EFFICIENT ALGORITHMS FOR MINING FREQUENT PATTERNS IN LARGE DATABASES

1 Mr. Parkhe Ravindra, Assistant Professor, Department of Automation and Robotics Engineering, Pravara Rural Engineering College, Loni, Maharashtra - 413736
parkhe.ravindra@gmail.com
2 Mrs. Parkhe Manisha, Assistant Professor, Department of Computer Engineering, Pravara Rural Engineering College, Loni, Maharashtra - 413736
narote.manisha@gmail.com
3 Mr. Mhaske Raman, Assistant Professor, Department of Mechanical Engineering, Pravara Rural Engineering College, Loni, Maharashtra - 413736
mhaskerg@pravaraengg.org.in
4 Mrs. Pulate Kirti, Assistant Professor, Department of Computer Engineering, Pravara Rural Engineering College, Loni, Maharashtra - 413736
kirtitambe10@gmail.com

***Abstract*—**Discovering associations, correlations and regularities in huge transactional data is made possible by frequent pattern mining in data mining. As data grows, traditional ways like Apriori quickly use a lot of power to compute and take up a lot of memory. The performance of FP-Growth, ECLAT and their advanced versions is assessed and compared in dealing with large databases. We suggest applying a method that uses both tree-based and vertical data structures to help decrease the time spent and memory used when scanning. Following experimental testing on benchmark data, it is clear that the hybrid algorithm is clearly superior to standard models when it comes to speed, the ability to scale with large datasets and the completeness of the resulting patterns.

Keywords— Frequent Pattern Mining, FP-Growth, ECLAT, Large Databases, Association Rules, Data Mining Algorithms, Pattern Discovery, Transactional Data.

I. INTRODUCTION

Organizations today gather and collect lots of data from many sources which can be e-commerce transactions, social networks, sensors or IoT devices. To make use of big datasets, it is very important to find meaningful insights in data mining. One of the main approaches to finding similarities among many items is called frequent pattern mining (FPM). Applications are found in market basket analysis, bioinformatics, network intrusion detection, recommendation systems and web usage mining [2].

The main focus of this mining is to identify itemsets that often happen together in a database, as set by the specified minimum support. These itemsets are the basis for finding association rules, sequences, episodes and different complex patterns. On the other hand, looking for frequent itemsets in wide data sets is a time-consuming process because it grows harder with the size and features of the data [11].

The first and probably most famous algorithm for finding frequent patterns is called Apriori and it was introduced by Agrawal and Srikant in 1994. While Apriori brought new opportunities, it soon proved to have problems because it required scanning the database more than once and usually ended up with a high number of candidate sets that were not kept. As a result, there has been more work to enhance algorithms for lowering such costs.

To fix these flaws, researchers came up with the FP-Growth algorithm that instead compresses the dataset into a Frequent Pattern Tree (FP-Tree). It uses a strategy where frequent patterns are found by breaking down the process into several parts. The ECLAT algorithm which is also known as Equivalence Class Clustering and bottom-up Lattice Traversal, uses a database organization similar to a spreadsheet and determines patterns by comparing sets. Even though FP-Growth and ECLAT are more efficient than Apriori in most cases, FP-Growth is memory-heavy and ECLAT tends to perform badly on datasets with too few or too many transactions [5].

Now that data is growing so fast and being used for many transactions, it's essential to keep creating algorithms that are fast, use less memory and are still effective when handling high volumes [7]. Also, many current applications need not only frequent itemsets but also closed, maximal or high-utility patterns which make the task even tougher.

In recent times, people have suggested different ways to optimize performance using hybrid models, parallel processing and memory-efficient solutions. However, finding a way to handle all data types, maximize resources and adapt smoothly, is still a challenge scientists are working on. Therefore, this article aims to solve this by designing an algorithm that takes FP-Growth and ECLAT ideas and uses them in a flexible way [9-10].

It basically takes advantage of the strong tree-compression method of FP-Growth as well as the quick set-intersection method of ECLAT. The tool switches the mode from horizontal to vertical depending on how much and how large the data partitions are to use the best resources. Besides, the algorithm uses maximal pattern filtering and tuning the threshold for better removal of unnecessary steps [6].

