Knowledge-Based Systems 153 (2018) 176–192

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

MRQAR: A generic MapReduce framework to discover quantitative association rules in big data problems

D. Martín[1,a], M. Martínez-Ballesteros[*,1,b], D. García-Gil[c], J. Alcalá-Fdez[c], F. Herrera[c,d], J.C. Riquelme-Santos[b]

[a] Department of Artificial Intelligence and Infrastructure of Informatics Systems, Technological University of Havana J.A Echeverría, La Habana, Cuba
[b] Department of Computer Science, University of Seville, Seville, Spain
[c] Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[d] Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah, Saudi Arabia

ARTICLE INFO

Keywords:
Quantitative association rules
Multiobjective evolutionary algorithms
Big Data
MapReduce
Spark

ABSTRACT

Many algorithms have emerged to address the discovery of quantitative association rules from datasets in the last years. However, this task is becoming a challenge because the processing power of most existing techniques is not enough to handle the large amount of data generated nowadays. These vast amounts of data are known as Big Data. A number of previous studies have been focused on mining boolean or nominal association rules from Big Data problems, nevertheless, the data in real-world applications usually consist of quantitative values and designing data mining algorithms able to extract quantitative association rules presents a challenge to workers in this research field. In spite of the fact that we can find classical methods to discover boolean or nominal association rules in the most well-known repositories of Big Data algorithms, such repositories do not provide methods to discover quantitative association rules. Indeed, no methodologies have been proposed in the literature without prior discretization in Big Data. Hence, this work proposes MRQAR, a new generic parallel framework to discover quantitative association rules in large amounts of data, designed following the MapReduce paradigm using Apache Spark. MRQAR performs an incremental learning able to run any sequential quantitative association rule algorithm in Big Data problems without needing to redesign such algorithms. As a case study, we have integrated the multiobjective evolutionary algorithm MOPNAR into MRQAR to validate the generic MapReduce framework proposed in this work. The results obtained in the experimental study performed on five Big Data problems prove the capability of MRQAR to obtain reduced set of high quality rules in reasonable time.

1. Introduction

In data mining, the discovery of interesting relations in the data is a frequent used technique known as association rules. Association rules can be expressed as $A \longrightarrow C$, where A and C are items of couples attribute-value and satisfy that $A \cap C = \varnothing$. Many proposals can be found in the literature to address the discovery of association rules in datasets with numerical values, known as quantitative association rules (QARs). Many of them are based on Evolutionary Algorithms (EA) due to the good performance presented in problems with complex search spaces.

However, the rule-matching process is the most costly phase of these algorithms in terms of execution time since need to process expensive fitness functions a high number of times. The evaluation of each solution requires the processing of all records in the dataset. Alternatively, other weakness usually presented is the memory consumption. The problem is especially emphasized when these algorithms have to handle large-scale datasets [1]. In general, the knowledge extraction process becomes a difficult and complex task since in many cases, the amount of generated data exceeds the processing capability of conventional systems. The design of efficient algorithms able to process and analyze this amount of data is becoming a big challenge to researchers. One of the most used approaches is the MapReduce programming model [2–4] that is a robust and effective computational paradigm to process big datasets in distributed environments. This programming model divides a problem into smaller and affordable subproblems and combines partial solutions to obtain the final result.

* Corresponding author.
E-mail addresses: dmartin@ceis.cujae.edu.cu (D. Martín), mariamartinez@us.es (M. Martínez-Ballesteros), djgarcia@decsai.ugr.es (D. García-Gil), jalcala@decsai.ugr.es (J. Alcalá-Fdez), herrera@decsai.ugr.es (F. Herrera), riquelme@us.es (J.C. Riquelme-Santos).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2018.04.037
Received 27 September 2017; Received in revised form 27 April 2018; Accepted 28 April 2018
Available online 30 April 2018

Several frameworks have emerged in the last years to tackle Big Data [5,6]. One of them is the open source cluster computing framework named Apache Spark [7,8] which is a fast and general engine for large-scale data processing based on in-memory computation.

Several authors have focused on redesigning classical methods such as Apriori or FP-Growth to discover boolean or nominal association rules in massive data. Indeed, the Machine Learning library (MLlib) [9] of Apache Spark provides a parallel version of the FP-Growth algorithm. Nevertheless, such methods are only able to deal with boolean or nominal values which require a previous discretization to be applied in quantitative domains leading to information loss. Thus, it is necessary to redesign the existing techniques to address the QAR discovery in large scale datasets but this requires a complex, expensive and specialised process for researchers. Our aim is to propose a general purpose methodology able to apply any existing algorithm to find QAR avoiding their adaptation or redesign to handle Big Data.

Hence, this work presents MRQAR, a new generic parallel framework to discover quality QARs from Big Data problems using any non-parallel algorithm to discover QAR. In particular, MRQAR is based on the MapReduce paradigm and uses Apache Spark due to the good performance presented handling large scale datasets. This proposal presents an incremental learning scheme that discovers QARs by four phases, three of which are MapReduce phases. The first MapReduce phase obtains rules from different splits of data applying any algorithm to discover association rules. The second MapReduce phase evaluates these rules using all the splits of the entire input dataset. The third phase (sequential) updates the nondominated solutions after being evaluated. The fourth phase (third MapReduce) builds a new dataset by the uncovered instances. The convergence property of the Maps in the MRQAR framework depends on the type of algorithm used in the first phase. Evolutionary algorithms usually provide good solutions. In particular, the pareto-based algorithms are able to converge into a high number of good solutions through the non-dominance and diversity criteria. Note that MRQAR is a general framework and the learning method of the attribute intervals of QAR depends on the type of the algorithm used in the first phase. For instance, many existing algorithms apply a domain partition technique to handle continuous variables that might give rise to information loss, however this issue is independent of the framework proposed in this work.

