







Computation of frequent itemset using matrix based apriori algorithm

Samin Jayaram Vivekanandan^{1,2*} and Gurusamy Gunasekaran³



¹Faculty of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamilnadu 600119, India; ²Department of Computer Science and Engineering, Dhanalakshmi College of Engineering, Chennai, Tamilnadu 601301, India; ³Department of Computer Science and Engineering, Dr. M. G. R. Educational and Research Institute, Chennai, Tamilnadu 600095, India

E-mail/Orcid Id:

SJV,  saivivekphd@gmail.com,  <https://orcid.org/0000-0001-7581-4728>;
GG,  gunasekaran.cse@drmgrdu.ac.in,  <https://orcid.org/0000-0003-2331-8014>

Article History:

Received: 27th Feb., 2023

Accepted: 12th Apr., 2023

Published: 30th Apr., 2023

Keywords:

Frequent Itemsets,
Matrix Based Apriori
(MB Apriori),
Transaction Matrix

Abstract: The Apriori Algorithm is a traditional method for determining the frequent itemsets from a lot of data. Association rules can be generated based on frequently occurring itemsets. The Apriori algorithm has two bottlenecks: it generates a large number of candidate sets and repeatedly examines the database. It takes a long time to execute and takes up a lot of space. We provide a novel strategy called Matrix-Based Apriori Algorithm to overcome these problems. It is easy to implement but effective in handling the issues of Apriori. We don't need to constantly scan the database because all operations are first applied to the matrix, after which the database is converted back into its original form. In addition, we have reduced the potential itemsets by using several pruning techniques. The Matrix Based Apriori algorithm outperforms the standard Apriori algorithm in terms of time, with an average time reduction rate of 71.5% with the first experiment and 86% with the second. In a similar vein, we contrasted our Matrix Based Apriori with an effective alternative known as improved apriori. We discovered that our method outperforms the upgraded apriori by 20%.

Introduction

From the huge amount of data, it is necessary to convert it into useful information, Data mining is used for this purpose. Data mining (Gupta, 2019) is used to find hidden information, unknown patterns or interesting rules from a large amount of data. Different data mining techniques exist (Ming-Syan et al., 1996) like classification, clustering, temporal data, and so on. Initially, data mining faced different requirements and challenges (Ming-Syan et al., 1996) to satisfy its need and goals. Data mining plays a key role in the KDD process. Knowledge Discovery from Data (KDD) process is the backbone of data mining; it follows the sequence of steps to extract knowledge from the data (Gupta, 2019). Data mining has various applications such as fraud detection, financial analysis, medical field, CRM, scientific applications, and other applications. The

Apriori algorithm is one of the prime algorithms for determining the frequent itemsets. But, it has many problems, i.e., repeatedly examining the database, not handling the redundant transaction, and generating a large number of candidate sets. A novel strategy called Matrix Based Apriori Algorithm has been proposed to overcome these issues.

Association Rule Mining

“Association Rules Mining is one of the most important research areas among the researchers. Finding strong association rules or the connections between itemsets from a large amount of data is the goal of association rules mining. $A \rightarrow X$, where A is the antecedent and X is the consequent, is the symbol for an association rule. It implies that X could happen as well if



A happens. It is mainly used in shopping analysis to identify customer purchase habits.

Basic Terminologies used in Association Rule Mining

Let $I = \{I1, I2, I3, \dots, In\}$ be a collection of items and $Tran_DB = \{T1, T2, T3, \dots, Tn\}$ a collection of transactions where every transaction is also a collection of items. Association rules can be dominated by support and confidence. Support is the number of times the items occur in the database. $Support(i1) = \text{ratio of the number of times } i1 \text{ appears in the database to the total number of transactions}$. $Support(i1i2) = \text{ratio of the number of times } i1i2 \text{ appears together in the database to the total number of transactions}$. Confidence is the probability of A occurring when B also occurs, where A and B are two different itemsets. $Confidence(i1 \rightarrow i2) = \text{ratio of Support}(i1i2) \text{ to the Support}(i1)$. Association mining is a process of finding strong association rules (Agrawal and Srikant, 1994; Agrawal et al., 1993; Vivekanandan and Gunasekaran, 2019). It can be completed in two steps.

Frequent Itemset Generation, in this step, it computes all itemsets that are not less than the support value.

Association Rule Generation, based on step 1, it has pulled out all the rules that were not less than the confidence value. Those rules are called strong association rules.

Different Algorithms used in Association Rule Mining

The second step of an association rule, i.e., generating an association rule is a very direct method. So, more research work is only in the first step of the association rule, i.e., finding frequent itemsets. In general, Association rule mining is categorized into pairs of methods.

Candidate Generation Method, this method generates a candidate itemset at every level. Example: “Apriori Algorithm (Agrawal and Srikant, 1994; Gupta, 2019).

No Candidate Generation Method, in this method, it will not generate any candidate itemset. Example: “Frequent Pattern algorithm” (Han et al., 2000).

