A New Approach for Optimizing the Extraction of Association Rules

Bilal Bouaita
Ferhat Abbas Setif 1 University, Algeria
bilal.bouaita@univ-setif.dz
(corresponding author)

Abdesselem Beghriche
Ferhat Abbas Setif 1 University, Algeria
abdesselem_beghriche@univ-setif.dz

Akram Kout
MISC Laboratory, Ferhat Abbas Setif 1 University, Algeria
akram-kout@univ-setif.dz

Abdelouahab Moussaoui
Ferhat Abbas Setif 1 University, Algeria
abdelouahab.moussaoui@univ-setif.dz

Received: 28 January 2023 | Revised: 19 February 2023 | Accepted: 23 February 2023

ABSTRACT

Association rule methods are among the most used approaches for Knowledge Discovery in Databases (KDD), as they allow discovering and extracting hidden meaningful relationships between attributes or items in large datasets in the form of rules. Algorithms to extract these rules require considerable time and large memory spaces. This paper presents an algorithm that decomposes this complex problem into subproblems and processes items by category according to their support. Very frequent items and fairly frequent items are studied together. To evaluate the performance of the proposed algorithm, it was compared with Eclat and LCMFreq on two actual transactional databases. The experimental results showed that the proposed algorithm was faster in execution time and demonstrated its efficiency in memory consumption.

Keywords-KDD; data mining; association rules; frequent itemset

I. INTRODUCTION

Association rule methods [1, 2] are widely used in data mining [3, 4] which is the heart of Knowledge Discovery in Databases (KDD) [5, 6]. This approach was introduced to analyze the shopping cart or transaction data [2]. As each database transaction contains all items purchased by a customer, an association rule method identifies attributes or items that are often purchased together and discovers some meaningful dependencies and relationships between items sold for making predictions or decisions [7]. These relations are in the form of a rule: if X, then Y ($X \rightarrow Y$ (75%)), where X is the condition of the rule and Y is its conclusion. In addition, 75% support of the rule indicates that 75% of the customers who buy item X also buy item Y at the same time. These rules provide a decision support tool [8] in many areas, including the commercial sector. The association rule extraction is an essential two-step process that discovers frequent item set lists and generates association rules from these. The first step is the most costly in terms of execution time and memory space [9]. In databases that contain thousands of items, the larger the number of items, the larger the number of generated itemsets. This condition produces an explosion in the number of association rules. In a database that contains n items, 2^n-1 itemsets can be generated [10], producing a very large number of rules.

Each item or itemset is characterized by its support (Supp) [7]. The Apriori algorithm [11] minimizes calculations and the number of itemsets based on the support value. It only maintains frequent itemsets, where Supp is above a predefined threshold by the user (minimal support, MinSupp). MinSupp belongs to the interval [0,1], varies in decreasing order from 1 to 0, and depends on the database characteristics. The complexity of extracting frequent itemsets is sensitive to

MinSupp, as it increases rapidly when it decreases, given that more items exist. Therefore, the number of generated itemsets increases, which implies more complexity in terms of computation time and memory space. Several methods were developed to extract frequent itemsets and restrict search memory space and computation time. These methods can be grouped into two categories: the candidate generation approach and the pattern growth approach [12].

The candidate generation approach generates a list of candidate itemsets of size k to find the list of sets of frequent k-itemsets. Then these itemsets are generated to build a list of new candidate itemsets of size k+1. This procedure is repeated until the set, k+1-frequent itemsets, is empty. This method allows a decrease in the amount of search memory space required based on the antimonotone property. If an itemset is not frequent, then all its super-itemsets are also not frequent. The Apriori algorithm [11] is the most well-known algorithm of this family. Nevertheless, other algorithms that belong to the same category are DHP [13], DIC [14], Eclat [15], and Bitmap [16]. The main drawback of these algorithms is the scanning of the transactional database several times.

