.007%	1.177	0.911	44115	148
4	0.009%	0.971	0.650	29857	8

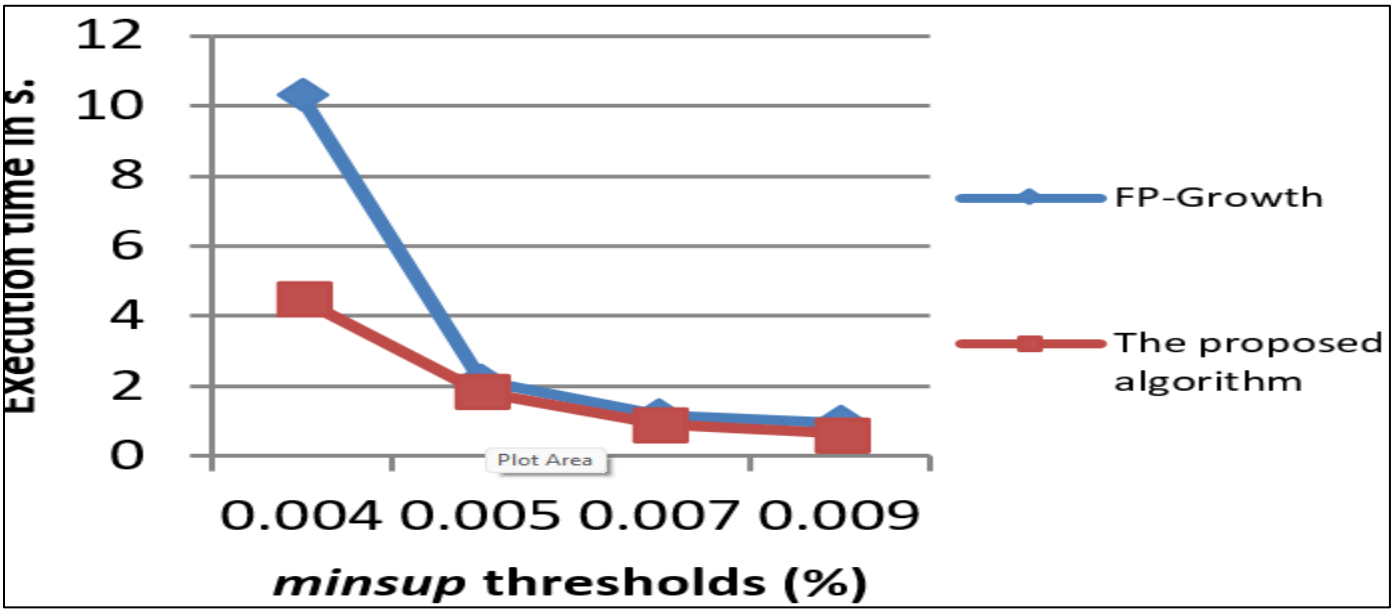

Fig 2 Contrasting the Outcomes of the Metro Interstate Traffic Volume Dataset's Minsup Thresholds and Execution Time.

This graph shows that the number of frequently generated itemsets and the execution time of three algorithms significantly decrease as the minsup threshold rises. This implies that by lowering the number of itemsets that must be taken into account, raising the minsup threshold may greatly enhance the algorithms' performance. As a result, practitioners can modify the minsup parameter to strike a balance between the amount of processing power required and the thoroughness of the itemsets discovered. This improvement is crucial in large datasets where processing time and memory usage are important factors. It also suggests that a higher minsup threshold for data mining tasks may result in more manageable outcomes and quicker execution times.

IV. DISCUSSION

Two different databases with different support values were used to compare the original [1] and suggested algorithms. The results of this study revealed significant differences between the values that each database produced, with the first database consistently producing much larger result values than its counterpart.

When the support threshold was set at 0.005%, the execution time for the first database was 9.019 seconds, which can be used to specifically illustrate this point. Under the same circumstances, however, the second database showed a significantly shorter execution time of just 1.834 seconds. The first database's intrinsic complexity and volume of attributes, which outnumber the second database's in terms of both quantity and richness of attribute values, are the main causes of this striking disparity. The longer execution time is a result of the need for more computation and processing when there are more attributes.

