10.18605/2175-7275/cereus.v13n3p218-245

<< Recebido em: 16/09/2021 Aceito em: 01/10/2021. >>

ARTIGO ORIGINAL

REVISTA CEREUS ISSN: 2175-7275

Applying Methods to Minimize the Number of Association Rules that Fully Represent a Database

Aplicação de Métodos para Minimizar o Número de Regras de Associação que Represente Totalmente uma Base de Dados

Diego Paixão Pinheiro[1], Dr. Marcelo Lisboa Rocha[2]

[1] Student at Computer Science Department, Federal University of Tocantins.

E-mail: diego.pinheiro@uft.edu.br

[2] Professor Researcher at Computer Science Department and Postgraduate Program in Computational Modelling of Systems, Federal University of Tocantins.

E-mail: mlisboa@uft.edu.br

RESUMO

As regras de associação são uma forma de representação de conhecimento utilizada em sistemas de tomada de decisão devido à sua estrutura simples e ao alto potencial de armazenamento de informações. Essas características pode ser obtida através de algoritmos de mineração de regras de associação, como o *Apriori*, que toma um conjunto de dados como parâmetro de entrada e retorna um conjunto de regras de associação. Entretanto, os algoritmos existentes retornam um grande número de regras, o que torna o uso de regras de associação oneroso para sistemas de computacionais e muito difícil de interpretar para especialistas de domínio. A fim de superar esta dificuldade e facilitar a aplicação das regras de associação na solução de problemas de tomada de decisão, muitas pesquisas têm procurado uma solução computacional para reduzir a quantidade de regras de associação, de tal forma que não haja perda significativa de informações. Este artigo apresenta dois procedimentos computacionais para minimizar o número de regras de associação que representam plenamente um conjunto de dados. Em seguida, os autores apresentam os testes realizados e um estudo comparativo com outros métodos da literatura. Tendo em vista o sucesso alcançado, os autores fazem suas considerações sobre os resultados e apontam a nova direção do projeto.

Palavras-chave: Aprendizado de Máquina. Pesquisa Operacional. Análise Combinatória. Mineração de Dados. Regras de Associação.

ABSTRACT

Association rules are a form of knowledge representation used in decision making systems due to their simple structure and high information storage potential. This feature can be obtained through association rule mining algorithms, such as *Apriori*, which takes a dataset as an input parameter and returns a set of association rules. However, the existing algorithms return a large number of rules, which makes the use of association rules costly for computer systems and very hard to interpret for domain experts. In order to overcome this difficulty and facilitate the application of association rules in solving decision making problems, many researches have been searching for a computational solution to reduce the amount of association rules in such a way that there is no significant loss of information. This paper presents two computational procedures for minimizing the number of association rules that fully represent a dataset. Then, the authors present the tests performed and a comparative study with other methods in the literature. In view of the success achieved, the authors make their considerations about the results and point out the new direction of the project.

Keywords: Machine Learning. Operations Research. Combinatorial Analysis. Data Mining. Association Rules.

1. INTRODUTION

The acquisition of information enables structured and planned decision making, which is fundamental for the growth and success of an organization (BOURGEOIS, 2014.). Thanks to the Digital Revolution (DREYER, 2006), most of the information generated by professional or personal activities is in virtual environments (ANTONOPOULOS, 2010). Therefore, ways to extract and analyze information in databases have become essential.

The data mining emerged as the process of exploring large masses of data in search of consistent standards that assist in decision making (CHUNG, 1999; HE, 2009). In this context, the association rules consist of a hidden pattern representation model in a database through a list of previous and consequential events (KAMSU, 2013).This model can be extracted from a database using a data mining algorithm, one of the most popular being algorithm *Apriori* (AGRAWAL, 1994).

The structure of association rules offers a great capacity of knowledge representation and also provides an accessible reading for laymen in Computing, which favors its application in interdisciplinary problems (KAMSU, 2013; MIRABADI, 2010). However, association rule mining algorithms create a large number of rules (AGRAWAL, 1994), which makes manual observation of the information and its subsequent application by the domain expert unfeasible.

In order to overcome this characteristic, some research in Artificial Intelligence to Optimization (ECKER, 1988) has developed methods to minimize the amount of association rules. One of the works that obtained good results was the *BruteSuppression* algorithm, published in the paper "BruteSuppression - a size reduction method for apriori rule sets" (HILLS, BAGNALL, IGLESIA, RICHARDS, 2013). This algorithm was able to reduce the amount of association rules generated by the *Apriori* algorithm without significant loss of information. This made *BruteSuppresion* a reference for minimizing the amount of association rules in the literature (CHENG, 2016; HILLS, 2014).

However, this problem has not been fully exhausted and further research needs to be considered. In order to propose an alternative solution to the problem in question, the authors of this work researched and developed two methods - being a greedy heuristic and an exact algorithm - for minimizing the amount of association rules that fully represent a database. In addition, the authors tested the two proposed algorithms in the same context in which *BruteSuppression* was tested and, later, performed a comparative study between the performances of the two solutions in order to qualify the proposed solution.

The results of this work can have significant impact for new applications and research. First, the proposed algorithms will enable the application of association rules in decision making processes that previously did not accept association rules due to the high number of items or low knowledge representation of the rule set (HUNYADI, 2011). Furthermore, the proposed methods will contribute to new research projects that may add results presented to their background. Thus, the authors hope to contribute to the further study and use of association rules.

To this end, this paper has been structured as follows:

- **Background:** a theoretical review on association rules, set covering problem and related works;
- **Methods Developed:** an exposition about the methods developed in this work, the greedy method and the linear programming method;
- **Experimental and Computational Results:** exposition of the test results and a comparative study between the performance of the proposed methods and BruteSuppression.
- **Conclusions:** a dissertation on the final considerations and future works.

2. BACKGROUND

This section is dedicated to explaining the literature review used in this research. The three pillars of this work are Association Rules, the Set Coverage Problem, and the BruteSuppression reference paper.