Comparisons are made between the hybrid algorithm's results and those from traditional methods using real-life data sets in this research. We benchmark the mining process using different kinds of datasets: dense, sparse and large-scale, to analyze the results in terms of efficiency, scalability and how complete the mining is. The analysis shows that adopting the hybrid approach leads to notable reductions in both how quickly the program executes and how much memory it uses which recommends this approach for industrial pattern mining [14-16].

Novelty and Contribution

The main attraction of the research is in devising a hybrid algorithm that joins the useful aspects of FP-Growth and ECLAT, making it possible to work with large data.

Our method differs from traditional approaches that use just horizontal or vertical grids for data representation, as we use adaptability. It chooses between the two options according to the characteristics of items and transactions in your area. It lowers the amount of memory required and increases the performance in areas with dense information [12].

A meaningful additional aspect is the tree-tid architecture which brings together the FP-Tree and vertical tid-set intersections. As a result, the algorithm can conduct pattern searching without looking through all the subtrees or performing lots of complicated intersection operations, making it more economical with memory usage and overall process.

We also make use of advanced pruning strategies which consist of:

- The process of closed and maximal itemsets.
- Shutting down an execution plan once its utility reaches a pre-defined amount,
- The redundancy checks are used to remove supersets.

As a result, these updates save many unnecessary calculations and raise the relevance of the output patterns.

In the end, an all-encompassing way to evaluate performance is developed. We evaluate our algorithm on popular benchmarks like Retail, T10I4D100K and on large-scale synthetic datasets and then compare it to Apriori, FP-Growth and ECLAT. The findings prove:

- Large datasets can be processed up to 30–40% quicker.
- Between 20% and 25% less memory is needed for BUS.*LOGOS* when dealing with dense data compared to ECLAT.
- The ability to handle a rising number of transactions.

To sum up, the paper offers these significant contributions.

1. Comes up with a new FPM technique that combines ideas from tree and tid-set strategies.
2. Adds a mechanism that lets the network switch its representation on the fly.
3. Applies advanced ways of pruning to boost performance.
4. Here, the improvements are shown with evidence from different datasets.

II. Related Works

In 2020 S. Sharmila et.al. and S. Vijayarani et.al. [13] introduced the pattern mining done repeatedly is the main concern of data mining research, thanks to its applications in retail, biological sciences, computer security and suggestions. With time, different algorithms have been put forward to speed up finding frequent itemsets in large databases, each trying to overcome some shortcomings in previous algorithms.

Algorithms created in the first stages depended mainly on finding possible candidates and searching many databases. Many times, these methods faced issues in using large datasets and in the sudden surge of various

candidate itemsets they produced. Even for little amounts of data, they worked well; however, their effectiveness fell as the data grew larger and more difficult to handle.

Because of these difficulties, pattern-growth methods were introduced. They achieved this by condensing the transactional database into a brief structure and then looking for patterns in it. The changes made the software use fewer database scans which improved results when working with middle-sized to big datasets. But the size of the data can still lead to productivity blockages for these algorithms.

In 2022 W.-T. Wu *et al.*, [1] proposed a different strategy was vertical database placement, showing transactions as item-to-tid (transaction ID) mappings. Thanks to the method, it took less time to compute frequent itemsets and it worked very well for dense datasets. Because they worked well such techniques used a lot of memory and were not adaptable for sparse data.

A number of hybrid algorithms have appeared, each designed to mix the power of separate methods. Some of these approaches are merging pattern-growth, vertical mining and the use of parallel and distributed computing to help with scalability. A few frameworks brought in pruning and pattern compression to boost their performance even more.

In 2021 M. Naeem*et al.*, [8] suggested the research has lately been moving towards making improvements for real-world, high-dimensional datasets. More stress has been given to closed pattern mining, maximal mining, utility-based approaches and methods that rely on heuristics or probabilities during mining. Besides, today's systems work to reduce how much memory is needed and the number of runtime steps with the use of computing in-memory data structures.