The first database makes things more difficult by adding a layer of complexity to queries and operations. The algorithms must navigate a more complicated structure and carry out more intricate calculations, which inevitably take longer. When compared to the second database's simpler setup, this additional complexity not only slows things down but also helps push the final results to larger values.

In other words, the difference in execution time and magnitude of results between the two databases points out how much the number of attributes and the general operation complexity affect the performance of the algorithm. The underlying message here is that, while one is constructing the algorithm, the database architecture should be understood first because it reflects the efficiency and effectiveness of the data to be processed.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

Table 4 Comparison Average Results of the Execution Time for the New Algorithm and FP-Growth Algorithm**.**

Execution Time per second(s)		Dataset Name
The New Algorithm		The FP-Growth algorithm
9.047	Single Elder Home Monitoring	11.492
1.972	Metro Interstate Traffic Volume	3.653

When compared to the original algorithm, the proposed algorithm exhibits a significant reduction in execution time across two different databases. Specifically, the proposed algorithm reduced the average execution time for the first database, Single Elder Home Monitoring, from 11.492 seconds with the original algorithm to 9.047 seconds. Similarly, the average execution time for the second database, Metro Interstate Traffic Volume, was lowered from 3.653 seconds to 1.972 seconds by applying the recommended algorithm. These results show how effective the proposed algorithm is and suggest that it could be used more widely in scenarios where processing time reduction is essential. Consequently, the notable reduction in execution times for both databases validates the effectiveness and efficiency of the proposed algorithm.

This study has clearly indicated the new algorithm is outstanding compared to the old one on both databases. Importantly, its faster average runtime shows its efficiency and effectiveness. Such quicker processing signifies that the proposed approach deals with data tasks more deftly across different databases and points to its usefulness in real-world data processing scenarios. Besides speed, the analysis found higher accuracy with the new algorithm than with the original. With both faster execution and better accuracy, the proposed algorithm represents a formidable option for data processing in a wide range of practical contexts. In other words, a mix of efficiency and precision suggests that it can substantially raise data processing performance in a wide variety of applications.

Table 5 Comparison Average Results of the Discovered Frequent Itemsets for The New Algorithm and FP-Growth Algorithm.

Number of discovered frequent itemsets		Dataset Name
The New Algorithm		
25374.75	Single Elder Home Monitoring	Single Elder Home Monitoring
15545.25	Metro Interstate Traffic Volume	Metro Interstate Traffic Volume

It is evident how successful the proposed algorithm is at identifying and eliminating duplicate elements when comparing its performance to the original algorithm across two databases. In the first database, Single Elder Home Monitoring, the original algorithm discovered an average of 291357.25 duplicate elements, whereas the proposed algorithm only discovered 25374.75 duplicates. Similarly, in the second database, Metro Interstate Traffic Volume, the original algorithm found an average of 119800.75 duplicates, while the proposed algorithm reduced this number to 15545.25. These findings show that the number of duplicate elements found by the suggested algorithm has significantly decreased. As a result, the suggested algorithm turns out to be more effective and efficient at reducing redundancy in the datasets, improving processing efficiency and data integrity. The superiority of the suggested algorithm in duplicate detection and elimination tasks is highlighted by this comparative analysis.

Data quality and database processing efficiency could be greatly improved by putting the proposed algorithm into practice. The algorithm can significantly reduce the amount of redundant data in databases, according to empirical results. Decision-making and data analysis procedures can be streamlined by removing unnecessary components, resulting in more effective operations. Additionally, using this method is probably going to increase data management's productivity and accuracy. All things considered, the data points to the suggested approach as a workable way to enhance organizational performance and optimize database operations.

V. CONCLUSION

The FP-Growth and Apriori algorithms are two well-known methods that make up the association rule algorithm. The Apriori algorithm is very useful because it is easy to use and can find the best combinations of rules. The FP-Growth algorithm, on the other hand, is very hard to use. It needs to scan a lot of data and build a lot of conditional sub-trees to find frequent item sets, which takes a lot of time and memory. This study seeks to compare the efficacy of three algorithms—FP-TDA, FP-Growth, and Apriori—across diverse datasets to assess their scalability and efficiency. The research aims to furnish insights that will assist practitioners in choosing the most suitable algorithm for particular data mining tasks by evaluating the strengths and weaknesses of each method. This study presents an improved FP-Growth technique aimed at enhancing the efficiency of frequent itemset mining in extensive data environments. The suggested method uses Optimized Frequent Item List (OFIL) structures to make the FP-TDA. This makes mining more efficient and cuts down on the number of frequent itemsets that are made. As a result, the better FP-Growth method is a better way to get frequent itemsets, which makes it a useful tool for data mining.