2.1 ASSOCIATION RULES

Association rules is a knowledge representation model in which patterns are represented by means of a relationship between preceding and following events (AGRAWAL, 1993; AGRAWAL, 1994; HAN, KAMBER, 2006).

In order to elucidate the subject, some fundamental concepts must be defined:

- Let $T = \{t_1, t_2, ..., t_n\}$ a dataset storing a set of n transactions;
- Let the set of m itens $I = \{i_1, i_2, ..., i_m\}$ available to constitute each transaction $t_i \in T$, such that $t_i \subseteq I$.
- Let an *itemset* a set of items and a *k-itemset* an *itemset* with k items.

Now, consider two itemsets A and B, such that $A \subseteq I$ and $B \subseteq I$ and $A \cap B = \emptyset$. So, a t_i transaction contains the itemset A if, and only if, $A \subseteq t_i$.

In view of this, an association rule is an implication of the form: $A \rightarrow B$, in which $A \subseteq I$, $B \subseteq$ I and $A \cap B = \emptyset$. In this case, reads *A* implies *B*, *A* being the antecedent and *B* and the consequent of the rule.

In order to evaluate the association rules, several metrics were created. The two most important are addressed below: confidence and support. First, consider that the frequency of an itemset, denoted by *c*, is the number of *T* transactions that contain this itemset.

The *S* support of an association rule, $A \rightarrow B$, is the percentage of transactions which contain $A \cup B$ in relation to the overall of transactions *n* of *T*. This can be calculated:

$$S(A \rightarrow \mathrm{B}) = P(A \cup B) = \frac{c(A \cup \mathrm{B})}{n} \qquad (1)$$

The support therefore indicates the relative frequency of the rules. Therefore, the support determines the applicability of the rule.

The *c* confidence of an association rule is the percentage of transactions that contain A ∪ Bi *n* relation to all *T* transactions that contain *A*. It can be calculated as:

$$C(A \rightarrow \mathrm{B}) = P(A \cup B) = \frac{P(A \cup \mathrm{B})}{P(A)} = \frac{c(A \cup \mathrm{B})}{c(A)} \qquad (2)$$

Confidence indicates the ability to predict the rules. The rules with values high levels of confidence stand out qualitatively from the others, for the level of certainty of occurrence of the consequent of the rule, from the cases where its antecedent occurs. However, rules with low confidence do not offer estimation security and are therefore of limited use.

An association rule can be classified in terms of its coverage and accuracy as follows:

- **Strong:** when it has highly accuracy and covering many cases;
- **General**: when it has low accuracy and cover many cases;
- **Exception:** when it has highly accuracy and covering few cases.

The knowledge about association rules provides the computational foundation for this work, since this is the main object of this research.

2.2 SET COVERING PROBLEM

The Set Covering Problem (SPC) is an Integer Programming problem 0-1 and can described as: let *S*, a finite collection of finite sets; *T*, a sub-collection of *S*; and *E*, a finite

set. *T* covers *E* if every element of *E* be longs to some set of *T*. Therefore, the Set Covering Problem consists of finding a cover with minimum cost (CAPRARA, TOTH, FISCHETTI, 2000; IGNIZIO, CAVALIER, 1994).

In Figure 1, there is an example of the definition of the set covering problem.

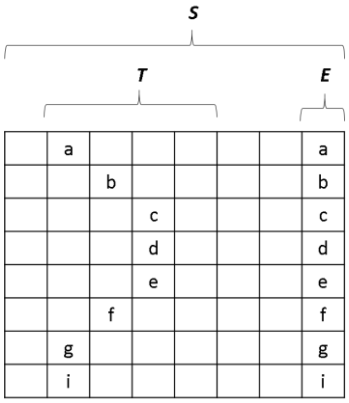

Figure 1. Illustrative example of S, T and E.

This problem can be represented by a mathematical programming model:

$$Minimize: \sum_{j=1}^{n} c_j x_j \qquad (3)$$

$$Subject\ to: \sum_{j=1}^{n} a_{ij} x_j \geq 1, \forall_i = 1, \ldots, m \qquad (4)$$

$$Where: x_j \in \{0, 1\}, \forall_j = 1, \ldots, n \qquad (5)$$

In this formulation, $x_j = 1$ if the column j is in solution and $x_j = 0$, otherwise; c is the cost of coverage; the in equation (4) concludes that each line of the matrix a_{ij} is covered by at least one column; and (5) is the binary restriction.

The Set Coverage Problem provides the mathematical basis for solving the problem in this paper, since the method developed aims to find a minimum amount of association rules that covers a complete data set.

2.3 RELATED PAPER

When starting this work, the authors of BruteSuppression aimed to develop an algorithm to reduce the number of association rules without causing significant loss of knowledge (HILLS, BAGNALL, IGLESIA, RICHARDS, 2013). To achieve this objective, the authors of this work developed two new measures called of Swing and Swing

Surprisingness, proposed a method to reduce the number of association rules and developed the BruteSuppresion algorithm. This will be discussed in more detail below.

The authors of BruteSuppresion start from the principle that good rules are rules that improve on the individual predictive power of the ATs in their antecedent and that rule is more interesting if it is composed of ATs that are poor individual predictors of the target class. In this sense, they propose two measures: Swing (an adaptation of relative surprisingness and confidence gain) and Swing Surprisingness (an adaptation of attribute surprisingness). These formulas can be defined as:

$$Swing(R): \sum_{i=1}^{n} \frac{Conf(R)Xn}{Conf(AT_i \rightarrow C)} \qquad (6)$$

$$SwingSurprisingness(R): \sum_{i=1}^{n} \frac{n}{Conf(AT_i \Rightarrow C)} \qquad (7)$$

Such that, for any rule ***R***, let ***ATi ⇒ C*** be the rule where the antecedent is the ***ith*** AT of ***R***, the consequent (***C***) is the consequent of ***R***, and rule ***R*** has ***n*** ATs. According to the authors of BruteSuppresion, these measures are particularly well-suited for data mining, as they reveal cases where combinations of poor predictors have yielded a good rule.