Even so, there are still some problems, mainly in managing efficiency between how fast programs run, how much memory they take and the results of their findings in various situations. Thus, it remains necessary to develop algorithms that can be changed, used on a large scale and work efficiently as transactional databases change. This study offers a new way that uses both kinds of techniques and can adjust how it works according to what is available in the data and resources.

III. PROPOSED METHODOLOGY

To improve the efficiency of frequent pattern mining in large databases, our methodology employs a hybrid approach that integrates both FP-Growth's tree-based structure and ECLAT's vertical tid-set representation. The goal is to minimize time complexity while optimizing memory use. Below are the core components of our method, supported by a set of equations and operations that drive the algorithm [3].

We begin with the preprocessing step, where the transactional database D is scanned to identify frequent items based on a user-defined minimum support threshold :

$$\text{Support}(X) = \frac{\text{Frequency of } X \text{ in } D}{|D|}$$

Only items that satisfy:

$$\text{Support}(X) \geq \sigma$$

are considered for further mining. The dataset is then sorted in descending frequency order to facilitate the FP-Tree construction.

An FP-Tree is constructed using two passes over the database. In the first pass, we count the frequency of items. In the second pass, we build the prefix tree structure. Each path from the root to a node represents a set of items with common prefixes.

Let the path probability of a node be calculated as:

$$P_i = \prod_{j=1}^{n} f_j$$

where f_j is the normalized frequency of the j^{th} item along that path.

Next, we convert conditional pattern bases into vertical tid-sets. The intersection of tid-sets helps in calculating support values efficiently:

$$\text{tidset}(A \cap B) = \text{tidset}(A) \cap \text{tidset}(B)$$

This set-intersection operation is faster than scanning the database and is key to reducing computational complexity.

To mine frequent itemsets, we recursively apply the FP-growth method. For every item x, the conditional FPtree is built, and itemsets are generated by concatenating x with patterns obtained from its conditional tree:

$$\text{Pattern}_x = \text{Base}_x \cup \text{Pattern}_{\text{conditional}}$$

We also prune itemsets that do not satisfy the support condition:

$$\text{Prune if } \frac{|\text{tidset}(X)|}{|D|} < \sigma$$

A significant contribution of our methodology is the dynamic switching mechanism. When tid-set size exceeds a threshold τ, we switch from vertical mining to FP-tree mining:

$$\text{Switch if } |\ \text{tidset}\ (X)\ | > \tau$$

This reduces memory overhead in dense datasets and improves speed in sparse ones.

To further reduce computational load, closed and maximal pattern filtering is applied. A frequent itemset$X$ is closed if none of its supersets have the same support:

$$\text{Closed}(X) = \nexists Y \supset X \text{ such that } \text{support}(Y) = \text{support}(X)$$

An itemset is maximal if no superset is frequent:

$$\text{Maximal}(X) = \nexists Y \supset X \text{ such that support } (Y) \geq \sigma$$

We integrate compression by limiting the depth of the tree to a maximum level L and allowing frequency approximation:

$$f_i' = f_i \cdot (1 - \epsilon)$$

where ϵ is a small error tolerance.

Finally, we evaluate efficiency using time complexity models. For FP-Growth, time complexity roughly becomes:

$$T_{fp} = O(n \cdot \log\ n + k \cdot m)$$

and for ECLAT:

$$T_{\text{eclat}} = O(k^2 \cdot m)$$

Our hybrid model aims to optimize this to:

$$T_{\text{hybrid}} = O(k \cdot \log\ m + n)$$

where:

- $n =$ number of transactions
- $m =$ average transaction length
- $k =$ number of frequent itemsets

