ORIGINAL PAPER

PROPOSAL OF A DISTRIBUTED APRIORI ALGORITHM FOR HETEROGENEOUS PERFORMANCE MACHINES

FERNANDO ALMEIDA[1]

Manuscript received: 13.09.2019; Accepted paper: 20.11.2019;
Published online: 30.12.2019.

Abstract. *The Apriori algorithm is considered a classic in the association rules extraction field. This algorithm makes recursive searches in a dataset looking for frequent sets that satisfy given minimum support. Apriori has several properties to optimize its performance, such as reducing the number of generated itemsets and its parallelization by multiple processors. These features have led to the emergence of several studies that present parallel versions of Apriori. However, these proposals do not explore the heterogeneous capabilities of each machine, which causes a significant part of the algorithm's processing time to be spent on I/O processes and not exactly on the execution of the algorithm. In this sense, this study proposes a mathematical modeling of the Apriori algorithm in which heterogeneous machines are considered. The findings identified a better performance of this algorithm when compared to the original and parallel versions of Apriori, but in which all processors are considered homogeneous. The findings reveal the time reducing rate increases with the growth in the number of itemsets and the number of considered processors.*

Keywords: *decision-making, data mining, mathematical modeling, Apriori, performance.*

1. INTRODUCTION

The discovery of association rules is currently one of the most relevant tasks in data mining. This model is applied to several problems in the business area, such as online commerce, education performance analysis or insurance risk analysis [1-4].

It is pertinent to present a set of fundamental concepts that formalizes the problem of mining association rules. Let's be $T = \{t1, t2, ..., tn\}$ containing a set of n transactions, in which we have a set of m available items $I = \{i1, i2, ..., in\}$ to build each transaction $t1 \varepsilon T$, such that $ti \sqsubseteq I$. A set of items is called an *itemset*. If an itemset set has k items, it is a *k-itemset*. Considering two *itemsets* A and *B*, such that $A \sqsubseteq I$ and $B \sqsubseteq I$ and do not have items in common, that is, $A \cap B = \phi$. In this case, *A* implies *B*, where *A* is called antecedent and *B* is the consequence of the rule.

The general objective of mining is to find, from the set of *T* transactions, all the rules that associate the presence of an *itemset*, *A*, for example, with any other. However, since *I* has a total of *m* items, the search space for all rules is theoretically exponential $O(2^m)$, because all items can constitute *itemsets*.

In practice, not all *I* items are present in *T* transactions and others, occur in a very low number of transactions. According to [5], this sparseness is harnessed by mining methods, making them viable and efficient. In this sense, two measures were created for the rules,

[1] Instituto Superior Politécnico Gaya (ISPGaya) Av. dos Descobrimentos, 333. 4400-103 Santa Marinha - V.N.Gaia, Portugal. E-mail: almd@fe.up.pt; dee06003@fe.up.pt.

known as support and confidence, which are calculated from the frequency of occurrences of the *itemsets* involved in the rule.

The support *S* of an association rule, $A \rightarrow B$, is the percentage of the transaction that contains $A \ U \ B$ considering the amount of transaction *n* of *T*. The frequency of an *itemset* is denoted by *f*, which is the number of *T* transactions that contain this *itemset*. The support can be seen as the probability of occurrence of *itemset* $A \ U \ B$ in *T* and is calculated:

$$S\ (A\ \rightarrow\ B) = P\ (A \cup B) = \frac{f\ (A\ \cup B)}{n} \quad (1)$$

The support therefore indicates the relative frequency of rules and can be used to compare them, i.e., rules with high values indicate a strong presence of the itemset. On the other hand, very low support rules can only represent a random occurrence.

The confidence *C* of an association rule, $A \rightarrow B$, is the percentage of transactions that contains $A \ U \ B$ considering all the *T* transactions that contain *A*. In this case, *C* is given by the conditional probability *P(B|A)* or the probability of *B* occurrence, when *A* occurs. The calculation can be made using the formula below:

$$C\ (A\ \rightarrow\ B) = P\ (B|A) = \frac{P\ (A\ \cup B)}{P(A)} = \frac{f\ (A\ \cup B)}{f(A)} \quad (2)$$

The confidence indicates the ability to predict the rules. The rules with high confidence values stand out qualitatively from the others, by the level of certainty of occurrence of the consequent of the rule, from the cases where its antecedent occurs. On the contrary, rules with low confidence do not provide prediction safety and, therefore, are of limited use.