As a case study to validate the performance of the proposed framework, we use the recent multiobjective EA (MOEA) named MOPNAR [10] in the first phase of MRQAR due to the good behavior in the Pareto convergence sequentially. Spark's implementation of the MRQAR algorithm including MOPNAR can be downloaded from the Spark's community repository[2].

An experimental study has been conducted on five Big Data problems to asses the performance of the proposal and the quality of the rules obtained. Furthermore, the proposal has been compared with a parallel version of the well-known classical algorithm FP-Growth named PFP [11] that follows a MapReduce scheme implemented in Spark and is available in MLlib [9]. Finally, an statistical analysis has been applied to compare the performance of MRQAR and PFP.

This work is organized as follows. Section 2 introduces the basic concepts of QARs, Big Data, MapReduce programming paradigm and Apache Spark, in addition to review several existing methods to discover association rules in Big Data. Section 3 details the proposed framework to obtain QARs from Big Data problems. Section 4 describes the experimental setup, the algorithm used as case study named MOPNAR and the configuration parameters for the methods analyzed. Furthermore, that section shows the results obtained in the five Big Data problems and provides a comparative with other Big Data approach. Finally, Section 5 summarizes the conclusions drawn from the analysis conducted.

[2] https://spark-packages.org/package/djgarcia/MRQAR.

2. Preliminaries

2.1. Quantitative association rules

Association rules aim at discovering frequent set of related attributes in the dataset that are represented by understable rules. Agrawal et al. formally described the association rules for the first time in [12]. Let $I = \{i_1, i_2, ..., i_n\}$ be a set of n items or attributes, and $D = \{tr_1, tr_2, ..., tr_N\}$ a set of N records in a dataset. Each record includes a subset of items or attributes. A rule is expressed as $A \Rightarrow C$ where A is the antecedent and C is the consequent of the rule, where $A, C \subseteq I$ and $A \cap C = \emptyset$, A and C.

The association rules are named QARs in continuous domains. Let $F = \{F_1, F_2, ..., F_n\}$ be a set of n attributes, with values in $\mathbb{R}$. Let X and Y be two disjoint subsets of F, that is, $X \subset F$, $Y \subset F$, and $X \cap Y = \emptyset$. A QAR is a rule $A \Rightarrow C$ that defines a relationship between the attributes from the antecedent and the attributes from the consequent and each attribute has an interval of membership values. The antecedent A is composed of the attributes of X and the consequent C is composed of the attributes of Y. A conjunction of multiple boolean expression $F_i \in [v_1, v_2]$ (with $v_1, v_2 \in \mathbb{R}$) composes A and C.

For instance, a QAR could be numerically expressed as *Temperature* $\in [38, 42] \wedge$ *Humidity* $\in [2, 25] \Longrightarrow$ *Tropospheric Ozone* $\in [0, 140]$, where the antecedent is composed of the attributes *Temperature* and *Humidity* and the consequent has the attribute *Tropospheric Ozone*.

The QARs obtained by any algorithm can be evaluated using different quality metrics with the aim at selecting the best rules. All these measures are conceived to separately evaluate different properties of the rules such as generality or reliability [13].

Many authors have used the support and confidence as quality and optimization measures but these metrics do not cover some properties of the rules. For instance, the consequent support is not involved in the evaluation of the confidence, then this measure does not identify negatively dependent attributes. In the literature, we can find other quality measures to evaluate different features of QARs [14].

Table 1 details the formula, properties evaluated by some of the most popular quality measures and the interval values of these measures. The QARs obtained in the experimentation presented in this paper are going to be evaluated using the described measures. The number of instances of the dataset that contains the item A is represented by $|A|$. N refers to the amount of records of the dataset.

Authors in [20] classify the QAR mining into several catagories according to the type of computational method used and also discuss the advantages and disadvantages of each category. The main categories identified by these authors are as follows: Partitioning-based approaches in which the domain of the attributes is divided into disjoint intervals [21,22]; Clustering-based approaches in which the intervals are generated by meaningful and dense regions [23,24]; Statistical-based approaches that define the intervals of QAR analyzing the data distribution using statistical metrics [25]; Fuzzy-based approaches that use linguistic terminologies to define partitions to represent the associations [26,27]; Finally, EA-based approaches that have been extensively applied in the last years [28–30]. For instance, one of the most common evolutionary approaches are based on MultiObjective Evolutionary Algorithms (MOEAs) [31] where multiple conflicting objectives, are optimized simultaneously. Instead of providing only one solution as monoobjective approaches do, this kind of algorithms provides the user an optimal set of non-dominated solutions (rules), named the Pareto-optimal set, where each solution represents a rule with a different trade-off between the objectives optimized. In fact, different types of MOEAs have been proposed in the literature to deal with the discovery of QARs [32] such as the works proposed in [33,34] or the approach based on MOEA/D-DE named MOPNAR [10]. This kind of MOEAs addresses a multiobjective problem as N subproblems optimized at the same time using an EA. A MOEA to discover rare and interesting QAR was presented in [35]. Other EA-based approaches,

Table 1
Measures to assess the quality of QAR.