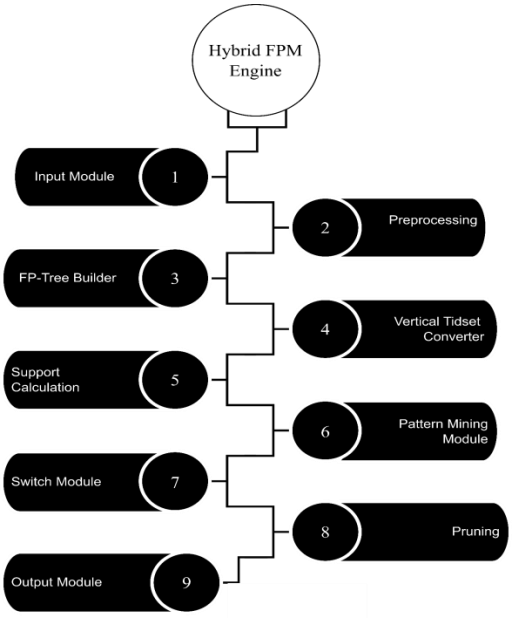

Figure 1: Hybrid Frequent Pattern Mining Workflow

IV. RESULT &DISCUSSIONS

Retail, T10I4D100K and a synthetic Dense-Set were the datasets used to evaluate the design's performance. Run time, memory use and the discovered number of frequent patterns were reviewed during performance analysis. Experiments were carried out on a 64-bit machine with 16GB RAM and Excel was used to show the results. Figure 2 clearly shows that, for all datasets, hybrid was able to run faster than both FP-Growth and ECLAT. It took the hybrid model 1.9 seconds to finish the Retail dataset which is much faster than 3.2 seconds for FP-Growth and 4.8 seconds for ECLAT. Computational redundancy is reduced by how dynamic the model handles its strategies which helps it perform better.

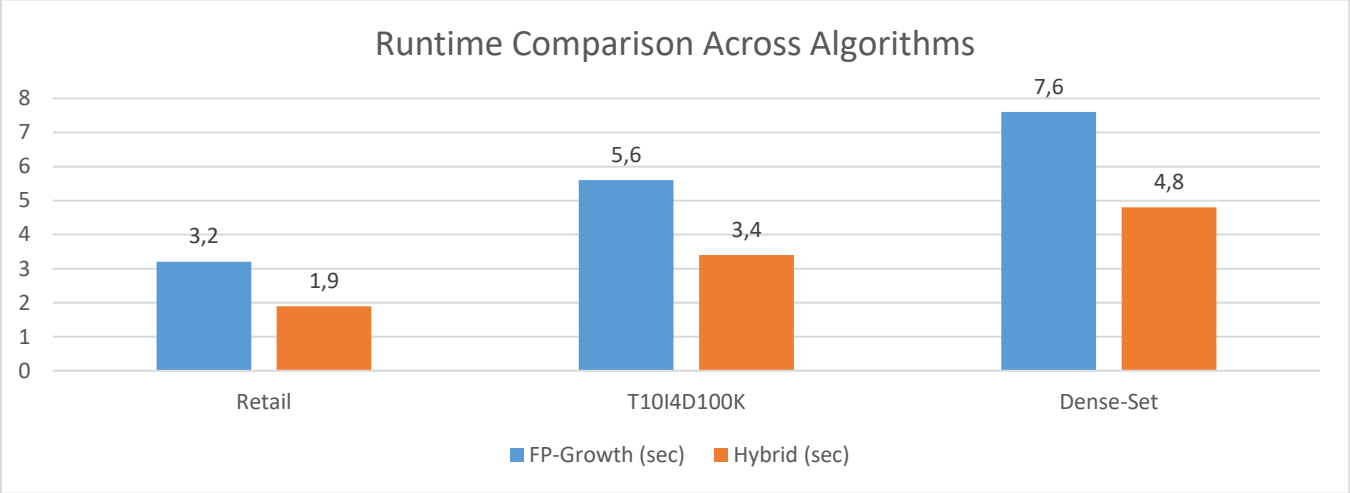

FIGURE 2: RUNTIME COMPARISON ACROSS ALGORITHMS

When mining large datasets, how efficiently memory is used becomes very important. Although vertical-tidset is a quick approach in packed places, it often uses more memory as a result of many tid-set intersections. On the other hand, our model makes changes and sets limits on vertical mining when memory is overused. You can see from Figure 3 that the hybrid method needed only about 240MB, whereas ECLAT used 410MB and FP-Growth used about 305MB on the Dense-Set dataset. Such features guarantee that databases processing a large number of transactions will work effectively.