Apriori Algorithm

Apriori algorithm is a famous algorithm for finding association rules. There are two important properties used in apriori algorithm. First, if an item is frequent, all its subset combinations are also frequent. Second, if an item is a rare, all its superset combinations also rare (Agrawal et al., 1993; Vivekanandan and Gunasekaran, 2019). There are two major steps in apriori algorithm. The join step is generated by connecting by itself, and the Prune

step is the process of removing infrequent itemset. First, it scans the database and finds the support of each item and it’s taken as C1. Then it is compared with the min_sup value, and all the items which are lesser than min_sup will be deleted. The remaining items will be taken as L1. To generate C2, L1 is self-connected with another L1 and it creates C2. Again min_sup is compared with C2 and deletes the itemsets which are lesser than min_sup. Repeat this process, until there are no more itemsets.

Table 1. Sample Database

Transaction ID	Itemsets
T1	I1, I2, I3, I5
T2	I2, I4
T3	I2, I3
T4	I1, I4, I6
T5	I1, I3
T6	I3, I6
T7	I1, I3
T8	I1, I2, I3, I5
T9	I1, I2, I3
T10	I4, I5

In Table 1, it contains ten different transactions with six different items. Apriori algorithm applies to this transaction and we have different itemsets at each level. The overall result is described in Table 2.

Table 2. Result and Explanation of Apriori Algorithm

C1				L1	
Itemsets	Support	Itemsets	Support	Itemsets	Support
I1	6	I1	6	I1	6
I2	5	I2	5	I2	5
I3	7	I3	7	I3	7
I4	3	I4	3	I4	3
I5	3	I5	3	I5	3
I6	2	I6	2	I6	2
C2				L2	
Itemsets	Support	Itemsets	Support	Itemsets	Support
I1, I2	3	I3, I6	1	I1, I2	3
I1, I3	5	I4, I5	1	I1, I3	5
I1, I4	1	I4, i6	1	I1, I5	2
I1, I5	2	I5, I6	0	I2, I3	4
I1, I6	1			I2, I5	2
I2, I3	4			I3, I5	2
I2, I4	1				
I2, I5	2				
I2, I6	0				
I3, I4	0				
I3, I5	2				
C3				L3	
Itemsets	Support	Itemsets	Support	Itemsets	Support
I1, I2, I3	3	I1, I2, I3	3	I1, I2, I3	3
I1, I2, I5	2	I1, I2, I5	2	I1, I2, I5	2
I1, I3, I5	2	I1, I3, I5	2	I1, I3, I5	2
I2, I3, I5	2	I2, I3, I5	2	I2, I3, I5	2

C4		L4	
Itemsets	Support	Itemsets	Support
I1, I2, I3, I5	2	I1, I2, I3, I5	2

Demerits of Apriori Algorithm

First, it consumes most of the time in the scanning databases to estimate the support value of the things or itemsets (Agrawal and Srikant, 1994). Second, it generates a vast number of candidate itemsets. If a transactional database has 'n' different items, then it will generate $2^n - 1$ candidate itemsets (Al-Maolegi and Arkok, 2014; Vivekanandan and Gunasekaran, 2019). For each itemset, it scans the database to find the support count of the itemset. Third, it is not handling the repeated transactions.

Related Works

A mathematical model and template algorithm were proposed to address the complicated mining association rules. This mathematical model was the base for ARM. With the help of this template algorithm, the Apriori algorithm was proposed. Each iteration constructed a set of frequent itemsets, counted the itemset appearances, and discovered frequent itemsets based on min_sup. "Apriori-TID algorithm was proposed to compute the frequent occurrence itemsets. $\langle Tid, \{Xk\} \rangle$ was used to compute the frequent occurrence itemsets. Tid represented the transaction identifier and Xk represented the large itemsets in the transaction" (Agrawal and Srikant, 1994). It was more efficient than the original algorithm only when the database was small. "DHP algorithm proposed to enrich the original apriori algorithm. It has utilized a hash method for candidate item generation during the initial iterations and then gradually, the database size has reduced by using pruning techniques" (Park et al., 1995). This algorithm also worked well only if the database size was small; if the database was large, it took a large amount of memory for the hash table.

"Dynamic Itemset Counting (DIC) (Gupta, 2019) is an approach to enhance the traditional apriori. It separates the database into different partitions. It scans the first partition for 1-frequent itemset and combines with the next partition until it completes entire partitions". It works well when the database is in homogeneous. If not, it will not work well. "An enhanced version of apriori algorithm for association rules (Al-Maolegi and Arkok, 2014)" was used to improve the performance of traditional apriori. It reduces the number of times the transactional database has been scanned. In the original