It is quite common in the mining algorithms of association rules the adoption of pre-established user limits for support and confidence, defined as minimum support (*sup-min*) and minimum confidence (*conf-min*), reducing the amplitude of the problem.

The Apriori algorithm is one of the best-known algorithms for the application of association rules with wide practical application in the market. The Apriori algorithm can also be easily personalized, which is a fundamental characteristic of its success in the market. In this sense, this study seeks to look at the traditional Apriori parallelization processes that consider homogeneous machines and presents a new proposal of a distributed Apriori algorithm for heterogeneous performance machines.

2. MATERIALS AND METHODS

2.1. MATERIALS

A test environment composed of several virtual machines with different CPU, processor speed and RAM was considered. The characteristics of each machine are shown in Table 1.

Table 1. Environment characteristics

ID	CPU	Processor speed	RAM
MC1	Intel Core i7-8565U	1.8 GHz	16 GB
MC2	Intel Core i5-8265U	1.6 GHz	16 GB
MC3	Intel Core i5-8250U	1.6 GHz	8 GB
MC4	Intel Core i3-8130U	2.2 GHz	4 GB

Three versions of the Apriori algorithm were adopted in the testing process: (i) the original version proposed by R. Agrawal and made available by [6]; (ii) the parallel version of Apriori proposed by [20]; and (iii) the version proposed in this study and presented in the methods section. The code was implemented in Java in the Eclipse IDE 2019-06 interface. Each test was performed 10 times and the geometric mean was calculated since it is less affected by extreme values than the arithmetic mean. The number of transactions ranged between 1000 transactions and 5000 transactions. A value of 0.10 was defined as the minimum support value. This value was kept constant throughout all the tests. It is important to mention that according to [6] a lower support level originates a longer the processing time of the Apriori algorithm since it becomes necessary to explore a larger number of *itemsets*. For each test, the time reducing rate was calculated considering the two parallel implementation versions of Apriori. In the first group of tests, two processors (MC1 and MC2) were considered, in the second group three processors (MC1, MC2 and MC3) were incorporated and finally, the third group considers the existence of four processors (MC1, MC2, MC3 and MC4).

2.2. METHODS

The Apriori algorithm was proposed by R. Agrawal and R. Srikant in 1994 for extraction of association rules in data mining. It is currently one of the most widely used algorithms in data mining and is considered a classic [6-9]. According to [10] its name derives from the fact that its method uses characteristics of a frequent pattern already found previously (prior) to search for more patterns. The Apriori is based on the AIS algorithm proposed by [11], but whose main limitation is the fact that the discovery of association rules is limited to only one item in the consequence of the rule. Therefore, the Apriori was the first algorithm to efficiently reduce the search space in the data, which substantially improved the performance in the discovery of association rules [6]. It improved the AIS since it made it possible to extract association rules with more than one item in the consequent of the rule.

Apriori is based on the principle that if an item set is frequent, then all subsets are also frequent, and uses a wide search strategy [12]. This principle is due to the following support property:

$$\forall X, Y : (X \sqsubseteq Y) \Longrightarrow S(X) \geq S(Y) \tag{3}$$

This way, the support of an *itemset* is never greater than the support of its subsets. This property is known as the support anti-monotonics [6]. This algorithm has an organization that guarantees great flexibility in the generation of association rules and its parameterization is made based on minimum support and minimum confidence [13-14].

The support of a two items series is given by the total number of database records in which the two items appear together. It is typically analyzed in percentage terms when considering the total of records. However, as the number of items in a data series increase, the possible combinations between these items increase exponentially, which leads to an increase in processing time. The solution as proposed by [15] is to reduce the number of items involved and, consequently, the combinations to be considered in each processing step. The definition of a minimum support eliminates the items that do not appear in a sufficient number of records to achieve the previously defined minimum criterion. Consequently, in each phase, there is verification of its support and this combination can be eliminated from the

process if it does not reach the minimum support, thus reducing the number of combinations that will be considered in the next phases [16]. The support can be calculated through:

$$Support = \frac{frq\ (X,Y)}{N} \tag{4}$$

Another measure used in the discovery of association rules is the confidence of the set of frequent items. In general terms, confidence measures the conditional probability of *Y* occurring given that *X* occurred. The confidence of a rule is given by:

$$Confidence = \frac{frq\ (X,Y)}{frq\ (X)} \tag{5}$$

In this way, a frequent item is understood to be an item that appears in large quantities on a dataset. In general terms, an item is considered frequent if it fulfills the following condition:

$$Support\ \{item\} \geq Minimal\ Support \tag{6}$$

The lift is also another relevant measure in the Apriori and is used to evaluate dependencies between the antecedent and consequent of the rule. In general, the higher the value of the lift, the more relevant the rule becomes because the greater the dependence between the items that constitute it [17]. Given a rule $X \rightarrow Y$, this measure indicates that the more frequent *Y* becomes, the more *X* occurs. The lift of a rule is given by the following formula:

$$Lift\ (\mathrm{X} \rightarrow \mathrm{Y}) = \mathrm{confidence}\ (\mathrm{X} \rightarrow \mathrm{Y}) / \mathrm{support}\ (\mathrm{Y}) = \frac{support\ (X \cup Y)}{support\ (X) * support\ (Y)} \tag{7}$$