Metrics	Formula	Definition	Values
Supp(A)	$\|A\|/N$	Frequency of A	[0, 1]
Supp($A \Longrightarrow B$)	$\|(A \cap C)\|/N$	Proportion of instances containing the rule	[0, 1]
Conf($A \Longrightarrow C$)	$supp(A \Longrightarrow C)/supp(A)$	Reliability of the rule	[0, 1]
Lift($A \Longrightarrow C$) [15]	$supp(A \Longrightarrow C)/(supp(A) \cdot supp(C))$	Interest of the rule • Lift $<$ 1: A and C are negatively dependent • Lift $=$ 1: A and C are independent • Lift $>$ 1: A and C are positively dependent	[0, + ∞)
Conviction($A \Longrightarrow C$) [16]	$(1 - supp(C))/(1 - conf(A \Longrightarrow C))$	Dependence between A and C • Conv $<$ 1: A and C are negatively dependent • Conv $=$ 1: A and C are independent • Conv $>$ 1: A and C are positively dependent	(0, + ∞)
Certainty Factor($A \Longrightarrow C$) [17]	• If $conf(A \Longrightarrow C) > sup(C)$: $(conf(A \Longrightarrow C) - supp(C))/(1 - supp(C))$ • If $conf(A \Longrightarrow C) \leq supp(C)$: $(conf(A \Longrightarrow C) - supp(A))/supp(C)$	Gain normalized • CF $<$ 0: A and C are negatively dependent • CF $=$ 0: A and C are independent • CF $>$ 0: A and C are positively dependent	[–1, 1]
NetConf($A \Longrightarrow C$) [18]	$(supp(A \Longrightarrow C) - supp(A)supp(C))/(supp(A)(1 - supp(C))$	Interest of the rule • NetConf $<$ 0: A and C are negatively dependent • NetConf $=$ 0: A and C are independent • NetConf $>$ 0: A and C are positively dependent	[–1, 1]
YulesQ($A \Longrightarrow C$) [19]	$\frac{(supp(A \Longrightarrow C) \cdot (supp(\neg A \Longrightarrow \neg C)) - (supp(\neg A \Longrightarrow C) \cdot (supp(A \Longrightarrow \neg C))}{(supp(A \Longrightarrow C) \cdot (supp(\neg A \Longrightarrow \neg C)) + (supp(\neg A \Longrightarrow C) \cdot (supp(A \Longrightarrow \neg C))}$	Odds ratio • YulesQ $<$ 0: A and C are negatively dependent • YulesQ $=$ 0: A and C are independent • YulesQ $>$ 0: A and C are positively dependent	[–1, 1]

such as niching genetic algorithms, have been applied to discover QAR [36]. Alternatively, other authors have not based their proposals following general approaches as the aforementioned categories do. For instance, the work presented in [37] defines syntactic constrains to represent QAR by a metarule-guided approach.

2.2. Big Data, MapReduce programming model and Apache Spark

The process of knowledge extraction has become a big challenge to data mining and machine learning algorithms due to the scalability issues caused by the vast volume of data generated nowadays. Hence, such algorithms need to be redesigned in order to be applied into real-world problems. Thus, the emerging concept of Big Data comprises the complex and large amounts of data that the classical techniques can not process or analyzed [5].

MapReduce is a well-known programming model designed by Dean and Ghemawat [3,4,38] to address Big Data problems. This computational paradigm was designed for easily writing reliable and fault-tolerant applications capable of handling vast amounts of data in-parallel on large clusters. Generally, the MapReduce framework operates on ⟨key, value⟩ pairs working in two phases: *Map* and *Reduce*. In the *Map* phase, the input data is processed generating intermediate results as the input of the *Reduce* phase to produce the final output. Both *Map* and *Reduce* phases process data structured in terms of ⟨key, value⟩ pairs.

Apache Hadoop is one of the most popular implementations of MapReduce. This is an Open-source project overseen by the Apache Software Foundation based on Java and created by Doug Cutting team [39,40]. Hadoop has a distributed storage system named *HDFS* (*Hadoop Distributed File Systems*).

Recently, other Big Data projects have emerged to deal with large-scale datasets. For instance, the fast and general engine for large-scale dataset processing named Spark [7,8] is a new alternative to Hadoop. Spark overcomes the main issues of Hadoop [5] such as the intensive disk usage, poor performance on iterative computing, insufficiency for in-memory computation, low inter-communication capacity. Spark uses a distributed data structure called Resilient Distributed Datasets (RDDs) [41]. An RDD is a read-only and a fault-tolerant partitioned collection of records that can be operated on in parallel. An RDD is able to load a dataset in memory and read it multiple time in whereas Hadoop has to load the dataset in each iteration. Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. RDDs provide both transformation and action operations. Transformations are not evaluated when are defined and return a new RDD from an existing one. Actions evaluate all the previous transformations and calculate a new value.

Other remarkable Spark-related project at Apache is the machine learning library named MLlib [9,42] which is composed by several learning algorithms and statistic tasks such as classification, regression, clustering, collaborative filtering, optimization and dimensionality reduction. Recently, this library has been divided into two different packages, MLlib and ML, depending if they are built on top of RDDs or DataFrames, respectively. Other projects developed on the top of the Spark core are SparkSQL that provides SQL language support in the Spark programs, Spark Streaming to analyze data streams, Spark GraphX to process graphs in Spark.