apriori algorithm, they have made a slight variation in the logic that gives more efficiency for this improved algorithm. Association rule mining applied in direct marketing applications (Carter et al., 1997; Wang et al., 2005) to yield profits in their business depends on buyer history. The association rule is used to form a model to predict the group of customers. HDO algorithm (Ji et al., 2006) is used to find association rules in the high-dimensional data. This procedure embraces another strategy to remove candidate itemsets with rare itemsets instead of frequent itemsets, which can be removed validly with the rare itemsets with lower dimension (Ji et al., 2006). A novel improved algorithm (Wu et al., 2010) was used to improve the mining capabilities of apriori algorithm. In this algorithm, they introduced some concepts called interest items and frequency threshold, which decrease the database search time. Also, dynamic mining is used to increase the performance of the algorithm. "Another improved algorithm was used to find association rules based on utility weighted score which are extracted from weightage constraint and utility gain (Sandhu et al., 2010). In this algorithm, association rules are generated based on frequency as well as significant of the itemsets. A matrix based apriori algorithm (Wang and Li, 2008; Yang et al., 2018) was used to improve the capability of the algorithm. In the matrix-based method, transactional Boolean matrix was used to find the different candidate itemset generation. It is highly efficient than the original approach. An enhanced algorithm based on time series (Wang and Zheng, 2020) was used to enhance the algorithm's performance. This enhanced algorithm uses a Boolean matrix and different new concepts like sequence association rule and frequent item sequence generation. A new apriori algorithm was proposed in (Wang and Li, 2008; Singh and Dhir, 2013; Sun, 2020; Yang et al., 2018) which is based on a support weight matrix. It uses 0-1 transaction matrix and gives association rules and significance to the user. A new matrix-based apriori algorithm proposed by (Shuwen and Jiyi, 2020) is based on horizontal index and vertical index value.

A new matrix-based apriori algorithm proposed by (Singh and Dhir, 2013) is based on tag values with all transactions. It mainly reduces the 2-candidate itemsets. Improved apriori algorithm used by (Vivekanandan and Gunasekaran, 2020) reduced the candidate itemsets and repeated transactions efficiently. Categories of itemsets were discussed efficiently using frequent and infrequent itemsets (Vivekanandan and Gunasekaran, 2022). Utility Mining is an enhanced version of Apriori algorithms discussed deeply (Vivekanandan and Gunasekaran, 2019;

Vivekanandan et al., 2021). “The Apriori algorithm is used in rural tourism applications and is more effective than state-of-art algorithms” (Xiao, 2022). In this study, various applications of the apriori method that are used in China are successfully explored (Xie, 2021). An effective incremental frequent itemset mining with closed itemsets has been proposed. In this approach, new transactions were added dynamically (Magdy et al., 2022). “By using

Pruning Strategies used in Matrix Based Apriori Approach

To reduce the candidate itemsets, we have proposed five different pruning strategies.

Strategy I: If an itemset is not frequent, all its supersets are also not frequent.

Strategy II: If an itemset is frequent, then all its non-empty subsets are also frequent.