The pattern growth approach suggests removing the need to generate candidates to extract frequently occurring itemsets. This approach uses certain data structures, in particular trees, to accurately describe the transactional database. One of the most well-known algorithms in this family is the FP (Frequent Pattern)-Growth [17], which uses a specialized data structure called FP-Tree. The H-Mine [18], AFOPT [19], MFIs [20], and LCMFreq [21] are other methods that apply this approach. Recently proposed ideas employ prefix trees as a data structure to find frequent itemsets [22-26]. These approaches have the benefit of not performing candidate generation and avoiding multiple database scans. On the contrary, the data structures used in these algorithms result in significant memory consumption, and the development of these algorithms is more complex than in the candidate solution approach.

This study proposes a new algorithm based on Apriori because it is the foundation of other algorithms and is regarded as a reference. The proposed algorithm reduces the amount of memory space and time required for the discovery of frequent itemsets.

II. BASIC NOTIONS

Let $I = \{i_0, i_1,...,i_n\}$ be a set of n items, $T = \{t_1,...,t_m\}$ a database of m transactions, where each transaction t_i has a unique identifier and is a subset of items $t_i \subseteq I$.

- Definition 1: An itemset X is any item subset of I. For example $\{i_1\ i_5\ i_8\}$ is an itemset composed of three items.
- Definition 2: A k-itemset is a set of itemsets, where each itemset is composed of k items. For example, the itemset $\{i_1\ i_5\ i_8\}$ is considered an element of the 3-itemset.
- Definition 3: The support Supp(X) of an itemset X is the frequency of appearance of this itemset in the transactions of a database T or the ratio between the number of transactions t containing X ($X \subseteq t$) and the total number of transactions in the database T.

$$Supp(X) = \frac{|t_i \in T / X \subseteq t_i|}{|T|} \quad (1)$$

where $\|$ is the cardinal operator in the sets theory.

- Definition 4: An itemset X is frequent in a database T if its support is greater than or equal to a given threshold: Supp(X) ≥ MinSupp.
- Definition 5: An association rule is an implication expression among itemsets of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$.

III. THE PROPOSED ALGORITHM

A. Example

In a transaction database that contains n frequent items, the Apriori algorithm can generate a large number of itemsets k-itemset (a subset of items of size k), which is equal to all possible combinations C_n^k. For example, for n=100 items, 3921225 itemsets of size 4 can be generated. The proposed algorithm can reduce the number of generated k-itemsets by studying the items by category according to their support. To better understand the proposed algorithm, its principle of operation is illustrated through the following example: Table I shows the transaction base T, which contains 10 transactions and 14 items (a, b, c, d, e, f, g, h, m, n, p, x, y, z), where each transaction contains one or more items.

TABLE I. EXAMPLE OF TRANSACTION DATABASE

Transaction ID	Items
1	*a, b, f, n, x*
2	*a, n*
3	*b, c, d, f, g, h, m*
4	*b, c, g, h, m, n*
5	*a, b, f, z*
6	*a, f, g, y*
7	*a, b, c, f, g, n*
8	*a, c, d, e, h, y, n, p*
9	*a, f, n*
10	*a, d, e, g, h, y, p*

The support of each item is: Supp(a) = 0.8, Supp(f) = Supp(n) = 0.6, Supp(b) = Supp(g)= 0.5, Supp(c) = Supp(h)= 0.4, Supp(d) = Supp(y)= 0.3, Supp(e) = Supp(m) = Supp(p) = 0.2, Supp(x) = Supp(z)= 0.1.