2.3. Big data in association rule mining

In the last years, many researches have focused on how to deal with the problem of association rule discovery in parallel and distributed environments but they are only focused on adaptations of classical algorithms that deal with nominal or boolean data, although the real-world applications have quantitative data. Recently, the authors in [43] provide a summary of the issues presented in Big Data analytics such as information loss, high computational cost and useless rule generation. Different methods to improve the speed up of the pattern mining algorithms ranging from traditional methods and new trends in Big Data such as MapReduce are analyzed in [44].

Authors in [45] summarize several improved techniques based on the parallelization of the Apriori algorithm using Hadoop MapReduce framework. This survey studies several techniques according to the goal of the work, datasets and platform used. For instance, an improved version reducing the number of scans of the dataset is proposed in [46]. An approach based on vertical dataset partitioning using a MapReduce model is described in [47]. This algorithm performs better when the number of instances increases but presents bad behavior when the number of instances is low. Both Dist-Eclat and BigFIM algorithms based on the MapReduce framework are presented in [48]. In particular, Dist-Eclat algorithm is an optimized implementation of the classical Eclat algorithm [49] focused on the speed but only works if the

data can be fitted into memory. BigFIM algorithm is an hybrid optimization of Apriori [50] and Eclat [49] to be executed on truly massive amounts of data. Note that the aforementioned algorithms have been tested using datasets with a maximum number of instances below of 6.5 millions. Other authors have retrieved frequent itemsets from large datasets instead of mining association rules such as the Input Split Frequent Pattern Tree algorithm proposed in [51]. Lately, an efficient implementation to mine maximally informative k-itemsets based on joint entropy and following a MapReduce design is proposed in [52]. The SILVERBACK algorithm is presented in [53] that discover frequent itemsets and association rules in activity logs of users in social datasets. This algoritm includes probabilistic columnar infrastructure, bloom filters, sampling techniques and pruning techniques based on the principles of Apriori algorithm. Other examples based on classical algorithms such as Apriori, FP-Growth or combination of them can be found in [54,55].

Other authors have addressed the discovery of association rules in Big Data following a Spark-based implementation. For instance, an adaptation of the basic Apriori algorithm is proposed in [56]. Other authors have proposed an efficient distributed algorithm named DFIMA implemented using Spark to discover frequent itemsets in [57]. This algorithm follows the Apriori concept reducing the number of candidates by the use of a matrix-based pruning method. Recently, the algorithm named HFIM has been proposed in [58] and also is implemented using Spark. This work try to solve the limitations of the Apriori algorithm in distributed environment following the concept of vertical dataset. Finally, a parallel version of the well-known FP-Growth algorithm [11] has been implemented in Spark available in MLlib [9]. It can be noted that the aforementioned proposals are focused on binary or discrete data although the domain of most real-world applications are continuous.

After reviewing the existing literature, it can be drawn that most of the existing techniques designed to address the discovery of association rules in Big Data scenarios are based in classical algorithms such as Apriori, FP-Growth or combination of them. Thus, these algorithms only deal with binary or discrete values. Then, a data discretization procedure is required before applying them in quantitative domains to obtain rules. Finally, we would like to remark that we propose a framework that allows running any sequential QAR algorithm in Big Data problems without designing a particular adaptation of a specific algorithm, whereas most existing algorithms are adaptations of a specific sequential association rule algorithm following the MapReduce paradigm.

3. MRQAR: a multiobjective MapReduce design to mine QARs in Big Data

This section describes the proposed framework MRQAR, a MapReduce implementation that uses Apache Spark to discover efficiently QARs in Big Data problems using other algorithm as rule extraction model in each Map. As stated before, the rule-machine process is the most costly phase in EAs, then, this new approach is devoted to reduce the run-time costs focused on the number of instances of the training dataset and memory consumptions without quality loss in the results.

3.1. Scalable approach to mine QARs for Big Data: MRQAR

Current association rule algorithms need to be redesigned to handle Big Data problems to perform an efficient evaluation of the objectives and keep the quality of the rules obtained simultaneously. To accomplish that, we propose an incremental learning scheme that follows a MapReduce design to discover QARs from different proportions of the data.

As stated in Section 2.2, the MapReduce paradigm splits the training dataset into a number of subsets of instances. The challenge is how to discover quality association rules that represent the complete dataset through subset of rules obtained in different splits of the dataset.

To fulfil this goal, the association rule algorithm is only executed in a subset (Map), thus, the fitness function evaluates only a subset of instances. The evolutionary process for each Map ends when a number of evaluations is reached (N_{eval}) and a set of QARs is returned (*RuleSet*).

Once all Maps are processed by the rule mining algorithm, the *RuleSet* from each Map is collected. Then, the global quality of the QARs of each *RuleSet* are evaluated using the entire dataset. To accomplish that, antecedent support, consequent support and rule support are partially calculated for each Map, and then they are aggregated to obtain the global evaluation that allow us to calculate the quality measures of the complete dataset. After that, all the QARs of each *RuleSet* are used to update an external set of rules henceforth named *GlobalRulePool*, that store the nondominated solutions found for the entire dataset considering the quality measures previously calculated. These quality measures will be the objective functions used by the sequential association rule algorithm. Subsequently, the redundant QARs of the *GlobalRulePool* are removed.