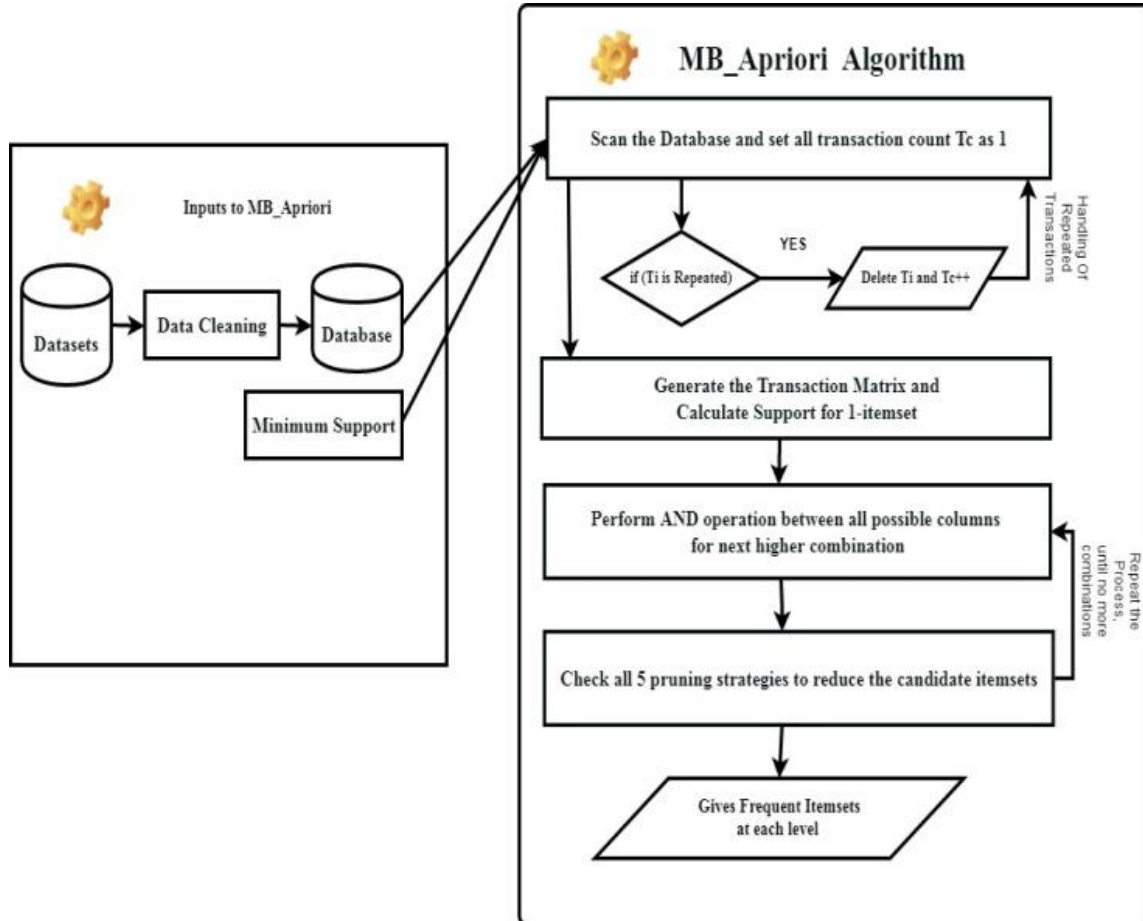


Figure 1. Architecture Diagram of Matrix Based Apriori Algorithm

Apriori algorithm, acupoints were identified effectively for the treatment of COVID-19” (Zheng, 2022). A frequent itemset based method called “SHFIM (spark-based hybrid frequent itemset mining)” (Al-bana and Farhan, 2022) were proposed to handle the big data.

Proposed Methodology

In this section, we have discussed our proposed matrix based apriori algorithm with example. Let ‘n’ be the set of transactions with ‘m’ items. In the matrix, each row indicates the transaction and each column indicates the item. If an item is available in a transaction, then it denotes with 1 otherwise 0 i.e., $M_{nm} = 1$ if available otherwise 0. Transaction count is a mechanism to handle the repeated transaction. Initially, we take the value 1 for all transactions, whenever the repeated transaction occurs, it removes that transaction and increments the value of the transaction count.

Strategy III: In a matrix, whenever a column entries count is less than the minimum support, we can delete that column from the matrix.

Strategy IV: In a matrix, whenever we get some row entries are 0, then we can delete that row from the matrix.

Strategy V: If a transaction contains only ‘K’ items, then it cannot compute K+1 itemsets. Then the transaction can be eliminated (Ye, 2020).

Architecture Diagram of Matrix Based Apriori Approach

Datasets have lots of irrelevant data, so it is necessary to do preprocessing to remove irrelevant and unwanted data. Minimum support and database are the inputs given to the MB_Apriori approach. In this approach, we have used a concept called Transaction Count (Tc) to remove repeated transactions. Once all the repeated transactions are removed, we create a transaction matrix and calculated each item’s support.

Higher itemsets can be generated by performing AND operation between all possible columns. At each level, we have to check all pruning strategies to eliminate unwanted candidate itemsets. Finally, it gives a frequent itemset which is greater than or equal to minimum support.

Matrix Based Apriori Algorithm

Input: Transaction Database (TB), Minimum Support (Min_Support)

Output: Frequent Itemsets

Scan the TB and Set all transaction count is set to 1 i.e. $T_c = \{1, 1, \dots, 1\}$. //Repeated Transaction

Generate the transaction matrix

If (Ti is repeated) then delete the transaction in the matrix and increment its transaction count Tc by 1. //Repeated Transaction

Support = Summation of (Tc*I) i.e., support can be calculated by transaction count * entries (0 or 1) in the column and adding all the values.

Combination can be generated by performing AND operation between the columns

Apply all pruning strategies to reduce the candidate itemsets.

Repeat the steps 2 and 3, until there is no more combination of itemsets.

Example of Matrix Based Apriori Algorithm

Let us take the Table 1 for our example and we assume minimum support = 2, In step 1, $T_c = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$. In step 2, it generates the transaction matrix. It is depicted in Figure 2. In Table 2, T1 and T8 has same items i.e., T8 is a repeated transaction, so T8 can be deleted from the matrix and increment the value of Tc for T1 by 1. Therefore, $T_c = \{2, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$. Similarly, T5 and T7 have the same items i.e., T7 is a repeated transaction. So, T7 can be deleted from the matrix and increment the value of Tc for T5 by 1. Therefore, $T_c = \{2, 1, 1, 1, 2, 1, 1, 1, 1, 1\}$. In Figure 3, T8 and T7 are deleted from the table.

I ₁	I ₂	I ₃	I ₄	I ₅	I ₆
1	1	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	0	1

1	0	1	0	0	0
1	1	1	0	1	0
1	1	1	0	0	0
0	0	0	1	1	0

Figure 2. Transaction Matrix I

I ₁	I ₂	I ₃	I ₄	I ₅	I ₆
1	1	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
0	0	1	0	0	1
1	1	1	0	0	0
0	0	0	1	1	0

Figure 3. Transaction Matrix II

Support(I1) = 6, Support(I2) = 5, Support(I3) = 7, Support(I4) = 3, Support(I5) = 3, Support(I6) = 2. In step 3, AND operation between all the possible columns. So, 2-Itemsets have been generated in Figure 4.

In step 3a, we checked all six pruning strategies and eliminated the candidate itemsets. By using Strategy III, we eliminated the following itemsets columns i.e. I1I4, I1I6, I2I4, I2I6, I3I4, I3I6, I4I5, I4I6 and I5I6. Now we have only 6 columns. By using Strategy IV, we eliminated the following transactions, i.e., T2, T4, T6 and T10. Now, we have got only 4 transactions in the matrix. By using Strategy V, we eliminated T3 and T5. So, after applying all the pruning strategies, we have got the matrix like Figure 5. It contains only two transactions, i.e. T1 and T9. So, through our pruning strategies, we eliminated 9 columns and 6 rows.

