7th International Conference on Web Research
May 19-20, 2021; Tehran, IRAN ICWR2021

An Approach to Improve Apriori Algorithm for Extraction of Frequent Itemsets

Mohammad Javad Shayegan*, Parsa Asgari Namin
Department of Computer Engineering, University of Science and Culture, Tehran, Iran.
shayegan@usc.ac.ir, parsa.asgari.namin@gmail.com

***Abstract*— The amount of data generated today regarding volume, generation velocity, and variety is quite immense. This, in turn, has created a great challenge for scientists and researchers. To devise a solution, researchers have suggested a variety of schemes to help alleviate this problem. One of the suggested schemas is Association Rule Mining, and it is primarily focused on finding the associations in transaction-like data. To assist in finding such associations, Frequent Itemsets should be discovered first. Therefore, this research is a new approach to finding Frequent Itemsets and it is based on the Apriori algorithm and Apache Spark distributed platform. Further, we introduce an extended version of Apriori which tends to find Maximal Frequent Itemsets first to help speed up the mining process. The results and comparison to algorithms like YAFIM and HFIM and the original Apriori show the suggested algorithm outperforms them in dense datasets by an average of 38 percent.**

Keywords*— *Apache Spark, Frequent Patterns, Association Rule Mining, Apriori Algorithm.

I. Introduction

Today's data generation velocity has led to a state in which, although data is abundant, but it is no longer feasible to make use of this data with the traditional techniques. Data is needed to understand the past and predict the future. According to [1],[2],[3] [1-3], the solution to this problem is mining the available data. Data Mining consists of a variety of tools and approaches. Topics like Clustering, Classification, and Association Rule Mining are all part of the scope of Data Mining. As one of the topics within the discipline of Data Mining, Association Rule Mining is primarily about extracting frequent patterns like itemsets, sequences, and graphs. This method consists of combining techniques originally introduced to augment the Apriori algorithm and Apache Spark as a distributed processing solution. Since Association Rules are about discovering patterns inside data, they are in the interest of academia and the market alike. Therefore, this research endeavor aims to address that by finding frequent patterns that are needed to derive association rules.

The contents of this paper are organized in the order that the second chapter is mainly about reviewing relevant literature, the third chapter presenting the methodology of the suggested algorithm, while the fourth and fifth chapters present findings and conclude, respectively.

II. Literature Review

It is evident that since the original introduction of the Apriori algorithm [4] there are numerous attempts at enhancing and augmenting it. Algorithms like [5],[6],[7],[8] [5-8] each have contributed their visions to the original Apriori. However, none of the above researches were in the context of big data platforms. Therefore, the literature in the focus of this research is primarily about current big data platforms.

SRMINE [8], is an attempt at improving the original Apriori algorithm by employing a different perspective at the process of itemset counting in the original algorithm. The strategy presented in SRMINE starts the process of itemset counting in reverse since it is after Maximal Itemsets. The key idea is that, by finding the maximal itemsets, the time complexity and the space complexity of the algorithm decrease significantly. However, it is important to note that the approach taken in SRMINE is not big data. The research in [9] introduces an algorithm named MRApriori, which is a direct adaptation of Apriori in Apache's Hadoop platform, utilizing its underlying facilities for distributed processing across several nodes. SPC, DPC, and FPC are three algorithms presented in [10], which are variants of Apriori and based on Apache's Hadoop platform. SPC's approach is counting k-itemsets in one MapReduce phase, FPC counts fixed numbers of itemsets in each phase, and DPC counts the dynamic number of itemsets in each phase, determining the number by analyzing each phase's execution time. Dist-Eclat and BIGFIM are two algorithms introduced in [11], which both are based on the Apache Hadoop platform. Dist-Eclat is a distributed ECLAT based on the MapReduce approach of problem-solving. BIGFIM, on the other hand, is Apriori-based and employs a partitioning technique, and thus, it gains high scalability and the ability to process very big datasets. The first algorithm to use Apache Spark for adapting Apriori is YAFIM [12] which is a direct adaptation with no modifications. DFIMA [13] is another Spark-based method. However, DFIMA differs from Apriori by introducing a special pruning system based on mathematical matrices. R-Apriori [14] is a Spark-based algorithm that aims to adapt Apriori while adding enhancements by utilizing a different processing strategy and achieving a significant reduction in processing time. HFIM [15] is an effort to enhance Apriori by taking ideas from the ECLAT algorithm and infusing it with the Apriori principle while changing the itemset traversal scheme to include breadth-first approaches. Adaptive-miner [16] introduced by the researchers who had introduced R-Apriori earlier, is an algorithm that aims to infuse Apriori and Apache Spark while adding execution plan analysis to each phase of the processing that takes place. Lastly, the research in [17] is based on the combination of Apriori and Apache Spark and borrows ideas from the ECLAT algorithm while utilizing an alternative candidate counting strategy in each phase. Karimatabar and Fard [18] recently proposed another extension of Apriori by adding k-itemset stages. Fard and Namin [19] presented a review for the Apriori-based algorithms on big data.