The following parameters are used: $MaxSupp_1$ is the greatest support, $MinSupp_1$ is the first minimum support, $MinSupp$ is the minimum support, and a parameter $\alpha \in]0, 1[$. In this example, $MaxSupp_1$= 0.8, a = 0.72, $MinSupp_1$ = 0.35, and $MinSupp$= 0.2. Then, instead of processing all items that are greater than $MinSupp$ at the same time as the majority of the algorithms do, the items are considered by the group of intervals [$MaxSupp_i$, $MinSupp_i$], and the intersection between two successive intervals must be non-empty. The following equations are used to find the $MaxSupp_i$ and $MinSupp_i$:

$$MaxSupp_i = \alpha * MaxSupp_{i-1} \quad (2)$$

$$MinSupp_i = \alpha * MinSupp_{i-1} \quad (3)$$

Table II shows the items obtained in each interval. Table III shows the complexity of (C_n^k) to generate the 2- and 3-itemsets by the proposed and the Apriori algorithms:

TABLE II. EXAMPLE OF THE NEW ALGORITHM

Step (*i*)	$MaxSupp_i$	$MinSupp_i$	Items of each interval	Number of items
1	0.8	0.35	*a, f, n, b, g, c, h*	7
2	0.576	0.252	*b, g, c, h, d, y*	6
3	0.414	0.181	*c, h, d, y, e, m, p*	7
STOP *MinSupp* = 0.181 < 0.2				

TABLE III. COMPUTATIONAL COMPLEXITY

Algorithm	k=2	k=3
Proposed	$C_7^2+C_6^2+C_7^2$= 21+10+21=42	$C_7^3+C_6^3+C_7^{23}$= 35+20+35=90
Apriori	C_{12}^2=66	C_{12}^3=220

The Apriori algorithm builds up to 220 3-itemsets, whereas the proposed algorithm builds 90. Thus, the proposed algorithm reduced the number of generated itemsets, implying a reduction in the number of product association rules and, therefore, reducing calculation time.

B. Constraints

The parameters α and $MinSupp_1$ must be selected in a way to have acceptable and non-disjoint intervals to maintain the association between the items of the different intervals.

- Disjoint intervals: for example, α = 0.6, $MaxSupp_1$= 0.7, and $MinSupp_1$=0.6 gives the first interval [0.7, 0.6]. $MaxSupp_2 = \alpha \times MaxSupp_1$ = 0.6 × 0.7 = 0.42 and $MinSupp_2 = \alpha \times MinSupp_1$ = 0.6 × 0.6 =0.36 gives the second interval [0.42, 0.36]. Therefore, the items that support the interval]0.6, 0.42[are ignored and the association between the items of the two intervals [0.7, 0.6] and [0.42, 0.36] is lost.
- Acceptable intervals: avoid intervals that are sufficiently close to each other to ignore the same items (repetitive intervals). For $MaxSupp_1$=0.7, $MinSupp_1$=0.6, and α=0.98, the next interval is [0.686, 0.588]. Therefore, the same items of the interval [0.7, 0.6] can be studied.

The selection of α and $MinSupp_1$ must be studied.

C. The Proposed Algorithm

The principles of the proposed algorithm are:

1. The following parameters are set in advance: $MaxSupp_1$, $MinSupp_1$, *MinSupp* ($MinSupp_1$>*MinSupp*), $\alpha \in]0, 1[$.
2. The Apriori algorithm is applied to the items with support in [$MaxSupp_1$, $MinSupp_1$] to find the generated *k*-itemsets.
3. Step 2 is repeated for the intervals [$MaxSupp_i$, $MinSupp_i$] until *MinSupp* > $MinSupp_i$, using (2) and (3).

The final global set of *k*-itemsets obtained is the union of the sets of *k*-itemsets obtained in each interval. The pseudocode representation for this algorithm is:

```
Algorithm frequent_itemsets
Input: T, α, MinSupp1, MaxSupp1, MinSupp
//Knowing that MinSupp<MinSupp1<MaxSupp1, 0<α<1)
Output: List of Frequent Itemsets (LFI)
1. Repeat
2. L_1i = list of items ∈ [MaxSupp_i, MinSupp_i]
3. Apply the Apriori algorithm on L_1i to find the
LFI_i list that represents the set of frequent k-
itemsets in the interval [MaxSupp_i, MinSupp_i]
4. i++
5. MaxSupp_i= α×MaxSupp_i-1
6. MinSupp_i= α×MinSupp_i-1
7. Until (MinSupp>MinSupp_i)
8. LFI= U_i LFI_i
```

IV. EXPERIMENTAL RESULTS AND EVALUATION

A. Dataset Description

To measure the performance of the proposed against the Apriori algorithm, their computational complexity was compared on two real benchmark databases of different sizes (i.e. different numbers of items and transactions) [27]. Table IV shows the characteristics of these databases.