I ₁ I ₂	I ₁ I ₃	I ₁ I ₄	I ₁ I ₅	I ₁ I ₆	I ₂ I ₃	I ₂ I ₄	I ₂ I ₅	I ₂ I ₆
1	1	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0

I₃I₄	I₃I₅	I₃I₆	I₄I₅	I₄I₆	I₅I₆
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0

Figure 4. Transaction Matrix III

I₁I₂	I₁I₃	I₁I₅	I₂I₃	I₂I₅	I₃I₅
1	1	1	1	1	1
1	1	0	1	0	0

Figure 5. Transaction Matrix IV

Support(I1I2) = 3, Support(I1I3) = 3, Support(I1I5) = 2, Support(I2I3) = 3, Support(I2I5) = 2, Support(I3I5) = 2. Now, again performs AND operation between all the possible columns. Its result shows in Figure 6. By using Strategy V, T9 is deleted, so the updated matrix would have only 1 transaction i.e., T1. The updated matrix shows in Figure 7. Again, perform AND operation between all the possible column, then finally we have got the 4-Itemsets which is depicted in Figure 8. Thus, this algorithm ends because there are no more itemsets.

I₁I₂I₃	I₁I₂I₅	I₁I₃I₅	I₂I₃I₅
1	1	1	1
1	0	0	0

Figure 6. Transaction Matrix V

I₁I₂I₃	I₁I₂I₅	I₁I₃I₅	I₂I₃I₅
1	1	1	1

Figure 7. Transaction Matrix VI

I₁I₂I₃I₅
1

Figure 8. Transaction Matrix VII

Results and Discussion

We have implemented our algorithm using Python with a different set of transactions. TB1 contains 570 transactions, TB2 contains 930 transactions, TB3 contains 1260 transactions, TB4 contains 2390 transactions and TB5 contains 3100 transactions. Our MB_Apriori performs well than the traditional Apriori algorithm. In our proposed algorithm, we handled the repeated transactions efficiently and also by using different pruning strategies, we have reduced the huge number of candidate itemsets efficiently. In our first experiment, we compare the time consumption of traditional apriori and MB_Apriori by applying different transactional databases. The experimental results are depicted in Figure 9. It clearly indicates that our proposed

MB_Apriori outperforms the traditional Apriori algorithm.

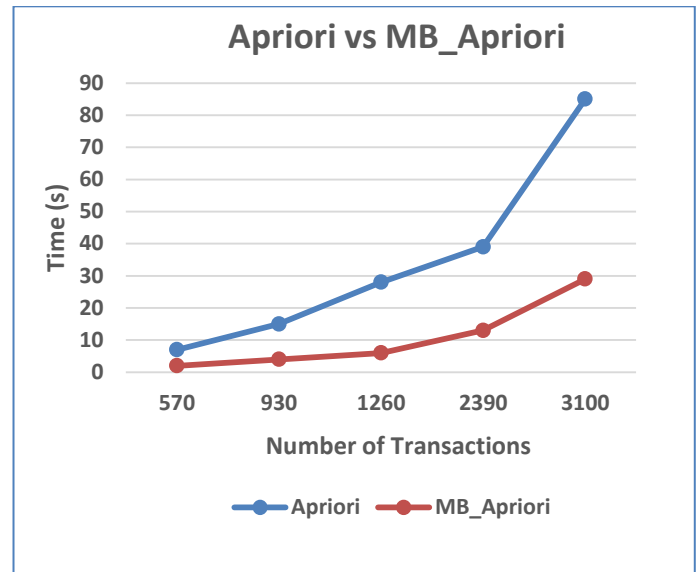


Figure 9. Time Comparison with different Transaction Database

Table 3. Rate of Time Reduction

Transaction Database	Traditional Apriori Algorithm	MB_Apriori Algorithm	Rate of time reduction (%)
TB1	7	2	71.4%
TB2	15	4	73.33%
TB3	28	6	78.57%
TB4	39	13	68.42%
TB5	85	29	65.88%

Table 3 showed the time consumption of MB_Apriori is less than the traditional apriori algorithm. Our experiment found that the average reducing time rate is 71.52%. In our next experiment, we compare the time with one group of transactions with different support values. The results show that our proposed MB_Apriori performs well than the traditional apriori algorithm. It's clearly depicted in Figure 10. The different support values are given on the horizontal axis and the time is given on the vertical axis. The graph clearly showed that MB_Apriori has taken not more time than the original apriori algorithm.

Table 4 showed the time consumption of MB_Apriori is less than the traditional apriori algorithm. With our experiment, we found that the average rate of time reduction is 86%. Our experiment took the Groceries dataset (*Groceries Dataset*, n.d.), which contains 38765 transactions with 150 different itemsets.