III. Methodology

A. Dataset

The Apriori algorithm is a Frequent Itemset Mining algorithm, and it is usually used to extract frequent itemsets from transactional datasets. For this research, the sources of the datasets were primarily the websites in [16,17] since they provide datasets specific to Frequent itemset mining. For the sake of testing and verification, datasets like Retail, Chess were used from the sources mentioned above. In Table I, the characteristics of each dataset used for this research are summarized.

B. Algorithm

The suggested algorithm is implemented in Apache Spark distributed processing engine, and it aims to introduce an augmented itemset pruning strategy while reducing dataset scans by using the Apriori Principle along with an idea adopted from the research [13]. The idea in [13] is about stopping the counting process where the number of k-itemsets is lower than 'k'.

To better explain the algorithm, the assumptions about the data and the inherent nature of frequent itemset mining are presented here:

- The maximum length of transactions in all datasets is less than or equal to the number of distinct items.
- There is no repeated or duplicated item in any of the transactions.
- The maximum number of the generated itemsets equals the powerset of the set of distinct items in the dataset.
- The Apriori Property states that all of the subsets of a frequent itemset are frequent also.
- The more the value of Minsup is, the faster a Frequent Itemset Mining algorithm terminates.

With the assumptions above, the approach of the suggested algorithm is explained in the following:

1. During the Preprocessing phase, infrequent items are detected and pruned from the entire dataset. It is obvious that if a transaction should consist of infrequent items in its entirety, the transaction itself is pruned.
2. During the actual process phase (Fig.1), the number of maximum transaction length is divided by 4. The reason that this number is used Is due to numerous trials and errors. Going forward, AltrenativeApriori creates four ranges to organize its counting strategy. According to (1), The first range is $[0,q_1]$ in which

$$q_1 = \frac{\max\_transaction\_size}{4} \qquad (1)$$

According to (2), The second range is $[q_1,q_2]$ in which

$$q_2 = q_1 + q_1 \qquad (2)$$

The third in (3), $[q_2,q_3]$ where

$$q_3 = q_1 + q_2 \qquad (3)$$

and lastly in (4), $[q_3,q_4\ ]$ where

$$q_4 = q_1 + q_3 \qquad (4)$$

TABLE I. SUMMARIZATION OF THE CHARACTERISTICS OF THE DATASETS

Dataset Name	*Minimum Transaction Length*	*Maximum Transaction Length*	*No. of Transactions*
FoodMart	1	14	4,141
T25I10D10K	3	63	9,976
Retail	1	76	88,162
Chess	37	37	3,196