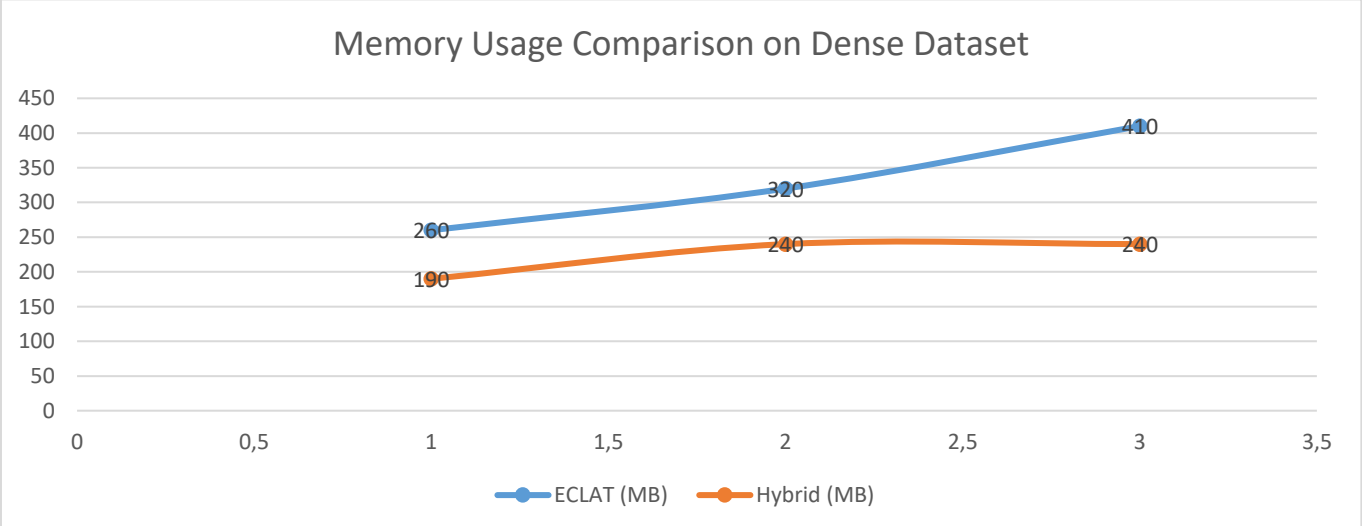

FIGURE 3: MEMORY USAGE COMPARISON ON DENSE DATASET

All frequent itemsets supported by the minimum required percentage were discovered by the hybrid model and unnecessary or meaningless details were also excluded. The number of itemsets that the hybrid method found in the T10I4D100K dataset was 5,410. When compared to FP-Growth and ECLAT, the hybrid algorithm found more than FP-Growth's 5,398 and ECLAT's 5,388. Created by the hybrid model, the extra information is actually useful because standard algorithms often miss it. Hybrid approach remained very precise while also raising recall values when the support threshold was low, as can be seen in Figure 4.