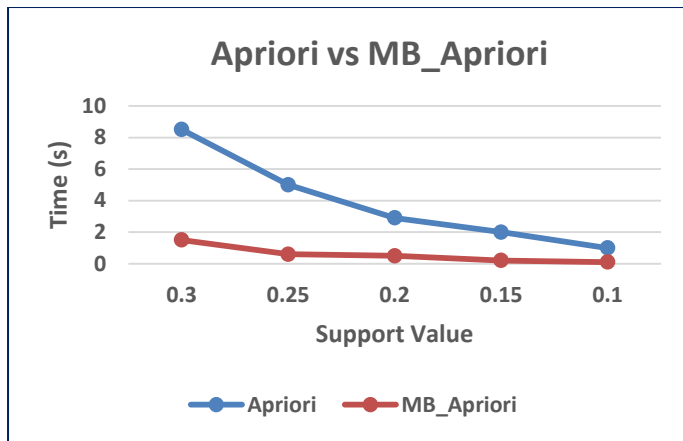


Figure 10. Time Comparison with different Support value.

Table 4. Time Reducing Rate of UF-Apriori Algorithm with various support values

Minimum Support (min_sup)	Traditional Apriori Algorithm	MB_Apriori Algorithm	Rate of time reduction (%)
0.3	8.5	1.5	82.35%
0.25	5	0.6	88.0%
0.2	2.9	0.5	82.75%
0.15	2	0.2	90%
0.1	1	0.1	90%

Our implementation differentiates the time consumed of traditional apriori and MB_Apriori by using Groceries dataset (*Groceries Dataset*, n.d.). The results show that our MB_Apriori outperforms the traditional apriori approach in terms of time and efficiency with different support values. The different support values are given on the horizontal axis and the time is given on the vertical axis. The graph clearly showed that MB_Apriori has taken not more time than the original apriori algorithm. It's clearly depicted in Figure 11.

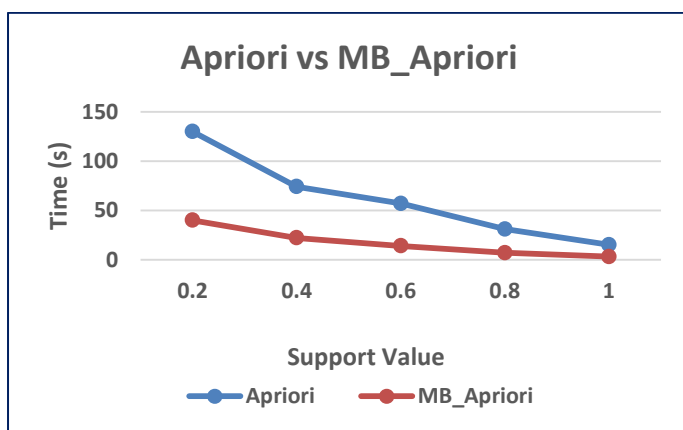


Figure 11. Groceries Dataset with different Support value

Table 5. Time Reducing Rate of UF-Apriori Algorithm with various support values

Minimum Support (min_sup)	Traditional Apriori Algorithm	MB_Apriori Algorithm	Rate of time reduction (%)
0.2	130	40	69.23%
0.4	74	22	70.27%
0.6	57	14	75.43%
0.8	31	7	77.41%
1	15	3	80%

Table 5 showed the time consumption of MB_Apriori is less than the traditional apriori algorithm for the groceries dataset. In this experiment, we found that the average time reduction rate is 74.46%. We can compare our MB_Apriori with the Traditional Apriori algorithm with our solved example. Traditional Apriori algorithm scans the database 26 times, but our proposed MB_Apriori algorithm scans only one time. It is clearly depicted in Figure 12. So, our proposed MB_Apriori is 96% efficient than the traditional apriori algorithm regarding the number of scans.

In our next experiment, we compared our MB_Apriori with another proposed matrix based algorithm (Yang et al., 2018). Both algorithms may look similar, but in our approach, we included 5 different pruning strategies, so it reduces the candidate itemsets more efficiently than the improved apriori (Yang et al., 2018). We have implemented our algorithm using Python with a different set of transactions. TB1 contains 570 transactions, TB2 contains 930 transactions, TB3 contains 1260 transactions, TB4 contains 2390 transactions and TB5 contains 3100 transactions.

Our MB_Apriori performs well than the traditional Apriori algorithm. In our proposed algorithm, we handled the repeated transactions efficiently, and also by using different pruning strategies, we have reduced the huge number of candidate itemsets efficiently. In this experiment, we compare the time consumption of improved Apriori (Yang et al., 2018) and MB_Apriori by applying the different sets of transactional databases. The experimental results are depicted in Figure 13. It clearly indicates that our proposed MB_Apriori outperforms the improved Apriori algorithm. Table 6 showed the time consumption of MB_Apriori is less than the improved apriori algorithm. Our experiment found that the average reducing time rate is 20%.