TABLE IV. CHARACTERISTICS OF EXPERIMENTAL DATA

Dataset name	Number of items	Number of transactions
Chess	75	3196
Mushroom	119	8124

B. Experiments and Validation

Different values of α, $MinSupp_1$, and *MinSupp* were examined for each database to find the lower computational complexity. Table V presents the obtained experimental results.

TABLE V. DATABASE TEST RESULTS

Dataset	Minimum support (*MinSupp*)	Apriori algorithm	Proposed algorithm		
		Number of frequent items	Settings α, $MinSupp_1$	Intervals	Number of items in the interval $[MaxSupp_i, MinSupp_i]$
Chess	0.6020	34	α = 0.9 $MinSupp_1$ =0.85	[0.9997, 0.85] [0.8998, 0.765] [0.8098, 0.6885] [0.7288, 0.6020]	16 9 8 10
Mushroom	0.0666	66	α = 0.55 $MinSupp_1$= 0.4	[0.9654, 0.4] [0.5310, 0.22] [0.2921, 0.121] [0.1607, 0.0666]	21 28 25 20

The computational complexity according to the k items appearing in the itemsets for the two algorithms was studied. Table VI shows the results of applying the two algorithms on the two datasets, pointing out the performance of the proposed algorithm in the reduction of the number of generated itemsets, indicating time-saving. For example, the number of 3-itemsets generated from the Mushroom database was 45760 itemsets for Apriori and 8046 itemsets for the proposed algorithm. Further experiments were carried out to compare execution time and memory space required by the proposed algorithm, Eclat, and LCMFreq. The total runtime was determined in seconds (s) and the memory space in megabytes (MB) at the end of each algorithm for the two databases. Tables VII and VIII show the obtained results.

TABLE VI. COMPUTATIONAL COMPLEXITY COMPARISON FOR k=2 AND k=3

Dataset	Computational complexity (C_n^k)			
	k=2		k=3	
	Apriori	Proposed	Apriori	Proposed
Chess	561	229	5984	820
Mushroom	2145	1078	45760	8046

TABLE VII. TOTAL RUNTIME COMPARISON

Dataset	Proposed	Eclat	LCMFreq
Chess	15.054	17.565	52.245
Mushroom	26.192	40.419	81.292

TABLE VIII. USED MEMORY SPACE COMPARISON

Dataset	Proposed	Eclat	LCMFreq
Chess	312.33	743.16	631.90
Mushroom	531.14	814.19	749.27

Figures 1 and 2 show the execution time and the memory space used by the three algorithms applied to the two transaction databases. Figure 1 shows that the time obtained by the proposed algorithm is less than the other two algorithms with times of ~15s and ~26s for the Chess and Mushroom databases, respectively. Figure 2 shows that the proposed algorithm is more efficient in terms of memory space used during execution, as it consumed only ~312MB and ~531MB for the Chess and Mushroom databases, respectively.

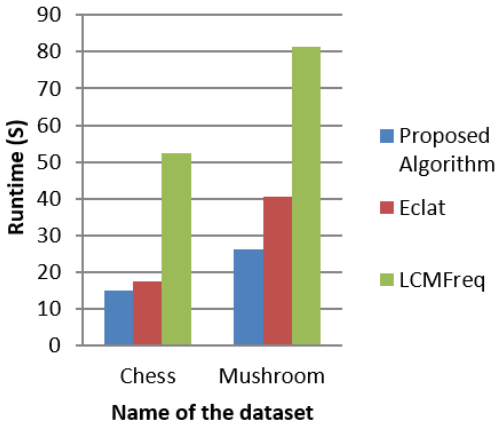

Fig. 1. Total runtime of each algorithm.