After that, the authors sought to measure the redundancy of the association rules. For this, the authors used the following measure found in the literature:

$$O(R,Q) = \frac{|D_R \cap D_Q|}{|D_R \cup D_Q|} \qquad (8)$$

Where ***R*** and ***Q*** are rules in terms of the records they cover (*DR and DQ*).

Soon after, Gebhardt's measure (GEBHARDT, 1991) was used to evaluate the similarity of the rules. These measure can be defined as:

$$V(Q) = V(Q) < (1+ \in)[S(R,Q)]\, V(R) \qquad (9)$$

Where *V* is some measure of rule interestingness, ∈ is a parameter for determining the intensity of the suppression and *S(R, Q)* some affinity function to measure the similarity of the rules.

The authors used 0.1 for ∈; ***O(R,Q)*** as our affinity function, as they wish to measure similarity in terms of overlapping coverage of records; and they use confidence for ***V***, as it

is the standard measure of the quality of a rule. The author's suppression function is as follows. Rule *R* suppresses rule ***Q*** if:

$$Conf(Q) < 1.1 \frac{|D_R \cap D_Q|}{|D_R \cup D_Q|} Conf(R) \qquad (10)$$

Where $|D_R \cap D_Q|$ is the number of records covered by both rule ***R*** and rule ***Q***, and $|D_R \cap D_Q|$ is the number of records covered by either rule ***R*** or rule ***Q***.

The BruteSuppression algorithm iterates through a rule set, testing pairs of rules with the suppression function and removing rules deemed to be redundant. He is shown below as Algorithm 1.

The BruteSuppression algorithm performed well in the experiments (HILLS, BAGNALL, IGLESIA, RICHARDS, 2013). Thus, the results of the BruteSuppression algorithm provide a reference for evaluating the results of the proposed algorithms in this work. From now, we call for short, BruteSupression as BS.

Algorithm 1 BruteSuppression

```
1:  procedure MAIN(Rule set R =< r_1, ..., r_n >, ordered by descending confifence (ID where confidence
    is equal), Data set D)
2:      i ← 2                                          ▷ Begin the process from the second rule
3:      ϵ ← 0.1
4:      while i ≤| R | do
5:          j ← i − 1          ▷ The first rule compared to rule i is the previous unsuppressed rule
6:          suppress ← false
7:          while j > 0 & !suppress do
8:              if Confidence(r_j) * |D_rj ∩ D_ri| / |D_rj ∪ D_ri| * (1 + ϵ) then
9:                  R ← R ∼ r_i                        ▷ Rule i is removed from the rule set
10:                 suppress ← true
11:             j − −
12:         if !suppress then
13:             i + +          ▷ The index is increased only if the previous rule was not suppressed
14:     Return R                                       ▷ Where R is the set of unsuppressed rules
```

Figure 2. BruteSuppression algorithm.

3 METHODS DEVELOPED IN THIS WORK

This section aims to present the computational methods proposed to solve the problem of minimizing the number of association rules that fully represent a database. In this sense, the authors developed two methods to address the problem: a greedy heuristic and an exact

algorithm based on integer linear programming. Thus, the logic and characteristics of the greedy heuristic and the exact algorithm is presented below.

3.1 GREEDY HEURISTIC

Greedy methods are simple methods of problem solving, where initially the elements that will be part of the solution are ordered according to the measure of interest (MICHALSKI, CARBONELL, MITCHELL, 2013). According to the specified order, the elements that will be part of the solution are chosen sequentially until a solution that satisfies all restrictions is built. Following this method, an algorithm to minimize the number of association rules covering the whole data set was developed in this work. From now on, it will be called GH.

This algorithm takes as input the rule set γ and the set of covered lines by each of these rules; and has as output the rule set that covers all the data lines **C**. The rule set containing the rule set **C** is initialized empty. Soon after r', set that stores the rows of the data set that are not covered by the **C** rules is initialized. On the following lines, the r_i sets that are used to store the lines in r' that are covered by rule **Xi** → **Y** are initialized. Then, iteratively, the rule in γ that matches the largest number of lines in r' is moved from the γ to the rule set coverage γ. This is repeated until all rows of the data set are covered by the rule set in **C**, this is, up to $\mathbf{r}' = \emptyset$**.**

Algorithm 2 GH

1: **procedure** MAIN(Rule set ← $\{X_i \rightarrow Y \mid i = 1,\ldots,n\}$, set of rules; and $(X_iY)\forall i \in \{1,\ldots,n\}, setofcoveredlines$)
2: $C \leftarrow \emptyset$ ▷ C rule coverage
3: $r' = \bigcup_{i=1}^{n} m(X_iY)$ ▷ Line not covered
4: **for** $i \in \{1,\ldots,n\}$ **do**
5: $r_i = m(X_iY)$
6: **while** $r' \neq \emptyset$ **do**
7: Choose $i \in \{1,..,n\}$, such that, $X_i \Rightarrow Y \in \gamma$ and $\mid r_i \mid$ is the biggest
8: $C = C \cup \{X_i \Rightarrow Y\}$ ▷ Adds the rule to coverage
9: $\gamma = \gamma \backslash \{X_i \Rightarrow Y\}$ ▷ Removes the rule from the original set
10: **for** $\{X_j \Rightarrow Y\} \in \gamma$ **do** ▷ Remove covered linese
11: $r_j = r_j \backslash m\{X_i \Rightarrow Y\}$
12: $r' = r' \backslash m(X_iY)$
13: Return C ▷ Where C is the rule converage

Figure 3. *HG* algorithm.

This greedy heuristic provides approximate solutions to the problem in question and has polynomial complexity in relation to the number of rules (**nr**), that is, **nr = |Y|** (DAVE, 1999).