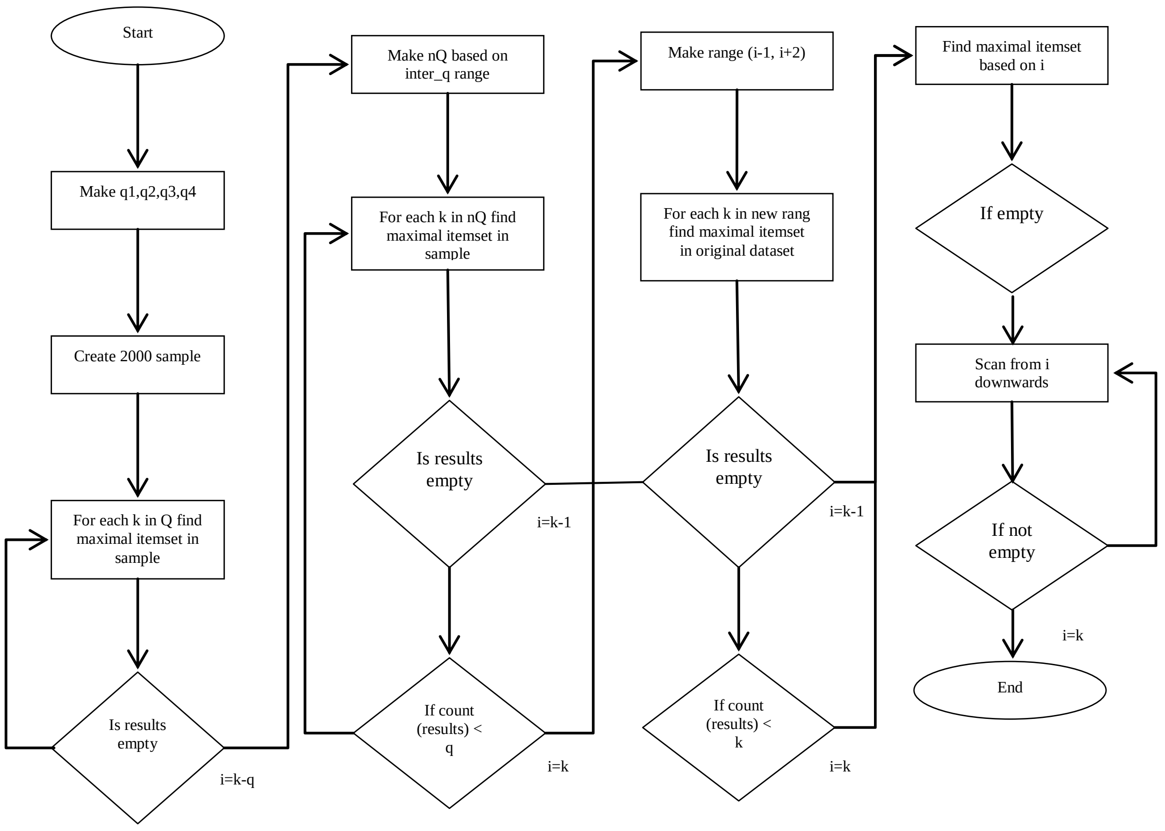

Fig. 1. Flowchart of the process phase

3. A random sample containing 2000 transactions is drawn. Then according to (5), the new *minsup* equals to

$$minsup = minsup * 2000 \quad (5)$$

4. The process begins. Traversing the ranges in descending order, each range right-edge is selected:
   - The transactions having more length than the selected number are selected.
   - All of the subsets of selected transactions are generated.
   - Using a map and reduce functions, the frequency of each subset (itemset) is calculated. Only those which meet the criteria of having a frequency value bigger than minsup are returned.
   - If the output of the calculation in the step above is 0, then from the inside of the current range which we're generating frequent itemsets for which the right-edge is selected.
5. The process continues for q_{n+1}. If the output is zero, the algorithm proceeds to step 6. If otherwise, the algorithm continues for q_{n+2}.
6. In this step, the inter-range levels are calculated to find the last k in which the number of frequent itemsets is not zero.
7. In this step, using the k found in step 6, the algorithm switches to the original dataset and tests the validity of the (k-1,k,k+1) levels. In other words, neighbors of the original k.