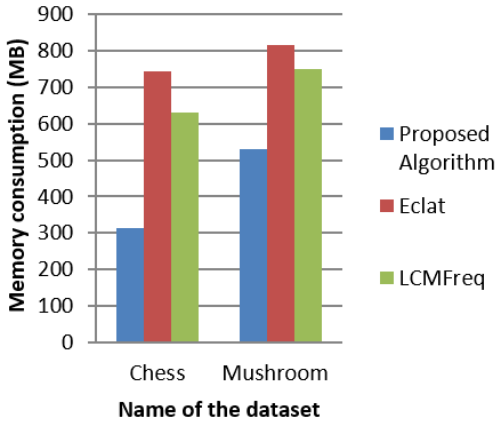

Fig. 2. Memory consumption of each algorithm.

V. CONCLUSION

This paper presented a new, based on Apriori, algorithm that allows the optimization of the extraction of association rules from databases. This optimization was based on the classification of items by categories according to their support. Experiments were carried out using the two transactional databases Chess and Mushroom [27] to compare the performance of the proposed algorithm with the Eclat and LCMFreq algorithms, showing that the proposed algorithm was faster in both cases and consumed less memory space. For example, the proposed algorithm can save more than half of the memory on the Chess database compared to the Eclat and LCMFreq algorithms. In future extensions of this study on improving the process of extracting frequent itemsets, the best value of the parameter α and the optimal number of intervals for reducing computational complexity in terms of execution time and memory consumption should be determined automatically. Furthermore, data structures, such as trees, could be investigated for the data representation of each interval of the transaction database instead of generating itemsets.

REFERENCES

[1] A. Alqahtani, H. Alhakami, T. Alsubait, and A. Baz, "A Survey of Text Matching Techniques," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6656–6661, Feb. 2021, https://doi.org/10.48084/etasr.3968.

[2] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Mar. 1993, pp. 207–216, https://doi.org/10.1145/170035.170072.

[3] S. Chakraborty, S. H. Islam, and D. Samanta, "Introduction to Data Mining and Knowledge Discovery," in *Data Classification and Incremental Clustering in Data Mining and Machine Learning*, S. Chakraborty, S. H. Islam, and D. Samanta, Eds. Cham, Switzerland: Springer International Publishing, 2022, pp. 1–22.

[4] H. Alizadeh and B. M. Bidgoli, "Introducing A Hybrid Data Mining Model to Evaluate Customer Loyalty," *Engineering, Technology & Applied Science Research*, vol. 6, no. 6, pp. 1235–1240, Dec. 2016, https://doi.org/10.48084/etasr.741.

[5] C. Kenneth and O. Chinecherem, "Knowledge Discovery in Databases (KDD): An Overview," *International Journal of Computer Science and Information Security*, vol. 15, no. 12, pp. 13–16, Dec. 2017.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Knowledge discovery and data mining: towards a unifying framework," in *Proceedings of the*

Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, May 1996, pp. 82–88.

[7] B. Bouaita, A. Moussaoui, and N. E. I. Bachari, "Rainfall estimation from MSG images using fuzzy association rules," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 1, pp. 1357–1369, Jan. 2019, https://doi.org/10.3233/JIFS-182786.

[8] N. Benmoussa, M. F. Amr, S. Ahriz, K. Mansouri, and E. Illoussamen, "Outlining a Model of an Intelligent Decision Support System Based on Multi Agents," *Engineering, Technology & Applied Science Research*, vol. 8, no. 3, pp. 2937–2942, Jun. 2018, https://doi.org/10.48084/etasr.1936.