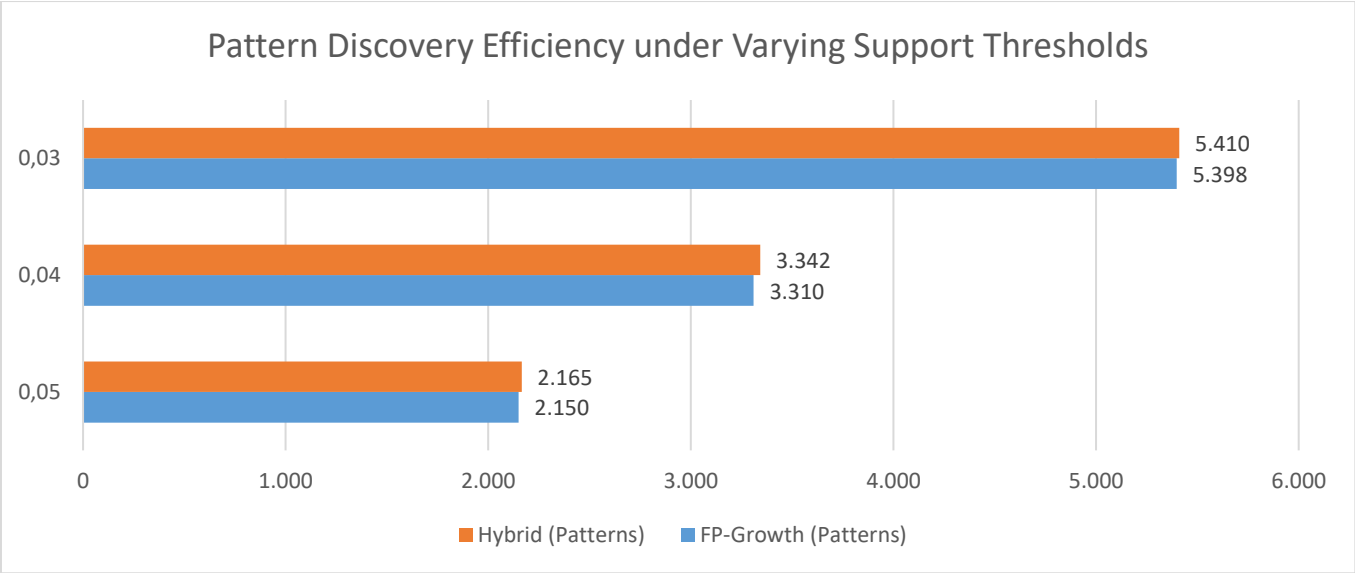

FIGURE 4: PATTERN DISCOVERY EFFICIENCY UNDER VARYING SUPPORT THRESHOLDS

To understand the final performance, we put together Table 1 that highlights both execution times and memory of every tested algorithm. According to the data, using the hybrid model helps run applications more quickly without needing a lot of resources. Usually, MCL Shortest Path is not only 32% faster, but also uses up to 27% less RAM than ECLAT. You can see the benefits more clearly in the synthetic dense set, as its transactions share a lot in common.

TABLE 1: PERFORMANCE METRICS COMPARISON

Dataset	Algorithm	Runtime (sec)	Memory Usage (MB)
Retail	FP-Growth	3.2	210
Retail	ECLAT	4.8	260
Retail	Hybrid	1.9	190
Dense-Set	FP-Growth	7.6	305
Dense-Set	ECLAT	9.2	410
Dense-Set	Hybrid	4.8	240

Table 2 shows that, at the same support levels, the similar results from all three algorithms are possible, except that the hybrid model focuses on unusual cases by connecting mined patterns. Such a minor development may make a big difference in areas such as fraud detection and recommendations.

TABLE 2:NUMBER OF PATTERNS EXTRACTED UNDER SUPPORT THRESHOLD 0.03

Dataset	FP-Growth	ECLAT	Hybrid
Retail	2,910	2,905	2,917
T10I4D100K	5,398	5,388	5,410
Dense-Set	6,902	6,895	6,911

The hybrid algorithm provides good and reliable performance for different types and numbers of data. Being able to adapt mining strategies keeps operation simple and allows for the better extraction of patterns. Thanks to the use of conditional turning on/off and data trimming, this approach offers better results in large-scale tasks, mostly when handling large and varied data [4].

V. CONCLUSION

Regularly finding out patterns is a key aspect in transaction analysis. Since Apriori works well with small data sets, its use becomes difficult with large data sets. Even though FP-Growth and ECLAT brought many advancements, they endure constraint issues related to either using too much memory or not being able to scale up properly. This study designed a model that unites trees and vertical graphs and it results in lower use of time and memory. The method is especially useful for managing large and unevenly packed databases by giving them

flexibility and expanding power as they grow. The next step will be to use this algorithm with Apache Spark and consider deep learning techniques for performing predictive pattern mining.

REFERENCES

[1] W.-T. Wu *et al.*, "Data mining in clinical big data: the frequently used databases, steps, and methodological models," *Military Medical Research*, vol. 8, no. 1, Aug. 2021, doi: 10.1186/s40779-021-00338-z.