In the post-process phase, all of the subsets of the maximal itemsets found during the process phase are generated. Then, the difference between the items present in the maximal itemsets and the original distinct items is calculated. Further, the intermediate itemsets containing the difference set items are generated to have full coverage.

IV. Findings

To test and compare execution results, a computer system with these specifications was used:

- CPU: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz , 12 Core
- RAM: 31 Gigabyte
- HDD: 50 GB

- Spark Version: 2.4.2
- JDK Version: 11.0.2
- OS: Ubuntu 16.04.5 LTS
- Scala Version: 2.12.8

Observations in Fig.2 indicate that the suggested algorithm performs best on dense datasets. The suggested algorithm execution time on the Chess dataset is quite interesting concerning those of HFIM's and YAFIM's. It is evident that in the case of the Chess dataset, our algorithms

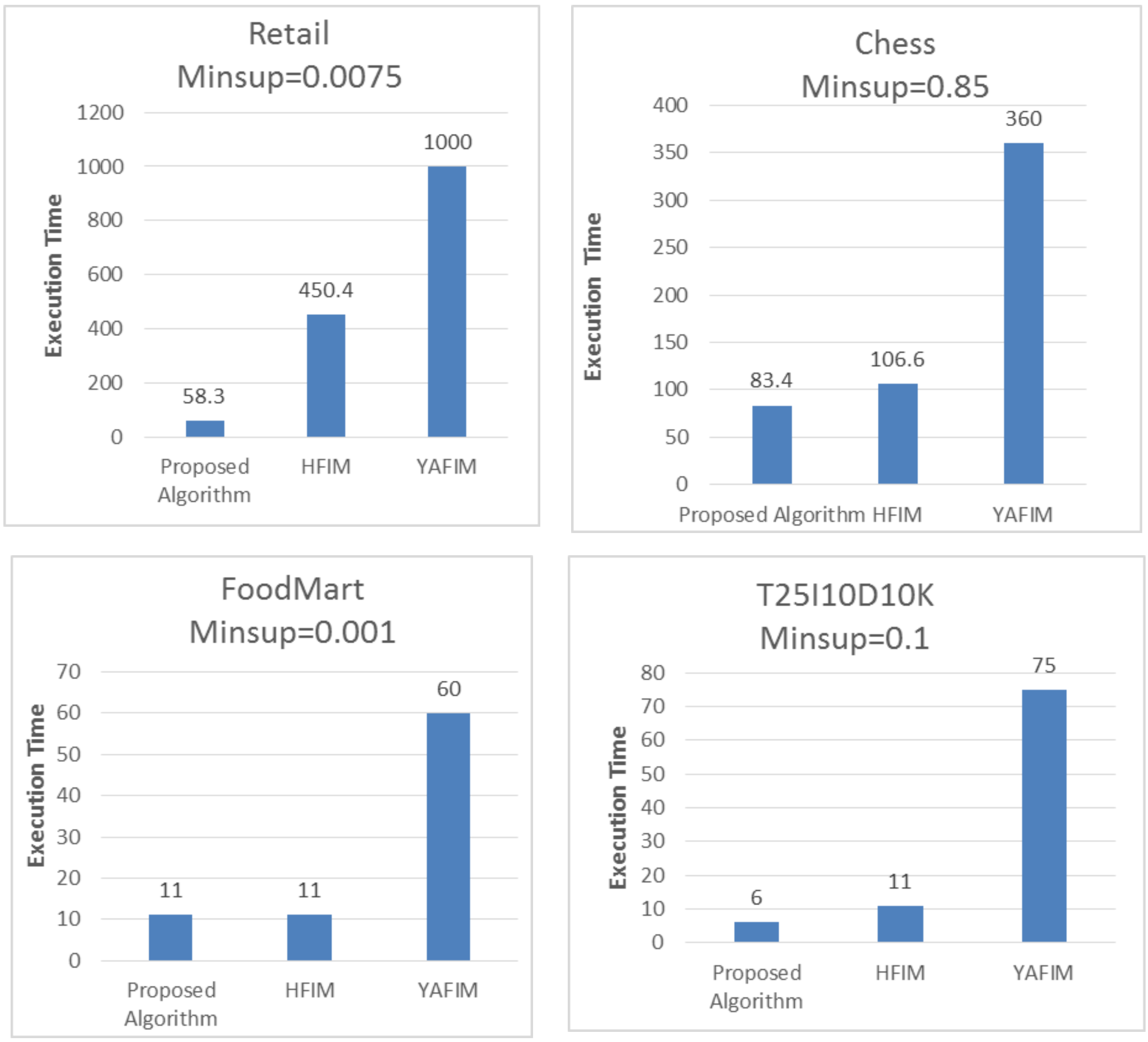

Fig. 2. Experiment Results on the datasets summarized in Table I

perform in 83 seconds while HFIM and YAFIM perform in 106 and 360 seconds, respectively. Note that YAFIM's execution time limit is set to 1000. Therefore, the YAFIM's process is terminated if it crosses the 1000 seconds limit. To explain further, the Chess dataset is considered dense, meaning that it has 37 distinct items while the average transaction length is 37. In other words, the presented algorithm counts all of the transactions in each and every iteration. However, Retail is different. The reason is that it is constrained by the minusp value specified at the beginning of the process; the pertaining frequent itemsets are found in the first k's. Therefore, our algorithm finds them faster because of the alternative itemset tree traversal approach it takes.

Table II demonstrates the relative percentages of execution times of the suggested approach and HFIM algorithms. It is important to note that the comparison excludes YAFIM since its execution time in the current hardware setup is not on par with that of our approach and HFIM. According to the execution times listed for each dataset, on average, the suggested algorithm performs about 38% faster than HFIM while having full coverage of the itemsets found.

V. Conclusion and Future Work