If the lift value is equal to 1 then *X* and *Y* are independent; if the lift is positive then *X* and *Y* are positively dependent; otherwise, there is a negative dependency between the items. The lift is symmetrical, that is:

$$Lift\ (\mathrm{X} \rightarrow \mathrm{Y}) = Lift\ (\mathrm{Y} \rightarrow \mathrm{X}) \tag{8}$$

Apriori is based on two properties [18]: (i) all subseries of a frequent item must also be frequent, i.e. a set of items (*k-itemset*) can only be frequent if all its subsets (*'k-1'-itemset*) are frequent; and (ii) the frequency of a set of items never increases when an additional element is added.

It is important to look to the complexity of Apriori. Given *d* items, there are 2^d possible candidate *itemsets*. Therefore, the total number of *itemsets* is equal to 2^d and the total number of possible association rules is given by the expression below:

$$R = \sum_{k=1}^{d-1} \binom{d}{k} * \sum_{j=1}^{d-k} \binom{d-k}{j} \tag{9}$$

Apriori demands high computational performance, due to its large search space involving the generation of candidates for sets of frequent items. However, one of the great advantages of Apriori is its ease in being parallelized. The parallel version of the Apriori algorithm proposed by [19] has two fundamental objectives: (i) identification of frequent sets that present higher computational demand; and (ii) reduce the volume of communication between processors.

The tests performed by [19] show a significant reduction in processing time and demonstrate that the model can be scalable according to the number of distributed machines. However, the proposed model assumes that the distributed processing is homogeneous, i.e., all machines have the same processing power. This situation only occurs in specific cases and does not take advantage of the full potential of distributed processing, since in the phase of synchronization between machines, it is necessary to wait for the machine with the worst performance. In this sense, this study presents a proposal of a distributed Apriori algorithm for heterogeneous performance machines. This proposal can be mapped in four phases: (i) the data set is partitioned by the available *N* processors according to their processing capacity; (ii) in each iteration of the algorithm, each processor computes the frequency of the sets of items in the local partition of the data set; (iii) at the end of each iteration, a sum of the local frequencies of each set of items is performed, calculating the global frequency of each item. This step requires synchronization between the processors to disclose the local frequencies determined by them; and (iv) the sets of infrequent items, i.e., those that have support below the stipulated minimum, are discarded. The frequent items are used to generate sets of candidate items for the next iteration.

3. RESULTS AND DISCUSSION

Table 2 presents the results obtained considering the three considered scenarios. In each scenario, the number of included processors was changed. The findings allow us to conclude that there was a significant reduction in the processing time of the parallel version of the Apriori (Parallel Apriori v.1) concerning its original version. The rate of reduction of processing time is higher than 50%, especially when the number of processors increased in the parallel version of the algorithm. These results are aligned with the findings found by [20], in which a reduction rate close to 80% was found when the number of *itemsets* was 2500. In our tests, this rate of reduction was higher than 95% when we have 5000 itemsets, but they are sustained by two factors: (i) the time reducing rate increases with the number of *itemsets* considered; (ii) the processors considered in this test present higher processing speeds. The execution time of the original Apriori remains the same for both scenarios because there was no parallelism, and the *itemsets* were all executed only in MC1.

When we compare the two parallel versions, we realize smaller differences in the performance of the algorithm. In the parallel Apriori v.1 are considered homogeneous machines, while in parallel Apriori v.2 are considered heterogeneous machines. The performance of both algorithms is clearly superior to the original version of Apriori, but version 2.0 of parallel Apriori offers reduced processing times, particularly when the number of *itemsets* increases.