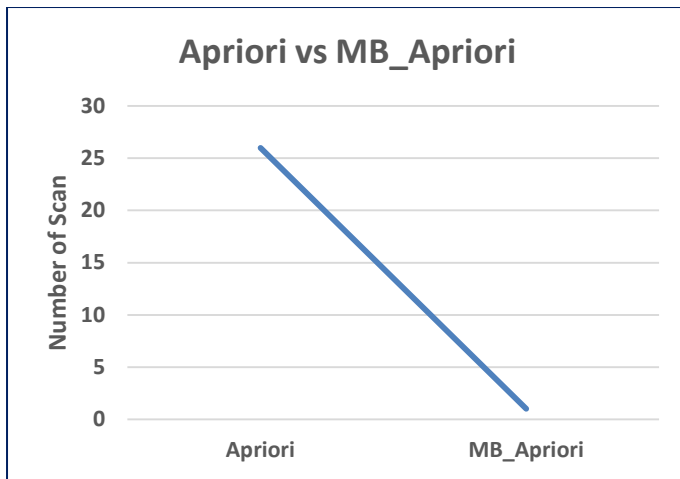


Figure 12. Apriori versus MB_Apriori

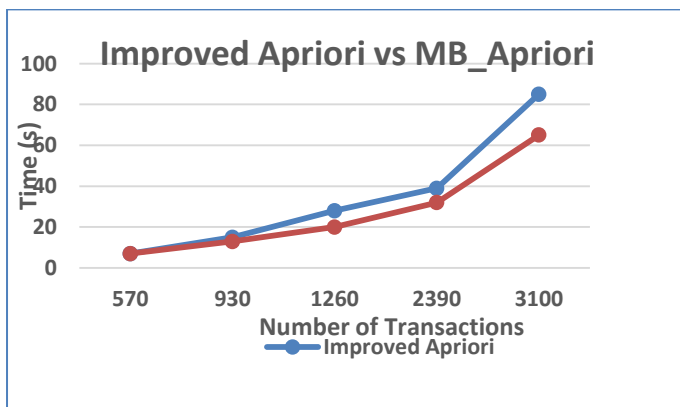


Figure 13. Improved Apriori versus MB_Apriori

Table 6. Rate of Time Reduction

Transaction Database	Improved Apriori Algorithm	MB_Apriori Algorithm	Rate of time reduction (%)
TB1	7	6	14.28%
TB2	15	13	13.33%
TB3	28	20	28.57%
TB4	39	32	17.94%
TB5	85	65	23.52%

Conclusion

In this paper, we developed a new algorithm called MB_Apriori to compute frequent itemsets with support value. Traditional Apriori algorithm has three bottle necks, huge time and memory usage due to a large number of candidate itemsets, and repeated transactions have not been handled. In our Matrix Based Apriori algorithm, transactions were converted into the matrix and all operations were performed on the matrix with different pruning strategies. Repeated transactions could be efficiently handled using transaction count. We conducted many experiments with different data sets, the

outcomes proved that MB_Apriori outperforms the traditional apriori algorithm. Also, we compared our MB_Apriori with an improved apriori algorithm, MB_Apriori is better than improved apriori. In future, MB_Apriori will be used to compute high-utility itemsets with support and utility values. Also, it will be useful to handle negative utility and average utility itemsets.

Conflict of Interest

The authors declare no conflict of interest.

References

- Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. *ACM SIGMOD Record*, 22(2), 207–216.
<https://doi.org/10.1145/170036.170072>
- Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. Proceedings of the 20th VLDB Conference Santiago, Chile, 1994, 487–499.
- Al-bana, M.R., & Farhan, M.S. (2022). An Efficient Spark-Based Hybrid Frequent Itemset Mining. *Data (MDPI)*, 7(11), 1–22.
<https://doi.org/https://doi.org/10.3390/data7010011>
- Al-Maolegi, M., & Arkok, B. (2014). An Improved Apriori Algorithm For Association Rules. *International Journal on Natural Language Computing*, 3(1), 21–29.
<https://doi.org/10.5121/ijnlc.2014.3103>
- Carter, C. L., Hamilton, H. J., & Cercone, N. (1997). Share Based Measures for Itemsets 1 Introduction. Principles of Data Mining and Knowledge Discovery, *First European Symposium, PKDD '97, Trondheim, Norway, June 24-27, 1997, Proceedings*, pp. 14–24.
- Groceries Dataset. (n.d.).
<https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset>
- Gupta, G.K. (2019). Introduction to data mining with case studies (Third Edit). PHI Learning Priivate Limited.
- Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation. *ACM SIGMOD Record*, 29(2), 1–12.
<https://doi.org/10.1145/335191.335372>
- Ji, L., Zhang, B., & Li, J. (2006). A New Improvement on Apriori Algorithm. *International*