3.2 METHOD BASED IN INTEGER LINEAR PROGRAMMING

This section presents the developed technique, based on mathematical programming. In this case, itis considered that the *Apriori* algorithm has already been executed on the data set ***D*** and the set of association rules ***R*** has been generated with ***nr*** rules. Here, the problem in finding the least number of rules that cover the entire data set used is the one specified, according to equations (3) to (5).

The relationship between a set of association rules and the Set Coverage Problem (SCP), described in equations 11 to 13, is as follows.

$$Minimize: \sum_{j=1}^{nr} x_j \qquad (11)$$

$$Subject\ to: \sum_{j=1}^{nr} \alpha_{kj} x_j \geq 1, \forall k = 1, .., m \qquad (12)$$

$$Where: x_j \in, \forall_j = 1, \ldots, nr \qquad (13)$$

The SCP's mathematical programming model presents ***m*** lines as a constraint, where each of them lines represents a line of data from the data set and x_j represents each of the generated association rules. In this model, the variable $\alpha_{kj} = 1$, if rule ***j*** covers the line ***k*** of the data set, and $\alpha_{kj} = 0$, otherwise. Thus, we seek to find the least number of association rules x_j that cover all data lines, which in this case is the optimal solution. As this method provides an exact solution to the problem, henceforth will be called EA.

4. EXPERIMENTAL AND COMPUTATIONAL RESULTS

This section is intended to discuss the results of the computational experiments. These experiments consist of the tests performed on the GH and EA procedures and the comparative study between the proposed procedures and BruteSuppression. In this context, the results obtained should be examined.

4.1 EXPERIMENTAL METHODOLOGY

In order to make a fair comparison between the algorithms developed in this work and BruteSuppressio*n*, the datasets and data processing used in testing the GH and EA algorithms are the same used in BruteSuppression's work.

Calculated on minimizing the number of association rules required to cover the data set, the methods proposed in this work and presented in section 3, were implemented in Java programming language, compiled in version 1.8.0.161 and running on an Intel I3-5005U 2GHz computer with 4Gb of RAM in the Windows 10 Enterprise Operating System.

The datasets used in this work are available in UC Machine Learnig Repository (FRANK, 2021) and their description follows below:

- **Adult**: Extraction was done by Barry Becker from the 1994 Census database. The person is described by means of socioeconomic data. Prediction task is to determine whether a person make saver 50K a year or not;
- **CreditApproval:** The source of this dataset is confidential. This file concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. The lines are classified as "+" or "-". In this dataset there is a good mix of attributes (continuous, nominal with small numbers of values, and nominal with larger numbers of values). There are also a few missing values;
- **HouseVote:** The source of this dataset is "Congressional Quarterly Almanac, 98th Congress, 2ndsession 1984, Volume XL: Congressional Quarterly In". This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified tonay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition). The records are classified as "Republican" or "Democrat";
- **Mushrooms:** The source of this dataset is the Audobon Society Field Guide. The mushrooms are described in terms of physical characteristics and classified as poisonous or edible;
- **Tic-Tac-Toe:** This dataset was created by David W. Aha. This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). Thus, the records are classified as "positive" or "negative".

The data processing was performed using the WEKA API software (HOLMES, DONKIN, WITTEN, 1994) and consisted of the following tasks:

- Data discretization by means of the Fayyad & Irani's MDL (Minimum Description Length) method (FAYYAD, IRANI, 1993);
- Deletion of data lines with missing elements;
- For each of the 5 datasets presented in Table 1, we split in 10 masses for training and testing, where the random partition of dataset was 65% for training and 35% for testing.

Then, we use the proposed methods of minimizing the number of association rules (GH and EA) to process the data. Table 1 shows the identity of 12 tests performed, the datasets used and the parameters according to the experiments performed in BruteSuppression work.

Furthermore, the authors ran the two algorithms with the same datasets and new parameters. The new parameters differ from the previous ones in the minimum support, as the new parameters have a value approximately equal to 1 divided by the number of rows (# Rows) each of the training files, which forces the algorithms to consider all association rules in the process. This improves the coverage of the resulting association rules over the database. In total 8 new tests were generated, as shown in the Table 2, making a total of 20 tests.

Each test presented on Table 1 and Table 2 comprises the average results of 10 runs performed with different data samples. In this manner, we can assure the confidence and unbiasedness of the results.

Table 1. List of the Test with BruteSuppression parameters.

Test	Dataset	Minimum Support	Minimum Confidence
1	Adult	0.02	0.25
2	Adult	0.02	0.44
3	Adult	0.05	0.25
4	CreditApproval	0.1	0.42
5	CreditApproval	0.1	0.75
6	CreditApproval	0.3	0.42
7	HouseVotes	0.3	0.4
8	HouseVotes	0.3	0.9
9	HouseVotes	0.37	0.4
10	Mushroom	0.2	0.63
11	Mushroom	0.3	0.63
12	Tic-Tac-Toe	0.05	0.35

Table 2. Test List with the Parameters of the Authors of this Paper.

Test	Dataset	# Rows	Minimum Support	Minimum Confidence
13	Adult	33345	0.00003	0.25
14	Adult	33345	0.00003	0.44
15	CreditApproval	500	0.002	0.42
16	CreditApproval	500	0.002	0.75
17	HouseVotes	286	0.0035	0.4
18	HouseVotes	286	0.0035	0.9
19	Mushroom	5000	0.0002	0.63
20	Tic-Tac-Toe	625	0.0016	0.35

4.2 COMPUTATIONAL RESULTS OF THE PROPOSED METHODS

In order to ensure the reliability of the evaluation of GH and EA methods, this paper has established two evaluation sources certified by the scientific literature. Thus, an acknowledgement of these chosen sources of evaluation follows.