Table 2. Comparative analysis of the performance of algorithms

Itemsets	Original Apriori [sec.]	Parallel Apriori v.1 [sec.]	Parallel Apriori v.2 [sec.]	Time reducing rate between the two parallel versions [%]
MC1 and MC2				
1000	0.151	0.092	0.090	2.17
2000	0.283	0.127	0.122	3.94
3000	0.871	0.298	0.277	7.78
4000	1.892	0.681	0.638	6.31
5000	4.327	1.379	1.282	7.03
MC1, MC2 and MC3				
1000	0.151	0.071	0.069	2.82
2000	0.283	0.098	0.093	5.10
3000	0.871	0.124	0.113	8.87
4000	1.892	0.189	0.163	13.76
5000	4.327	0.356	0.297	16.57
MC1, MC2, MC3 and MC4				
1000	0.151	0.066	0.065	1.52
2000	0.283	0.081	0.075	7.41
3000	0.871	0.098	0.088	10.20
4000	1.892	0.125	0.101	19.20
5000	4.327	0.213	0.153	28.17

Furthermore, the findings allow us to conclude that as the number of processors increases, the time reducing rate also increases significantly. Figure 1 depicts this situation when the number of processors is increased. For 1000 *itemsets* the reduction rate is negligible (i.e., 1.52%), while this value rises significantly to 28.17% when we consider 5000 *itemsets*.

The obtained results consider only the use of the original Apriori algorithm without modifications. Over time, modified versions of Apriori like I-Apriori and T-Apriori have been introduced in which the presented performance is superior by reducing the number of candidate items and transactions [21, 22]. Additionally, the characteristics of the datasets were not explored according to the study carried out by [23], in which three datasets (i.e., spect heart database, primary tumor database, and mushroom database) are considered. The results obtained by [23] indicate some insignificant differences in processing time considering the characteristics of the datasets and, consequently, this feature has not been explored in this study.

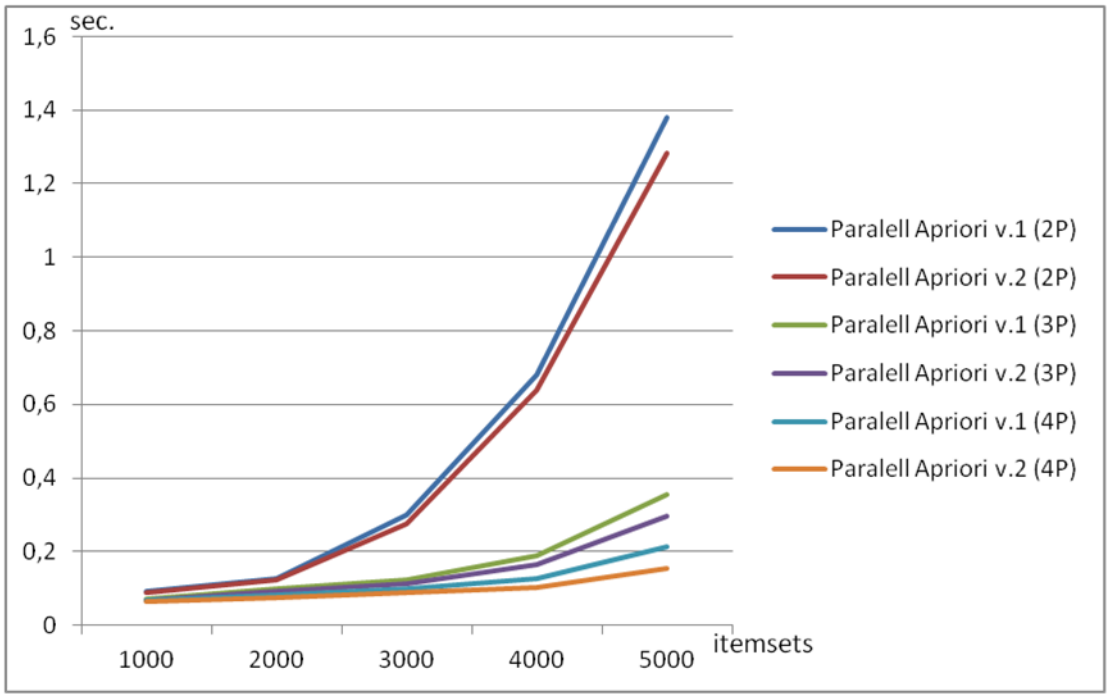

Figure 1. Comparative analysis of the parallel implementation of Apriori.

4. CONCLUSIONS

The Apriori algorithm is one of the best-known algorithms of association rules and with applications in the most diverse areas, particularly in e-commerce. Apriori can be divided into two stages: in a first stage, frequent *itemsets* are found, i.e., those with support higher than the minimum support, and rules should be generated from frequent items, i.e., those with confidence higher than the minimum confidence. To find rules that involve both frequent and rare items, the minimum support value should be relatively small. However, this can cause a combinatorial explosion in the number of *itemsets* which increases the number of considered transactions, which damages the performance of the Apriori algorithm.