Thereafter, the instances of the training dataset covered by the QARs belonging to the *GlobalRulePool* are marked. A new dataset is built only by the uncovered instances of the original dataset. This new dataset is again splitted into subsets of instances and the association rule algorithm is again executed for each of them. If the number of the remaining uncovered instances is less than a minimum threshold (*trNotCover*), the entire dataset is again used to process the new iteration and build the new input splits.

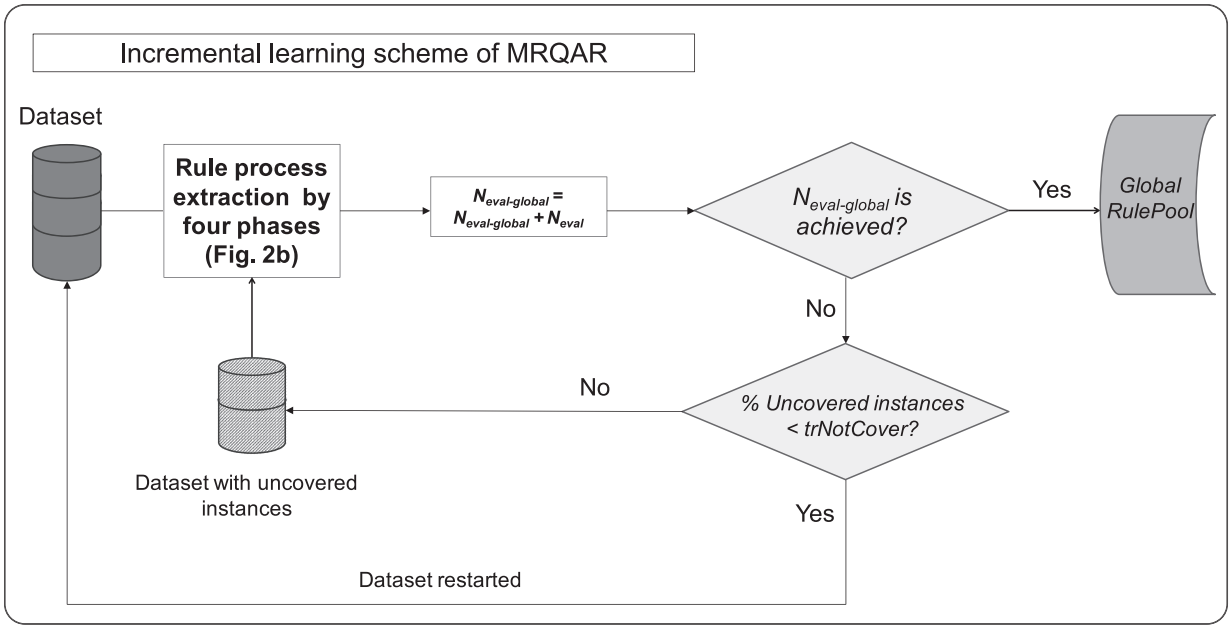
The aforementioned process is repeated until a global number of evaluations is reached ($N_{eval-global}$). The global number of evaluations is updated when all the subsets are processed for each iteration taking into account the number of trials of the slowest sub-problem, that is, the number of evaluations of the input split that requires more evaluations. Furthermore, the number of trials spent to evaluate the QARs belonging to each *RuleSet* in the entire dataset is also added.

An overview of the aforementioned incremental and parallel learning scheme is depicted in the Fig. 1. A specific example with a toy dataset to show the general process of our proposal step by step can be observed in Fig. 2.

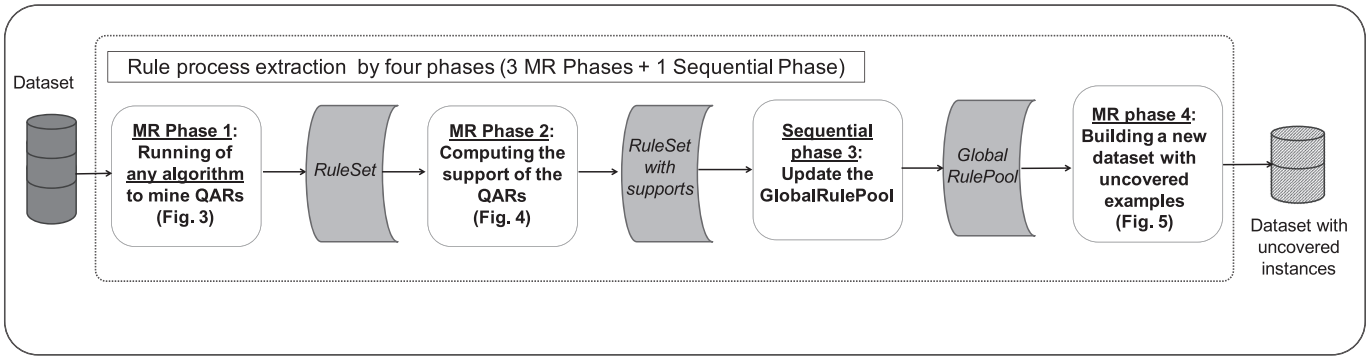
3.2. The MRQAR framework for Big Data: a MapReduce design

This section describes the MapReduce design details of the generic scalable approach proposed in this work. Furthermore, the Spark primitives used for the implementation of the framework are also detailed (Section 3.2.1). The learning scheme of MRQAR is composed by four phases where three of them follow a MapReduce scheme:

- The first phase, that follows a MapReduce design, is devoted to run an algorithm to obtain a set of QARs (*RuleSet*) for each subset in which the input dataset is divided (Section 3.2.2).
- The second phase, that also follows a MapReduce scheme, performs the support computation of the QARs obtained in the previous phase considering all the instances of the dataset (Section 3.2.3). This phase aims at evaluating the quality of the rules over the entire dataset because it is necessary in the next phase. Note each *RuleSet* of the first phase has been only evaluated using the instances of the subset in which they have been trained. Therefore, in the second phase is necessary to calculate the quality of the rules over the entire dataset with the aim at selecting the best rules that present the best performance in the original dataset with all the instances and not only in the subset trained.
- The third phase sequentially updates a global rule set denominated as *GlobalRulePool* that stores the non dominated solutions found in the entire dataset. To accomplish that, the same measures used by the sequential algorithm to evaluate the quality of the QARs are calculated by the support obtained in the second phase



(a) Incremental learning scheme of MRQAR



(b) Rule process extraction in each iteration

Fig. 1. General scheme of MRQAR.

(Section 3.2.4).

- The fourth phase builds a new dataset composed of the uncovered instances by the rules of *GlobalRulePool* following a *MapReduce* scheme (Section 3.2.5).

These four phases are part of an iterative process that is repeated while a minimum threshold of global evaluations ($N_{eval-global}$) is not reached. Fig. 1 depicts the general overview proposed method in which the four phases performed in each iteration of the learning process can be observed. Fig. 2 displays a running example to illustrate the steps of each phase of MRQAR.

It is well-known that there are two issues to consider in Big Data: redundancy and the consequences of data partitioning (inherent in a MapReduce model). The first reward the second in many cases as authors discussed in [59]. Data redundancy implies that the partitions have enough instances to have a good representation of data in the maps. However, it is true that there may be a lack of representation of data on a map and therefore the following may occur:

- Some rules could not be discovered due to the breaking of concrete niches with quality data that are broken between several maps. It is the risk to assume in a MapReduce model. It is important to remark that the framework of the MRQAR is designed to obtain rules in datasets that are large enough that it is not possible to obtain rules with a sequential algorithm due to their size or the high runtime required.
- It can be discovered rules that seemed interesting in a map but they could be not so important for the complete data set. It is necessary to have a phase to detect which rules are not important and then not to select them. In the particular case of MRQAR, MR Phase 2 evaluates the rules in the whole dataset and then, the Sequential Phase 3 selects the rules that have a good quality for the complete dataset.

The following sections describe the Spark primitives, in addition to the design of the three MapReduce phases.

3.2.1. Spark primitives

Some basic Spark primitives from Spark API have been used to implement the framework. These primitives offer much complex operations by extending the MapReduce paradigm. The most relevant to the algorithm are outlined as follows[3]:

[3] For a complete description of Sparks operations, please refer to Sparks API: http://spark.apache.org/docs/latest/api/scala/index.html.