The first evaluation source has been formulated specifically for the GH and EA methods and is called "Computational Results". This is a table that presents characteristic parameters of the proposed methods and allows to examine the effectiveness and efficiency of the algorithms. The elucidation of the parameters should be considered below:

- **Number of rules generated:** amount of association rules generated by the *Apriori* algorithm that will be processed by the proposed methods;
- **Number of rules of the Greedy Heuristics***:* amount of association rules used by the greedy heuristic to create the coverage;
- **Number of line not covered by Greedy Heuristics:** amount of line that the greedy heuristic could not represent;
- **Greedy Heuristics time:** the time taken by greedy heuristics to perform the processing;
- **Number of rules of Exact Algorithm***:* amount of association rules used by the exact algorithm to create the coverage;•Exact Algorithm time: time taken by exact algorithm to perform the processing;

The second source of evaluation is confusion matrices. These contingency tables allow the visualization of the performance of the classification algorithms, as well as the generation of more specific performance measures. For this reason, two confusion matrices will be presented for each test, one for each method (GH and EA).

Below you can see the table of computational results (Table 3 to Table 26) and the confusion matrices for the 12 tests performed with the parameters extracted from the BruteSuppression paper as shown in Table 1.

Table 3. Computational Results of Test 1.

Description	Value
Number of rules generated	38308.3
Number of rules of the Greedy Heuristics	10.5
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.769084587
Number of rules of Exact Algorithm	9
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	1.703767903

Table 4. Results of Confusion Matrix of Test 1 with GH and EA.

	> 50k	**≤ 50k**
> 50 k	3930.8	0.5
≤ 50 k	0	11896.6

(a) Test 1 with GH

	> 50k	**≤ 50k**
> 50 k	3929.84	1.6
≤ 50 k	0	11896.6

(b) Test 1 with EA

Table 5. Computational Results of Test 2.

Description	Value
Number of rules generated	34610.1
Number of rules of the Greedy Heuristics	10.2
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	1.180693589
Number of rules of Exact Algorithm	10
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	1.199886843

Table 6. Results of Confusion Matrix of Test 2 with GH and EA.

	> 50k	**≤ 50k**
> 50 k	3769.2	162.
≤ 50 k	0	11896.6

(a) Test 1 with GH

	> 50k	**≤ 50k**
> 50 k	3769.6	161.8
≤ 50 k	0	10896.6

(b) Test 1 with EA

Table 7. Computational Results of Test 3.

Description	Value
Number of rules generated	7481.9
Number of rules of the Greedy Heuristics	10.3
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.457412821
Number of rules of Exact Algorithm	8.9
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.686791299

Table 8. Results of Confusion Matrix of Test 3 with GH and EA.

	> 50k	≤ 50k
> 50 k	3930.9	0.5
≤ 50 k	0	11896.6

(a) Test 1 with GH

	> 50k	≤ 50k
> 50 k	3930.1	1.3
≤ 50 k	0	11896.6

(b)Test 1 with EA

Table 9. Computational Results of Test 4.

Description	Value
Number of rules generated	9345.5
Number of rules of the Greedy Heuristics	7.6
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	3.510290735
Number of rules of Exact Algorithm	4.1
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	3.060779466

Table 10. Results of Confusion Matrix of Test 4 with GH and EA.

	+	-
+	127.5	0
-	0.6	106.9

(a) Test 1 with GH

	+	-
+	126.9	0.6
-	0.6	106.9

(b)Test 1 with EA

Table 11. Computational Results of Test 5.

Description	Value
Number of rules generated	7510.1
Number of rules of the Greedy Heuristics	6.5
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	3.115556828
Number of rules of Exact Algorithm	6.5
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	1.561144443

Table 12. Results of Confusion Matrix of Test 5 with GH and EA.

	+	-
+	121.8	5.7
-	6.1	101.4

(a) Test 1 with GH

	+	-
+	121.8	5.7
-	6.1	101.4

(b)Test 1 with EA

Table 13. Computational Results of Test 6.

Description	Value
Number of rules generated	159.5
Number of rules of the Greedy Heuristics	7.2
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.024701827
Number of rules of Exact Algorithm	4.9
Number of lines not covered by Exact Algorithm	0.1
Exact Algorithm time	0.078000756

Table 14. Results of Confusion Matrix of Test 6 with GH and EA.

	+	-
+	127.5	0.5
-	0.7	106.8

(a) Test 1 with GH

	+	-
+	127.1	0.4
-	1.4	106

(b)Test 1 with EA

Table 15. Computational Results of Test 7.

Description	Value
Number of rules generated	2709.8
Number of rules of the Greedy Heuristics	5.1
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.07585122
Number of rules of Exact Algorithm	4.6
Number of lines not covered by Exact Algorithm	0.2
Exact Algorithm time	0.06834656

Table 16. Results of Confusion Matrix of Test 7 with GH and EA.

	Democrat	Republican
Democrat	37.7	1.7
Republican	0.2	30.3

(a) Test 1 with GH

	Democrat	Republican
Democrat	37.2	2.2
Republican	0.5	30

(b)Test 1 with EA

Table 17. Computational Results of Test 8.

Description	Value
Number of rules generated	2275.6
Number of rules of the Greedy Heuristics	2.8
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.080297401
Number of rules of Exact Algorithm	2.8
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.069323304

Table 18. Results of Confusion Matrix of Test 8 with GH and EA.

	Democrat	Republican
Democrat	37.2	2.2
Republican	0.1	30.5

(a) Test 1 with GH

	Democrat	Republican
Democrat	37.1	2.3
Republican	0.1	30.5

(b)Test 1 with EA

Table 19. Computational Results of Test 9.

Description	Value
Number of rules generated	74.6
Number of rules of the Greedy Heuristics	6.1
Number of line not covered by Greedy Heuristics	15.2
Greedy Heuristics time	0.021808412
Number of rules of Exact Algorithm	5.2
Number of lines not covered by Exact Algorithm	16.5
Exact Algorithm time	0.058233207

Table 20. Results of Confusion Matrix of Test 9 with GH and EA.

	Democrat	Republican
Democrat	92.6	0.9
Republican	5.4	38.7

(a) Test 1 with GH

	Democrat	Republican
Democrat	92.4	0.9
Republican	4.5	38.7

(b)Test 1 with EA

Table 21. Computational Results of Test 10.