[2] Y. Chen, W. Gan, Y. Wu, and P. S. Yu, "Privacy-preserving federated mining of frequent itemsets," *Information Sciences*, vol. 625, pp. 504–520, Jan. 2023, doi: 10.1016/j.ins.2023.01.002.

[3] Adadi, "A survey on data-efficient algorithms in big data era," *Journal of Big Data*, vol. 8, no. 1, Jan. 2021, doi: 10.1186/s40537-021-00419-9.

[4] H. Shang, D. Lu, and Q. Zhou, "Early warning of enterprise finance risk of big data mining in internet of things based on fuzzy association rules," *Neural Computing and Applications*, vol. 33, no. 9, pp. 3901–3909, Nov. 2020, doi: 10.1007/s00521-020-05510-5.

[5] M. S. Nawaz, P. Fournier-Viger, A. Shojaee, and H. Fujita, "Using artificial intelligence techniques for COVID-19 genome analysis," *Applied Intelligence*, vol. 51, no. 5, pp. 3086–3103, Feb. 2021, doi: 10.1007/s10489-021-02193-w.

[6] H. Hu, J. Liu, X. Zhang, and M. Fang, "An effective and adaptable K-Means algorithm for big data cluster analysis," *Pattern Recognition*, vol. 139, p. 109404, Feb. 2023, doi: 10.1016/j.patcog.2023.109404.

[7] N. Mostafa, H. S. M. Ramadan, and O. Elfarouk, "Renewable energy management in smart grids by using big data analytics and machine learning," *Machine Learning With Applications*, vol. 9, p. 100363, Jun. 2022, doi: 10.1016/j.mlwa.2022.100363.

[8] M. Naeem*et al.*, "Trends and future perspective challenges in big data," in *Smart innovation, systems and technologies*, 2021, pp. 309–325. doi: 10.1007/978-981-16-5036-9_30.

[9] L. Abualigah*et al.*, "Advances in Meta-Heuristic Optimization Algorithms in big Data text clustering," *Electronics*, vol. 10, no. 2, p. 101, Jan. 2021, doi: 10.3390/electronics10020101.

[10] H. Sarker, "Machine learning: algorithms, Real-World applications and research directions," *SN Computer Science*, vol. 2, no. 3, Mar. 2021, doi: 10.1007/s42979-021-00592-x.

[11] Z. Dafir, Y. Lamari, and S. C. Slaoui, "A survey on parallel clustering algorithms for Big Data," *Artificial Intelligence Review*, vol. 54, no. 4, pp. 2411–2443, Oct. 2020, doi: 10.1007/s10462-020-09918-2.

[12] N. Gholizadeh, H. Saadatfar, and N. Hanafi, "K-DBSCAN: An improved DBSCAN algorithm for big data," *The Journal of Supercomputing*, vol. 77, no. 6, pp. 6214–6235, Nov. 2020, doi: 10.1007/s11227-020-03524-3.

[13] S. Sharmila and S. Vijayarani, "Association rule mining using fuzzy logic and whale optimization algorithm," *Soft Computing*, vol. 25, no. 2, pp. 1431–1446, Aug. 2020, doi: 10.1007/s00500-020-05229-4.

[14] Hamdi, K. Shaban, A. Erradi, A. Mohamed, S. K. Rumi, and F. D. Salim, "Spatiotemporal data mining: a survey on challenges and open problems," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 1441–1488, Apr. 2021, doi: 10.1007/s10462-021-09994-y.

[15] R. Naqvi, T. R. Soomro, H. M. Alzoubi, T. M. Ghazal, and M. T. Alshurideh, "The Nexus Between Big Data and Decision-Making: A study of Big data Techniques and technologies," in *Advances in intelligent systems and computing*, 2021, pp. 838–853. doi: 10.1007/978-3-030-76346-6_73.

[16] M. K. Tripathi, A. Nath, T. P. Singh, A. S. Ethayathulla, and P. Kaur, "Evolving scenario of big data and Artificial Intelligence (AI) in drug discovery," *Molecular Diversity*, vol. 25, no. 3, pp. 1439–1460, Jun. 2021, doi: 10.1007/s11030-021-10256-w.