[9] H. Li and P. C.-Y. Sheu, "A scalable association rule learning heuristic for large datasets," *Journal of Big Data*, vol. 8, no. 1, Jun. 2021, Art. No. 86, https://doi.org/10.1186/s40537-021-00473-3.

[10] K. Fujioka and K. Shirahama, "Generic Itemset Mining Based on Reinforcement Learning," *IEEE Access*, vol. 10, pp. 5824–5841, 2022, https://doi.org/10.1109/ACCESS.2022.3141806.

[11] R. Agrawal, R. Srikant, H. Road, and S. Jose, "Fast Algorithms for Mining Association Rules," in *Proceedings of the 20th International Conference on Very Large Data Bases*, 487-499, 1994.

[12] A. Ceglar and J. F. Roddick, "Association mining," *ACM Computing Surveys*, vol. 38, no. 2, Apr. 2006, https://doi.org/10.1145/1132956.1132958.

[13] J. S. Park, M. S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 175–186, Feb. 1995, https://doi.org/10.1145/568271.223813.

[14] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, New York, NY, USA, Mar. 1997, pp. 255–264, https://doi.org/10.1145/253260.253325.

[15] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372–390, Feb. 2000, https://doi.org/10.1109/69.846291.

[16] G. Gardarin, P. Pucheral, and F. Wu, "Bitmap based algorithms for mining association rules," presented at the 14ème Journées Bases de Données Avancées, Hammamet, Tunis, 1998.

[17] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1–12, Feb. 2000, https://doi.org/10.1145/335191.335372.

[18] J. Pei, J. Han, H. Lu†, S. Nishio, S. Tang, and D. Yang, "H-Mine: Fast and space-preserving frequent pattern mining in large databases," *IIE Transactions*, vol. 39, no. 6, pp. 593–605, Mar. 2007, https://doi.org/10.1080/07408170600897460.

[19] G. Liu, H. Lu, W. Lou, Y. Xu, and J. X. Yu, "Efficient Mining of Frequent Patterns Using Ascending Frequency Ordered Prefix-Tree," *Data Mining and Knowledge Discovery*, vol. 9, no. 2, pp. 249–274, Nov. 2004, https://doi.org/10.1023/B:DAMI.0000041128.59011.53.

[20] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 10, pp. 1347–1362, Jul. 2005, https://doi.org/10.1109/TKDE.2005.166.

[21] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets," presented at the The Fourth IEEE International Conference on Data Mining (ICDM '04), Brighton, UK, Nov. 2004.

[22] Z. Deng, Z. Wang, and J. Jiang, "A new algorithm for fast mining frequent itemsets using N-lists," *Science China Information Sciences*, vol. 55, no. 9, pp. 2008–2030, Sep. 2012, https://doi.org/10.1007/s11432-012-4638-z.

[23] Z. H. Deng and S. L. Lv, "Fast mining frequent itemsets using Nodesets," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4505–4512, Aug. 2014, https://doi.org/10.1016/j.eswa.2014.01.025.

[24] Z. H. Deng and S. L. Lv, "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children–Parent Equivalence pruning," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5424–5432, Aug. 2015, https://doi.org/10.1016/j.eswa.2015.03.004.

[25] Z.-H. Deng, "DiffNodesets: An efficient structure for fast mining frequent itemsets," *Applied Soft Computing*, vol. 41, pp. 214–223, Apr. 2016, https://doi.org/10.1016/j.asoc.2016.01.010.

[26] N. Aryabarzan, B. Minaei-Bidgoli, and M. Teshnehlab, "negFIN: An efficient algorithm for fast mining frequent itemsets". In Expert Systems with Applications, vol. 105, pp. 129-143, Sep. 2018. https://doi.org/10.1016/j.eswa.2018.03.041.

[27] "Chess and Mushroom datasets," *Frequent Itemset Mining Dataset Repository*. http://fimi.uantwerpen.be/data/.