Description	Value
Number of rules generated	26878.2
Number of rules of the Greedy Heuristics	5.4
Number of line not covered by Greedy Heuristics	0.188007033
Greedy Heuristics time	1.693473604
Number of rules of Exact Algorithm	4
Number of lines not covered by Exact Algorithm	0.110741618
Exact Algorithm time	1.414844289

Table 22. Results of Confusion Matrix of Test 10 with GH and EA.

	p	e
p	533.5	367.7
e	0	1213.857143

(a) Test 1 with GH

	p	e
p	533.5	367.7
e	0	1213.857143

(b)Test 1 with EA

Table 23. Computational Results of Test 11.

Description	Value
Number of rules generated	10377.5
Number of rules of the Greedy Heuristics	4.4
Number of line not covered by Greedy Heuristics	0.166678419
Greedy Heuristics time	1.294086855
Number of rules of Exact Algorithm	3
Number of lines not covered by Exact Algorithm	0.057116619
Exact Algorithm time	1.04512526

Table 24. Results of Confusion Matrix of Test 11 with GH and EA.

	p	e
p	462.9	438.3
e	0	1213.857

(a) Test 1 with GH

	p	e
p	462.9	438.3
e	0	1213.857

(b)Test 1 with EA

Table 25. Computational Results of Test 12.

Description	Value
Number of rules generated	486.2
Number of rules of the Greedy Heuristics	10.2
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.037830151
Number of rules of Exact Algorithm	6
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.089093485

Table 26. Results of Confusion Matrix of Test 12 with GH and EA.

	1	2
1	117.2	0
2	0	220.8

(a) Test 1 with GH

	1	2
1	117.2	0.5
2	0	220.8

(b)Test 1 with EA

Below you can see the tables of computational results and the confusion matrices (Table 27 to Table 42) for the 8 tests performed with the parameters specified by the authors of this paper as presented in Table 2.

Table 27. Computational Results of Test 13.

Description	Value
Number of rules generated	8400
Number of rules of the Greedy Heuristics	4.7
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.59913164
Number of rules of Exact Algorithm	4.4
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.540488069

Table 28. Results of Confusion Matrix of Test 13 with GH and EA.

	> 50k	≤ 50k
> 50 k	3433.3	498.1
≤ 50 k	0	11896.6

(a) Test 1 with GH

	> 50k	≤ 50k
> 50 k	3933.3	498.1
≤ 50 k	0	11896.6

(b)Test 1 with EA

Table 29. Computational Results of Test 14.

Description	Value
Number of rules generated	3000
Number of rules of the Greedy Heuristics	7
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.336056925
Number of rules of Exact Algorithm	6.9
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.567112094

Table 30. Results of Confusion Matrix of Test 14 with GH and EA.

	> 50k	≤ 50k
> 50 k	3925.1	6.3
≤ 50 k	0	11896.6

(a) Test 1 with GH

	> 50k	≤ 50k
> 50 k	3925	6.4
≤ 50 k	0	11896.6

(b)Test 1 with EA

Table 31. Computational Results of Test 15.

Description	Value
Number of rules generated	30000
Number of rules of the Greedy Heuristics	7.4
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.469755529
Number of rules of Exact Algorithm	6.1
Number of lines not covered by Exact Algorithm	0.2
Exact Algorithm time	0.418782068

Table 32. Results of Confusion Matrix of Test 15 with GH and EA.

	+	-
+	128.3333	0.
-	2.3333	104.3333

(a) Test 1 with GH

	+	-
+	126.1	1.2
-	3.2222	104.2222

(b)Test 1 with EA

Table 33. Computational Results of Test 16.

Description	Value
Number of rules generated	30000
Number of rules of the Greedy Heuristics	5.1
Number of line not covered by Greedy Heuristics	1.9
Greedy Heuristics time	0.567179852
Number of rules of Exact Algorithm	5.1
Number of lines not covered by Exact Algorithm	1.9
Exact Algorithm time	0.182280482

Table 34. Results of Confusion Matrix of Test 16 with GH and EA.

	+	-
+	127.5	0
-	0.6	106.9

(a) Test 1 with GH

	+	-
+	126.9	0.6
-	0.6	106.9

(b)Test 1 with EA

Table 35. Computational Results of Test 17.

Description	Value
Number of rules generated	30000
Number of rules of the Greedy Heuristics	3.2
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.530311051
Number of rules of Exact Algorithm	3.1
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.167545992

Table 36. Results of Confusion Matrix of Test 17 with GH and EA.

	Democrat	Republican
Democrat	37.4	2
Republican	0.1	30.5

(a) Test 1 with GH

	Democrat	Republican
Democrat	37.1	2.3
Republican	0.1	30.5

(b)Test 1 with EA

Table 37. Computational Results of Test 18.

Description	Value
Number of rules generated	30000
Number of rules of the Greedy Heuristics	2.6
Number of line not covered by Greedy Heuristics	3
Greedy Heuristics time	0.507424552
Number of rules of Exact Algorithm	2.6
Number of lines not covered by Exact Algorithm	3
Exact Algorithm time	0.17195856

Table 38. Results of Confusion Matrix of Test 18 with GH and EA.

	Democrat	Republican
Democrat	37.1	2.2
Republican	0.4	27.3

(a) Test 1 with GH

	Democrat	Republican
Democrat	37.1	2.2
Republican	0.4	27.3

(b)Test 1 with EA

Table 39. Computational Results of Test 19.

Description	Value
Number of rules generated	156001.2
Number of rules of the Greedy Heuristics	93
Number of line not covered by Greedy Heuristics	47.3173453
Greedy Heuristics time	52.7807242
Number of rules of Exact Algorithm	53.88888889
Number of lines not covered by Exact Algorithm	43.18557272
Exact Algorithm time	2.55762542

Table 40. Results of Confusion Matrix of Test 19 with GH and EA.

	p	e
p	411.3	405.8
e	0	1150.571

(a) Test 1 with GH

	p	e
p	411.3	405.8
e	0	1150.571

(b)Test 1 with EA

Table 41. Computational Results of Test 20.