We look at the execution times of the three methods for different minsup values to see how well each one works. The proposed technique clearly works better than the Apriori and FP-Growth algorithms. When the recommended method greatly cuts down on calculation time while still making accurate frequent itemsets, this performance benefit is very clear at lower minsup thresholds. It is therefore a better choice for large datasets that are often used in real-world applications.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

The FP-Growth method needs to build a lot of conditional sub-trees before it can make a lot of frequent item sets. This procedure takes a long time and a lot of memory. The suggested method, on the other hand, makes this process easier and faster by lowering the need for big tree structures. This helps you get insights faster and use resources better, which is important when you're analyzing data in real time.

REFERENCES

[1]. WU, Bo, et al. An efficient frequent patterns mining algorithm based on apriori algorithm and the FP-tree structure. In: 2008 Third International Conference on Convergence and Hybrid Information Technology. IEEE, 2008. p. 1099-1102.

[2]. RIADI, Imam, et al. Implementation of association rule using apriori algorithm and frequent pattern growth for inventory control. Jurnal Infotel, 2023, 15.4: 369-378.

[3]. XIAO, Ling. Research on Education Data Association Rule Mining Based on Optimized Apriori Algorithm. In: 2023 7th Asian Conference on Artificial Intelligence Technology (ACAIT). IEEE, 2023. p. 1047-1051.

[4]. YU, Chengqi; LIANG, Ying; ZHANG, Xinxin. Research on Apriori algorithm based on compression processing and hash table. In: Third International Conference on Machine Learning and Computer Application (ICMLCA 2022). SPIE, 2023. p. 606-611.

[5]. ZHU, Siying. Analyse vehicle–pedestrian crash severity at intersection with data mining techniques. International journal of crashworthiness, 2022, 27.5: 1374-1382.

[6]. HUAMAN LLANOS, Alex Alfredo, et al. Toward Enhanced Customer Transaction Insights: An Apriori Algorithm-based Analysis of Sales Patterns at University Industrial Corporation. International Journal of Advanced Computer Science & Applications, 2024, 15.2.

[7]. WANG, Senzhang; CAO, Jiannong; PHILIP, S. Yu. Deep learning for spatio-temporal data mining: A survey. IEEE transactions on knowledge and data engineering, 2020, 34.8: 3681-3700.

[8]. SALAM, Abu, et al. Pencarian Pola Asosiasi Untuk Penataan Barang Dengan Menggunakan Perbandingan Algoritma Apriori dan FP-Growth (Study Kasus Distro Epo Store Pemalang). Dinamik, 2018, 23.2: 57-65.

[9]. FEBRIANTO, Moch Dwi; SUPRIYANTO, Aji. Implementasi Algoritma Apriori Untuk Menentukan Pola Pembelian Produk. JURIKOM (Jurnal Ris. Komputer), 2022, 9.6: 2010.

[10]. HASAN, Md Mahamud; MISHU, Sadia Zaman. 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. p. 1-4.

[11]. WEI, Fei; XIANG, Laisheng. Improved frequent pattern mining algorithm based on FP-Tree. In: Proceedings of The Fourth International Conference on Information Science and Cloud Computing (ISCC2015). 2015. p. 18-19.

[12]. KRUPALI, Rana; GARG, Dweepna. Survey on the techniques of FP-growth tree for efficient frequent item-set mining. International Journal of Computer, 2017, 160.1: 40-43.

[13]. GARG, Ritu; GULIA, Preeti. Comparative study of frequent itemset mining algorithms Apriori and FP growth. International Journal of Computer Applications, 2015, 126.4.

[14]. AL-MAOLEGI, Mohammed; ARKOK, Bassam. An improved Apriori algorithm for association rules. arXiv preprint arXiv:1403.3948, 2014.

[15]. AL-MAOLEGI, Mohammed; ARKOK, Bassam. An improved Apriori algorithm for association rules. arXiv preprint arXiv:1403.3948, 2014.