The endeavor in this research aimed to achieve a method for the extraction of frequent itemsets. Finding and extracting frequent itemsets in big data is a significant challenge since traditional methods like the original Apriori, FP-Growth, and ECLAT algorithms no longer apply. Therefore, new methods and tools are needed to make big data manageable. The Apache Spark platform is one of the numerous distributed processing tools that facilitate distributed processing while exposing facilities for adapting algorithms to distributed processing platforms. Numerous works are focusing on adapting the aforementioned algorithms to the Apache Spark platform while introducing modifications to help improve the existing methods.

TABLE II. Relative Reduction in Execution Time

Dataset	***Algorithm***			***Relative Reduction in Execution Time***
	Proposed Algorithm	***HFIM***	***YAFIM***	
Retail	58.3	450.4	1000	87
Chess	83.4	106.6	360	21
FoodMart	11	11	60	0
T25I10D10K	6	11	75	45
Average Execution Time Reduction				**38.25**

The methodology used in this research consists of gathering relevant datasets from credible sources and implementing modifications such as partitioning, sampling, heuristics and preprocessing approaches like pruning infrequent items from the whole dataset. Therefore, in the final iteration of the suggested algorithm, a preprocessing step involving pruning infrequent items, a sampling, and heuristics scheme along with using the Apriori Property together with a property adopted from [20] were used to achieve results. The key to the methodology used in the suggested algorithm is an interpretation of the Apriori Property. In other words, our algorithm aims to find the Maximal Itemsets first and foremost to reduce time and space complexity.

According to comparisons done between the suggested algorithm and HFIM and YAFIM algorithms on the hardware setting specified throughout the text, it outperforms the others by a 38% difference. It also guarantees full coverage of frequent itemsets since in the post-process phase, it scans the dataset for all the frequent itemsets that are not part of the maximal ones. For future work, the suggested algorithm will be examined on other datasets for a more in-depth evaluation.

References

[1] M. Cafaro and M. Pulimeno, "Frequent Itemset Mining," in Business and Consumer Analytics: New Ideas, 2019.