Description	Value
Number of rules generated	47188.4
Number of rules of the Greedy Heuristics	10.2
Number of line not covered by Greedy Heuristics	0
Greedy Heuristics time	0.428223365
Number of rules of Exact Algorithm	6
Number of lines not covered by Exact Algorithm	0
Exact Algorithm time	0.529045692

Table 42. Results of Confusion Matrix of Test 20 with GH and EA.

	1	2
1	117.2	0
2	0	218.8

(a) Test 1 with GH

	1	2
1	117.2	0.5
2	0	218.8

(b)Test 1 with EA

Once analyzed the tables, consider the following points about the proposed methods:

- The methods were compiled and tested on a computer with modest computational resources as explained in section 4.1. Despite this, the methods accomplished their task

in a considerably short time, which attests to their efficiency;

- The methods were able to reduce the amount of association rules from the range of thousands to tens, which shows their effectiveness in minimizing the number of rules;
- In some cases, the greedy heuristic (GH) and Exact Algorithm (EA) failed to achieve full coverage. But, both got reduce the number of final rules related to BruteSupression (BS);
- In some tests the confusion matrices of the proposed methods show false positives and false negatives. The original set of association rules provided by *Apriori* fully represents the database, but not all association rules provide full confidence. The proposed methods prioritize rules with full reliability by minimizing the number of association rules, but the original set does not always provide rules with 100% confidence. Consequently, the proposed methods are able to provide a minimal set that fully covers the database with a significant, but not always perfect confidence due to the rules originally generated by *Aprori*. Nevertheless, it should be considered that the algorithms presented a very high hit rate.

It is therefore understood that the methods for minimizing the number of association rules that fully cover a database showed an excellent result with regard to efficiency and effectiveness.

4.3 ADDITIONAL RESULTS AND COMPARISONS

In the literature about association rules there are many works that try to minimize the amount of association rules. In this sense, BruteSuppression, already presented in subsection 2.3, got good results, which was the state of art in literature until now. Thus, a comparison between the proposed methods (GH and EA) and BruteSuppression should be considered in order to validate the proposed methods against other solutions already developed.

In the tables below (Table 43 to Table 44), is possible to observe the performance of the algorithms with respect to reducing the amount of association rules. Remembering that each test was performed as stated in Table 2 and the results presented are the average of 10 runs, where PR is the mean percentage of reduction between Size Before and Size After in each test for each method.

Table 43. BruteSuppression's performance on reduction.

Rule Set	Max AT's	Size Before	Size After	PR
CreditApproval	3	178	61	65.73%
CreditApproval	7	388	61	84.28%
HouseVotos	3	269	63	76.58%
HouseVotos	7	428	61	85.77%
Mushroom	3	184	12	93.48%
Mushroom	7	519	12	97.69%

Table 44. BruteSuppression's performance on reduction.

Rule Set	Size Before	GH's Size After	GH's PR	EA's Size After	EA's PR
Test 1	38308.3	10.5	99.97%	9	99.98%
Test 2	34610.1	10.2	99.97%	10	99.97%
Test 3	7481.9	10.3	99.86%	8.9	98.88%
Test 4	9345.5	7.6	99.92%	4.1	99.96%
Test 5	7510.1	6.5	99.91%	6.5	99.91%
Test 6	159.5	7.2	95.49%	4.9	96.93%
Test 7	2709.8	5.1	99.81%	4.6	99.83%
Test 8	2275.6	2.8	99.88%	2.8	99.88%
Test 9	74.6	6.1	91.82%	5.3	92.90%
Test 10	26878.2	5.4	99.97%	4	99.99%
Test 11	10377.5	4.4	99.95%	3	99.97%
Test 12	486.2	10.2	97.90%	6	98.77%

Based on the results of the experiment as show in Table 43 and Table 44 for BruteSupression (BS) and the proposed methods of Greedy Heuristic (GH) and Exact Algorithm (EA), we can compare the performance of them, using the Size After metric presented and the correspondent PR (perceptual of reduction). We analyzed the results of the experiment using selected statistical methods. The statistical significance of experimental results on Table 43 and Table 44 is obtained by performing Kruskal-Wallistest and post-hoc Conover's test (CONOVER, 1999).

To test whether the differences in Size After values were significant by the different methods studied, we used the Kruskal-Wallis test, ANOVA's non-parametric counterpart, at a critical level ofα=0.01. The results showed significant differences in the median Size After values of each method (Kruskal-Wallis test$H$ = 15.649, p-value = 0.0003997470«0.01), rejecting the null hypothesis. In Table 45, we perform a post-hoc analysis with Conover's test with the Holm adjustment, which reveals a significant difference among proposed

methods (GH and EA) and the literature method (BS), with p-value«0.01 for EA against BS (2.603997e-09) and GH against BS (4.128762e-07).

Table 45. Results of post-hoc test over considered methods.

	BS	EA	GH
BS	1.000000e+00	2.603997e-09	4.128762e-07
EA	2.603997e-09	1.000000e+00	6.044429e-02
GH	4.128762e-07	6.044429e-02	1.000000e+00

To compare the performance among the three methods, we plot the mean rank in Fig. 2, where lower is better.

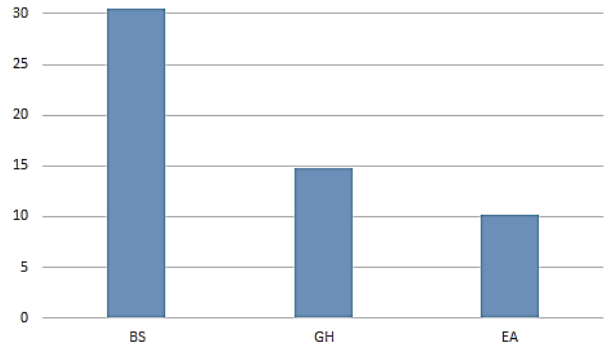

Figure 4. Mean rank plot for all methods.