- Conference on Computational Intelligence and Security, Guangzhou, China*, pp. 840–844.
<https://doi.org/10.1109/ICCIAS.2006.294255>
- Jiawei, H., & Micheline, K. (2006). *Data Mining: Concepts and Techniques (Second)*. Morgan Kaufmann Publishers.
- Magdy, M., Ghaleb, F.F.M., Mohamed, D.A.E.A., & Zakaria, W. (2022). CC-IFIM: an efficient approach for incremental frequent itemset mining based on closed candidates. *Journal of Supercomputing*, 79(7), 7877–7899.
<https://doi.org/10.1007/s11227-022-04976-5>
- Ming-Syan, C., Jiawei, H., & Philip, S.Y. (1996). Data Mining: An Overview from a Database Perspective. *IEEE transactions on knowledge and Data Engineering*, 8(6), 866–883.
<https://doi.org/10.1109/69.553155>
- Park, J.S., Chen, M.S., & Yu, P.S. (1995). An Effective Hash-Based Algorithm for Mining Association Rules. *ACM Sigmoid Record.*, 24(2), 175–186.
<https://doi.org/10.1145/568271.223813>
- Sandhu, P.S., Dhaliwal, D.S., Panda, S.N., & Bisht, A. (2010). An improvement in apriori algorithm using profit and quantity. *2nd International Conference on Computer and Network Technology, ICCNT 2010*, pp. 3–7.
<https://doi.org/10.1109/ICCNT.2010.46>
- Shuwen, L., & Jiyi, X. (2020). An Improved Apriori Algorithm Based on Matrix. *12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 488–491.
<https://doi.org/10.1109/ICMTMA50254.2020.00111>
- Singh, H., & Dhir, R. (2013). A New Efficient Matrix Based Frequent Itemset Mining Algorithm with Tags. *International Journal of Future Computer and Communication*, 2016, 355–358.
<https://doi.org/10.7763/ijfcc.2013.v2.184>
- Sun, L.N. (2020). An improved apriori algorithm based on support weight matrix for data mining in transaction database. *Journal of Ambient Intelligence and Humanized Computing*, 11(2), 495–501.
<https://doi.org/10.1007/s12652-019-01222-4>
- Vivekanandan, S.J., & Gunasekaran, G. A novel way to compute association rules. *Int. J. Syst. Assur. Eng. Manag.*, (2022).
<https://doi.org/10.1007/s13198-022-01676-4>
- Vivekanandan, S.J., Ammu, S.P., Sripriyadarshini, R., & Preetha, T.R. (2021). Computation Of High Utility Itemsets By Using Range Of Utility Technique. *Journal of University of Shanghai for Science and Technology*, 23(4), 94–101.
- Vivekanandan, S.J., & Gunasekaran, G. (2020). An Improvisation on Apriori Algorithm Applied in Medical Transaction. *Journal of Green Engineering (JGE)*, 10(10), 8574–8586.
- Vivekanandan, S.J., & Gunasekaran, G. (2019). A Survey on Association Rules Mining. *Asian Resonance*, 8(1), 1–4.
- Wang, F., & Li, Y.H. (2008). An Improved Apriori Algorithm Based on the Matrix. *2008 International Seminar on Future BioMedical Information Engineering*, Wuhan, China, pp. 152-155.
<https://doi.org/10.1109/FBIE.2008.80>
- Wang, C., & Zheng, X. (2020). Application of improved time series Apriori algorithm by frequent itemsets in association rule data mining based on temporal constraint. *Evolutionary Intelligence*, 13(1), 39–49.
- Wang, K., Zhou, S., Man, J., Yeung, S., Yang, Q., & Kong, H. (2005). Mining Customer Value: From Association Rules to Direct Marketing. *Data Mining and Knowledge Discovery*, 11(1), 57–79.
<http://www.kdnuggets.com/meetings/kdd98/kdd>
- Wu, L., Gong, K., Ge, H.X., & Cui, J. (2010). A Study of Improving Apriori Algorithm. *22010 2nd International Workshop on Intelligent Systems and Applications*, Wuhan, China, 2010, pp. 1-4.
<https://doi.org/10.1109/IWISA.2010.5473450>
- Xiao, H. (2022). Algorithm of Apriori-Based Rural Tourism Driving Factors and Its System Optimization. *Mobile Information Systems*, 2022, 9.
<https://doi.org/https://doi.org/10.1155/2022/3380609>
- Xie, H. (2021). Research and Case Analysis of Apriori Algorithm Based on Mining Frequent

Item-Sets. *Open Journal of Social Sciences*, 09(04), 458–468.

<https://doi.org/10.4236/jss.2021.94034>

Yang, Q., Fu, Q., Wang, C., & Yang, J. (2018). A matrix-based apriori algorithm improvement. *Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018*, pp. 824–828.

<https://doi.org/10.1109/DSC.2018.00132>

Ye, F. (2020). Research and Application of Improved APRIORI Algorithm Based on Hash Technology. *2020 Asia-Pacific Conference on*

Image Processing, Electronics and Computers (IPEC), pp. 64–67.

<https://doi.org/10.1109/IPEC49694.2020.9115141>

Zheng, Y. (2022). An Improved Apriori Association Rule for the Identification of Acupoints Combination in Treating COVID-19 Patients. *Computational Intelligence and Neuroscience*, 2022, 1-9.

<https://doi.org/10.1155/2022/3900094>

How to cite this Article:

Samin Jayaram Vivekanandan and Gurusamy Gunasekaran (2023). Computation of frequent itemset using matrix based apriori algorithm. *International Journal of Experimental Research and Review*, 30, 247-256.

DOI : <https://doi.org/10.52756/ijerr.2023.v30.022>



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.