[2] A. Maske and B. Joglekar, "Survey on Frequent Item-Set Mining Approaches in Market Basket Analysis," in Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018, 2018, doi: 10.1109/ICCUBEA.2018.8697776.

[3] W. Gan, J. C.-W. Lin, H.-C. Chao, and J. Zhan, "Data mining in distributed environment: a survey," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 7, no. 6, 2017, doi: 10.1002/widm.1216.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in 94 Proceedings of the 20th International onference on Very Large Data Bases, 1994, doi: 10.1.1.40.6757.

[5] M. J. Zaki, "Scalable algorithms for association mining," IEEE Transactions on Knowledge and Data Engineering, 2000, doi: 10.1109/69.846291.

[6] R. J. Bayardo, "Efficiently mining long patterns from databases," SIGMOD Record, 1998, doi: 10.1145/276305.276313.

[7] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," IEEE Transactions on Knowledge and Data Engineering, 1996, doi: 10.1109/69.553164.

[8] S. Chourasia, R. Vishwakarma, N. Shukla, and M. Utmal, "An Innovative Approach for finding Frequent Item sets using Maximal Apriori and Fusion Process and Its Evaluation," International Journal of Computer Applications, vol. 40, no. 4, pp. 23–26, Feb. 2012, doi: 10.5120/5033-7184.

[9] O. Yahya, O. Hegazy, and E. Ezat, "An efficient implementation of Apriori algorithm based on Hadoop-Mapreduce model," Proc. of the, 2012.

[10] M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on MapReduce," in Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication - ICUIMC '12, 2012, p. 1, doi: 10.1145/2184751.2184842.

[11] K. Chavan, P. Kulkarni, P. Ghodekar, and S. N. Patil, "Frequent itemset mining for Big data," in 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 1365–1368, doi: 10.1109/ICGCIoT.2015.7380679.

[12] H. Qiu, R. Gu, C. Yuan, and Y. Huang, "YAFIM: A parallel frequent itemset mining algorithm with spark," in Proceedings - IEEE 28th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2014, 2014, pp. 1664–1671, doi: 10.1109/IPDPSW.2014.185.

[13] F. Gui et al., "A distributed frequent itemset mining algorithm based on Spark," 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), vol. 18, no. 4, pp. 271–275, 2015, doi: 10.1109/CSCWD.2015.7230970.

[14] S. Rathee, M. Kaul, and A. Kashyap, "R-Apriori: An Efficient Apriori based Algorithm on Spark," ACM, pp. 27–34, 2015, doi: 10.1145/2809890.2809893.

[15] K. K. Sethi and D. Ramesh, "HFIM: a Spark-based hybrid frequent itemset mining algorithm for big data processing," The Journal of

Supercomputing, vol. 73, no. 8, pp. 3652–3668, 2017, doi: 10.1007/s11227-017-1963-4.

[16] S. Rathee and A. Kashyap, "Adaptive-Miner: an efficient distributed association rule mining algorithm on Spark," Journal of Big Data, vol. 5, no. 1, p. 6, Feb. 2018, doi: 10.1186/s40537-018-0112-0.

[17] F. Gao, A. Khandelwal, and J. Liu, "Mining frequent itemsets using improved apriori on spark," in ACM International Conference Proceeding Series, 2019, doi: 10.1145/3325917.3325925.

[18] M. J. S. Fard and P. A. Namin, "Review of Apriori based Frequent Itemset Mining Solutions on Big Data," in 2020 6th International Conference on Web Research (ICWR), 2020, pp. 157-164: IEEE.

[19] N. Karimtabar and M. J. S. Fard, "An Extension of the Apriori Algorithm for Finding Frequent Items," in 2020 6th International Conference on Web Research (ICWR), 2020, pp. 330-334: IEEE.

[20] P. F. Viger, "SPMF - An Open-Source Data Mining Library." [Online]. Available: http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php.