[16]. GUPTA, Aditya; GUSAIN, Kunal; GOYAL, Lalit Mohan. Improved FP-Linked List Algorithm for Association Rule Mining. In: Advances in Computer and Computational Sciences: Proceedings of ICCCCS 2016, Volume 2. Springer Singapore, 2018. p. 559-569.

[17]. SUBA, Saravanan; CHRISTOPHER, T. An Improved and Efficient Method to Discover the Frequent Patterns from Targeted Patterns in Transactional dataset using tpiitr-fpmm. international journal of advanced research in computer Science, 2017, 8.9.

[18]. JIANG, Hao; HE, Xu. An Improved Algorithm for Frequent Itemsets Mining. In: 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD). IEEE, 2017. p. 314-317.

[19]. SIVANANTHAM, S., et al. Association Rule Mining Frequent-Pattern-Based Intrusion Detection in Network. Computer Systems Science & Engineering, 2023, 44.2.

[20]. ZHIYUAN, Liu. Research on Association Rule Mining Algorithm Based on FP-tree. In: 2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI). IEEE, 2024. p. 1179-1184.

[21]. ZHANG, Baohua. Optimization of FP-growth algorithm based on cloud computing and computer big data. International Journal of System Assurance Engineering and Management, 2021, 12.4: 853-863.

[22]. CHANG, Hong-Yi, et al. A novel incremental data mining algorithm based on fp-growth for big data. In: 2016 International Conference on Networking and Network Applications (NaNA). IEEE, 2016. p. 375-378.

[23]. LI, Ye-Bai, et al. Frequent pattern mining algorithm based on improved FP-tree. Jisuanji Yingyong/ Journal of Computer Applications, 2011, 31.1: 101-103.

[24]. CHEN, Jin Bo, et al. A log analysis technology based on FP-growth improved algorithm. In: 2021 International conference on artificial intelligence, big data and Algorithms (CAIBDA). IEEE, 2021. p. 219-223.

[25]. AQRA, Iyad, et al. Incremental algorithm for association rule mining under dynamic threshold. Applied Sciences, 2019, 9.24: 5398.

[26]. EL HADI BENELHADJ, Mohamed; DEYE, Mohamed Mahmoud; SLIMANI, Yahya. Signature-based Tree for Finding Frequent Itemsets. Journal of

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec443

Communications Software and Systems, 2023, 19.1: 70-80.
[27]. CHANDRA, B.; BHASKAR, Shalini. A novel approach for finding frequent itemsets in data stream. International journal of intelligent systems, 2013, 28.3: 217-241.
[28]. XUN, Yaling, et al. Incremental frequent itemsets mining based on frequent pattern tree and multi-scale. Expert Systems with Applications, 2021, 163: 113805.
[29]. SONG, Wei; LIU, Wenbo; LI, Jinhong. Dynamic FP-Tree Pruning for Concurrent Frequent Itemsets Mining. In: International Symposium on Intelligence Computation and Applications. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 111-120.
[30]. DONG, Peng; CHEN, Bo. new algorithm of maximum frequent itemsets based on FP-tree for mining multiple-level association rules. In: 2008 International Conference on Computer Science and Software Engineering. IEEE, 2008. p. 263-266.
[31]. LEUNG, Carson Kai-Sang; MACKINNON, Richard Kyle; TANBEER, Syed K. Fast algorithms for frequent itemset mining from uncertain data. In: 2014 IEEE International Conference on Data Mining. IEEE, 2014. p. 893-898.
[32]. WINARTI, Titin; INDRIYAWATI, Henny. Data Mining Modeling Feasibility Patterns of Graduates Ability With Stakeholder Needs Using Apriori Algorithm. International Journal of Information Technology and Business, 2023, 4.2: 55-60.
[33]. SIVAIAH, Borra; RAO, Ramisetty Rajeswara. A Novel Nodesets-Based Frequent Itemset Mining Algorithm for Big Data using MapReduce. International journal of electrical and computer engineering systems, 2023, 14.9: 1051-1058.
[34]. RUSTOGI, Swati; SHARMA, Manisha; MORWAL, Sudha. Improved parallel apriori algorithm for multi-cores. International Journal of Information Technology and Computer Science (IJITCS), 2017, 9.4: 18.
[35]. BLAKE, Catherine L. UCI repository of machine learning databases. http://www. ics. uci. edu/~ mlearn/MLRepository. html, 1998.