One of the potentialities of the Apriori algorithm is its parallelization. This algorithm can be easily parallelized as shown in previous works in this field. However, the proposed models do not consider the heterogeneous characteristics of each machine. In this sense, the developed study proposes a parallelized heterogeneous model of the Apriori algorithm and analyzes its performance. The findings show that the heterogeneous Apriori version of the algorithm offers better performances than the traditional parallel versions of Apriori when there are processors with different processing speeds and characteristics. The time reducing rate increases as more *itemsets* are considered. However, for relatively low values of *itemsets*, the gains are not significant due to the I/O time spent in the process of synchronization between the performed tasks done by each processor.

This study offers both theoretical and practical impact. From a theoretical point of view, it provides a new approach to the parallelization of the Apriori when we have machines with different characteristics. From a practical point of view, the results obtained indicate a time reducing rate higher than 25% when we have four heterogeneous processors and 5000 *itemsets*. As future work, we intend to adopt the proposed model considering big data architectures like Hadoop and/or Apache Spark, in which the processing can be distributed among several geographically dispersed clusters.

REFERENCES

[1] Oprea, C., Popescu, D.M., Petrescu, A.G., Barbu, I., *Journal of Science and Arts*, **2**(39), 285, 2017.
[2] Florea, N.V., Duica, M.C., Mihai, D.C., Coman, D.M., *Journal of Science and Arts*, **4**(45), 989, 2018.
[3] Istrat, V., Lalic, N., *Applied Artificial Intelligence*, **31**(5-6), 543, 2017.
[4] More, N., Patil, S., *International Journal for Innovative Research in Science & Technology*, **1**(4), 71, 2014.
[5] Gonen, Y., Gudes, E., Kandalov, K., *Algorithms*, **11**(12), 194, 2018.
[6] Agrawal, P., Yadav, M.L., Anand, N., *International Journal of Computer Applications*, **74**(14), 6, 2013.
[7] Natingga, D., *Data Science Algorithms in a Week: Top 7 algorithms for scientific computing, data analysis, and machine learning*, 2nd Edition, Packt Publishing, 2018.
[8] Wang, J., *Journal of Software Engineering*, **11**, 219, 2017.
[9] Harun, N.A., Makhtar, M., Aziz, A., Zakaria, Z.A., Abdullah, F.S., Jusoh, J.A., *International Journal on Advanced Science, Engineering and Information Technology*, **7**(3), 764, 2017.
[10] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann Publishers, 2006.
[11] Agrawal, R., Imielinski, T., Swami, A., *Proceedings of the ACM SIGMOD International Conference on Management of Data*, **22**(2), 212, 1993.
[12] Gama, J., Carvalho, A., Faceli, K., Lorena, A., Oliveira, M., *Extração de Conhecimento de Dados – Data Mining*, 1st Edition, Edições Sílabo, 2012.
[13] Mohurle, S.D., Bogiri, N., *American International Journal of Research in Science, Technology, Engineering & Mathematics*, **26**(1), 113, 2019.
[14] Wang, X., Chen, M., Chen, L., *Cybernetics and Information Technologies*, **13**, 15, 2013.
[15] Tan, P.N., Steinbach, M., Karpatne, A., Kumar, V., *Introduction to Data Mining*, 2nd Edition, Pearson, 2018.
[16] Bhargava, M., Selwal, A., *International Journal of Advanced Research in Computer Science*, **4**(2), 328, 2013.
[17] Ali, F.M., Hamed, A., *Journal of Information and Telecommunication*, **2**(3), 236, 2018.
[18] Yabing, J., *International Journal of Computer and Communication Engineering*, **2**(1), 25, 2013.
[19] Ye, Y., Chiang, C., A Parallel Apriori Algorithm for Frequent Itemsets Mining, *Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, Seattle, WA, 89, 2006.
[20] Bhandari, A., Gupta, A., Das, D., *Procedia Computer Science*, **46**, 644, 2015.
[21] Aouad, L.M., Le-Khac, N., Kechadi, T.M., *Knowledge and Information Systems*, **23**(1), 55, 2010.
[22] Yuan, X., *AIP Conference Proceedings*, **1820**, 080005, 2017.
[23] Sinthuja, M., Puviarasan, N., Aruna, P., *World Applied Sciences Journal*, **35**(1), 43, 2017.