Analyzing the mean rank plot values of Figure 2 over the reduction of association rules number (size before against size after), is possible to observe that EA and GH present better performance than BS, and that EA perform slightly better than GH.

About the two methods is possible to do following analysis:

- The BruteSuppression (BS) algorithm was able to achieve a large reduction in the amount of association rules in all tested sets, in the range between 65.7% and 96.9%. The proposed methods(HG and EA) performed even better, reducing the amount of association rules in the ranges between91.82% and 99.98% for HG and 92.90% and 99.99% for EA;
- The BS algorithm limits the amount of association rule antecedents (AT), whose value is shown in the ”Max ATs” column of Table 43, to be processed. In the proposed algorithms there is no such limitation, which allows any association rule to be added to the set that will be processed;
- The BS algorithm showed no results for the Tic-Tac-Toe set. The proposed methods achieved a reduction of 97.90% (GH) and 98.88% (EA).

Therefore, the performance of the proposed procedures (GH and EA) were superior to the performance of BruteSuppression.

5. CONCLUSIONS

Here is performed a brief discussion of the results and pointing out future work.

In view of the success obtained by the proposed procedures, the following points can be considered.

- The GH and EA procedures were designed on a consolidated mathematical and computational foundation and developed with appropriate technologies;
- This infers that effectiveness and efficiency obtained from applications in BruteSuppression contexts is understood for application in any context.

In this way, the authors deliver a contribution to the increased use of association rules in studies and decision-making applications, presenting methods that reduce the number of association rules that represent a dataset. This turn the decision-making process easier and faster.

5.1 FUTURE WORKS

The proposed procedures were applied in the same context as BruteSuppression to allow a comparative study and its consequent qualification against the Association Rules literature. Going forward, the authors intend to perform further comparative studies with algorithms present in the literature that will use other datasets and mined the association rules by means of algorithms different from Apriori. Thus, the GH and EA can be empirically consolidated.

REFERÊNCIAS

AGRAWAL, R., IMIELIŃSKI, T., and SWAMI**,** A**. Mining association rules between sets of items in large databases**. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (1993), pp. 207–216.

AGRAWAL, R., SRIKANT, R., et al. **Fast algorithms for mining association rules**. In Proc. 20th Int. Conf. Very Arge Data Bases, VLDB (1994), vol. 1215, pp. 487–499.

ANTONOPOULOS, N., and GILLAM, L. **Cloud computing**. Springer, 2010.

BOURGEOIS, D. **Information systems for business and beyond**. The Saylor Foundation, 2014.

CAPRARA, A., TOTH, P., and FISCHETTI, M. **Algorithms for the set covering problem**. Annals of Operations Research 98, 1 (2000), 353–371.

CHENG, M., XU, K., and GONG, X. **Research on audit log association rule mining based on improved apriori algorithm**. In 2016 IEEE International Conference on Big Data Analysis (ICBDA)(2016), IEEE, pp. 1–7.

CHUNG, H. M., and GRAY, P. **Data mining**. Journal of management information systems 16, 1(1999), 11–16.

CONOVER, W. J**. Practical Nonparametric Statistics**, 3. Ed. John Wiley, New York, NY, 1999.

DAVE, P. H. **Design and analysis of algorithms**. Pearson Education India, 2007.

DREYER, K. J., HIRSCHHORN, D., THRALL, J. H., and PACS, M. **A guide to the digital revolution**. Springer, 2006.

ECKER, J. G., KUPFERSCHMIND, M., et al. **Introduction to operations research**. Wiley New York, 1988.

FAYYAD, U. M., and IRANI, K. B. **Multi-interval discretization of continuous-valued attributes for classification learning**. In IJCAI (1993), pp. 1022–1029.19.

FRANK, A. **UCI Machine Learning Repository**. Available in: <http://archive.ics.uci.edu/ml>. Accessed in: August 01, 2021.

GEBHARDT, F. **Choosing among competing generalizations**. *Knowledge Acquisition, 3*(4), 361–380, 1991.

HAN, J., and KAMBER, M. **Data mining concepts and techniques.** Morgan Kaufmann Publishers, 2006.

HE, J. **Advances in data mining: History and future**. In 2009 Third International Symposium on Intelligent Information Technology Application (2009), vol. 1, IEEE, pp. 634–636.

HILLS, J., BAGNALL, A., DE LA IGLESIA, B., and RICHARDS, G. **BruteSuppression: a size reduction method for apriori rule sets**. Journal of intelligent information systems 40, 3 (2013), 431–454.

HILLS, J. F. **Mining time-series data using discriminative subsequences**. PhD thesis, University of East Anglia, 2014.

HOLMES, G., DONKIN, A., AND WITTEN, I. H. **Weka: A machine learning workbench**. In Proceedings of ANZIIS'94-Australian New Zea lnd Intelligent Information Systems Conference (1994), IEEE, pp. 357–361.

HUNYADI, D. **Performance comparison of apriori and fp-growth algorithms in generating association rules**. In Proceedings of the European computing conference (2011), pp. 376–381.

IGNIZIO, J. P., AND CAVALIER, T. M. **Linear programming**. Prentice-Hall, Inc., 1994.

KAMSU-FOGUEM, B., RIGAL, F., AND MAUGET, F. **Mining association rules for the quality improvement of the production process**. Expert systems with applications 40, 4 (2013), 1034–1045.

MICHALSKI, R. S., CARBONELL, J. G., AND MITCHELL, T. M. **Machine learning: An artificial intelligence approach**. Springer Science & Business Media, 2013.

MIRABADI, A., and SHARIFIAN, S. **Application of association rules in iranian railways (rai) accident data analysis**. Safety Science 48, 10 (2010), 1427–1435.