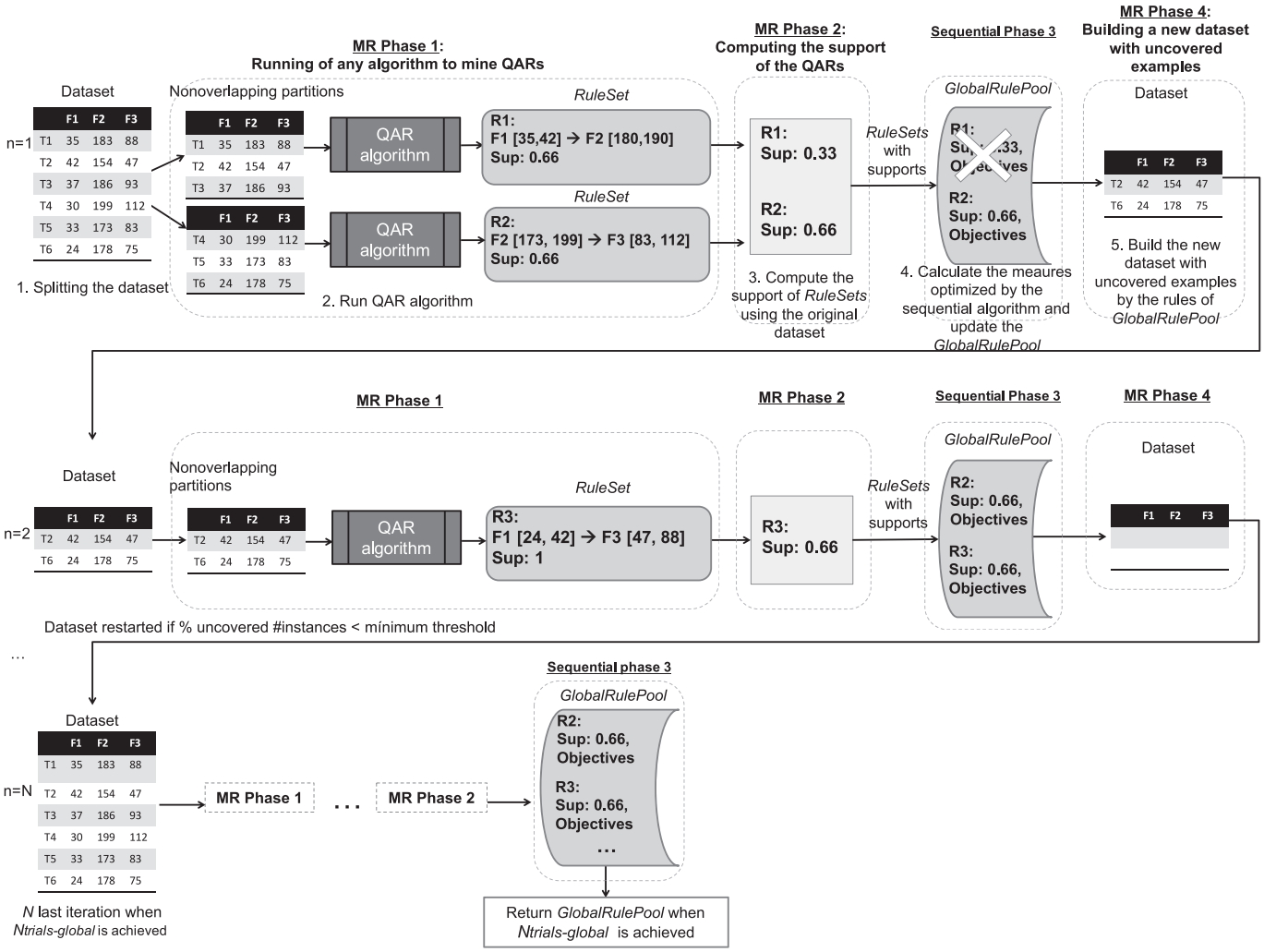


Fig. 2. Example of the rule process extraction of MRQAR. Note that the objectives depend of the measures optimized by the sequential algorithm used in the first phase of MRQAR.

- *mapPartitions*: applies a transformation to each partition of a RDD. Once the operation has been performed to each partition, the resulting RDD is returned.
- *reduce*: reduces the elements of a RDD using the specified commutative and associative binary operator.
- *filter*: return a new RDD containing only the elements that satisfy a predicate.

3.2.2. MR Phase 1 - Running of an algorithm to mine QARs

A set of QARs (*RuleSet*) is obtained in this phase. This is a MapReduce process where each *Map* task is devoted to execute an association rule algorithm using the available data in the corresponding partition (MR Phase 1 in the Fig. 1(b) as is illustrated in Fig. 3. It is remarkable to specify that this section does not provide details about the discovery process of the rules in each map or partition since it depends on the association rule algorithm used. This MapReduce process consists of three phases: **initial**, ***map*** y **final**.

The **initial** stage splits the input dataset into separate blocks that are distributed by the processing nodes. Then, the ***map*** stage is applied in which the algorithm of association rules is executed using the data block of the partition of each *Map* process and a *RuleSet* is generated for each one. Each execution ends when a given number of evaluations is reached ($N_{eval-partial}$). Each *RuleSet* has high quality measure values for the *Map* partition in which was obtained. Due to the random split of the dataset, the **final** stage combines the generated *RuleSets* since these high quality values can be keep for other instances of the dataset. This MapReduce process has not *Reduce* phase since the outputs of each *Map* are directly combined. The aggregation of the results obtained by the *Map* processes could also be accomplished by a *Reduce* phase but being unnecessary would cause a higher runtime consumption.

The pseudocode of the **map** stage of the MR Phase 1 for running an algorithm to mine QARs is presented in Algorithm 1. This process obtains a list of QARs by means the algorithm executed using the instances of each *Map* partition only.

3.2.3. ***MR Phase 2*** - Computing the support of the QARs

This phase execute a new MapReduce process (MR Phase 2 in the Fig. 1(b) that computes the rule support, the antecedent support and the consequent support for each QARs of the obtained *RuleSet* in the previous phase. The computation of these three support measures allows the calculation of all the metrics to evaluate the quality of the rules. This evaluation phase is performed to quantify the global quality of the obtained rules using the entire dataset since MR Phase 1 only computes the quality of the rules using the instances of the training map partition in which each set of rules has been obtained. This phase consists of four stages: **initial**, ***map, combiner, reduce*** y **final** as can be observed in Fig. 4.

The **initial** stage obtains all the QARs reported in the previous phase. In the ***map*** stage, each *Map* process computes the support of the QARs for each instance of its partition. In the ***reduce*** stage, each *Reduce*

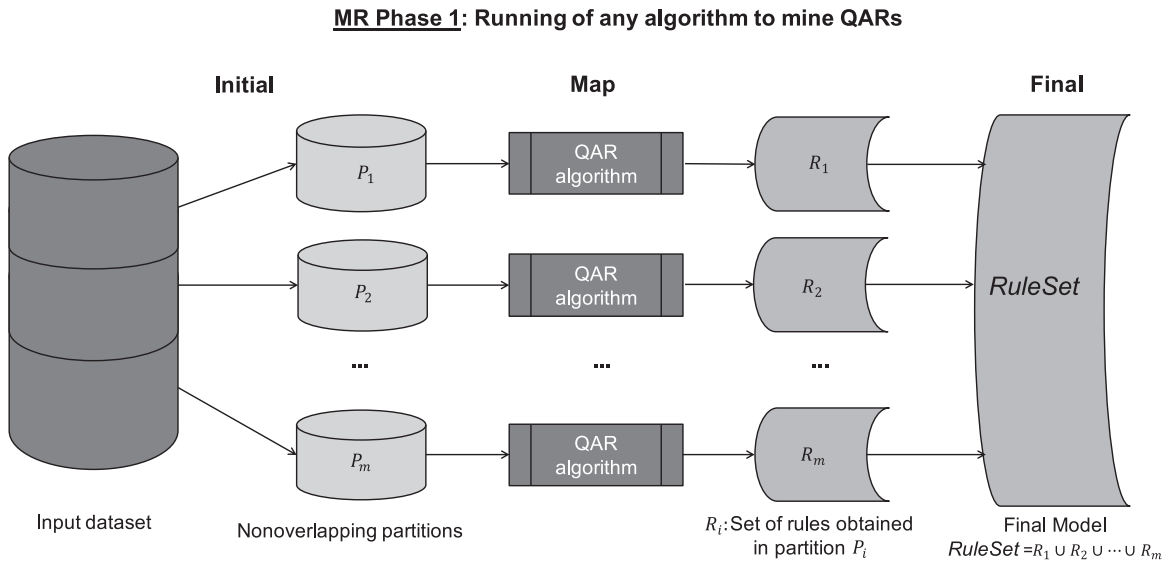


Fig. 3. Flowchart of how the running of an algorithm to mine QARs is executed over the nonoverlapping subsets.

process combines all the obtained supports for each rule to generate the global support of each one in the original dataset. The **final** stage returns all the QARs and the global support obtained. To reduce the amount of information that is received in the ***reduce*** stage, a local *reduce* is executed at the end of each ***map*** stage considering only the ⟨key, value⟩ pairs processed in the same split. This local *reduce* is known as ***combiner*** stage, which computes the global support of the QARs for all the instances of its partition.

The pseudocodes of the **map** and **reduce** stages of the MR Phase 2 to compute the support of QARs are provided in Algorithms 2 and 3. Algorithm 2 computes the support of rules obtained in the instance represented by the pair ⟨key, value⟩ (Line 7: Algorithm 2). Note that *supA, supC, supR* stand for support of the antecedent, consequent and rule, respectively. When each *Map* has finished, Algorithm 3 is called to accumulate the support of the antecedent (Line 4: Algorithm 3), support of the consequent (Line 5: Algorithm 3) and support of the rule (Line 6: Algorithm 3) of each rule represented by each instance and compute the total support. Finally, the association rule list (*RuleSet*) is returned as final output when Algorithm 3 finishes.

3.2.4. ***Sequential phase 3*** - *Update the GlobalRulePool*

This phase updates the nondominated solutions that compose the *GlobalRulePool* for all the instances of the entire dataset. Algorithm 4 provides the pseudocode of the third phase.

Each rule of *RuleSet* is checked if satisfies any minimum threshold in the case of algorithms that requires minimum thresholds such as minimum support or minimum confidence (Line 4: Algorithm 4).

This phase checks sequentially if each rule of *RuleSet* is better or worse than each one that belongs to the *GlobalRulePool* according to the concept of non-dominance. The non-dominance between two rules is applied taking into account the objectives to optimize. In the study case used in this work, we have used the same objectives optimized in MR Phase 1. Lines 6 to 11 in Algorithm 4) shows when a new rule is added or not to *GlobalRulePool* according to if a rule is dominated by the rules from *GlobalRulePool* or dominates other rules from *GlobalRulePool*.

Finally, *GlobalRulePool* is updated removing redundant rules (Line 16: Algorithm 4).

3.2.5. ***MR Phase 4*** - *Building a new dataset with the uncovered instances*

This phase executes a new MapReduce process (MR Phase 3 in the Fig. 1(b) and builds a new dataset through the uncovered instances by the QARs contained in *GlobalRulePool*. This phase consists of three stages: **initial**, ***map*** and **final** as can be observed in Fig. 5. The **initial** phase obtains independent blocks after splitting the input dataset that are distributed through the processing nodes. In the ***map*** stage, each *Map* process analyzes if each instance of its partition is covered by some QAR of *GlobalRulePool*. The instances uncovered by any QAR of *GlobalRulePool* are returned. The **final** stage builds a new *RDD* that contains the aggregation of the instances uncovered generated for each *Map* process. If the number of instances of the new dataset does not satisfied a minimum covered threshold (*trNotCover*), all the instances of the original dataset are again used. This new dataset will be used as input data in the phase described in Section 3.2.2. This MapReduce phase does not have a *Reduce* phase since the outputs of each *Map*, uncovered instances, are directly combined.

The *map* stage of the MR Phase 3 that builds a new dataset with the uncovered instances by the rules obtained in *GlobalRulePool* is described in Algorithm 5. If any rule of *GlobalRulePool* satisfies one instance, this instance does not belong to the new dataset (Line 6–7: Algorithm 5). The outputs of each Map are directly combined, therefore, the MR Phase 3 does not have a *Reduce* phase.

1: **Input:** *data* that represents the dataset, each row is an Array of values.
2: **Output:** The rules discovered by the QAR algorithm
3: $RuleSet \leftarrow$
4: **map partitions** $instances \in data$
5: {*instances* will contain all instances in each partition}
6: $associationRules \leftarrow algorithm.run(instances)$
7: **end map**

Algorithm 1. Function of the MR Phase 1 of the MRQAR framework that retrieves all the instances of the Map split and runs the association rule algorithm in the instances of the Map partition:

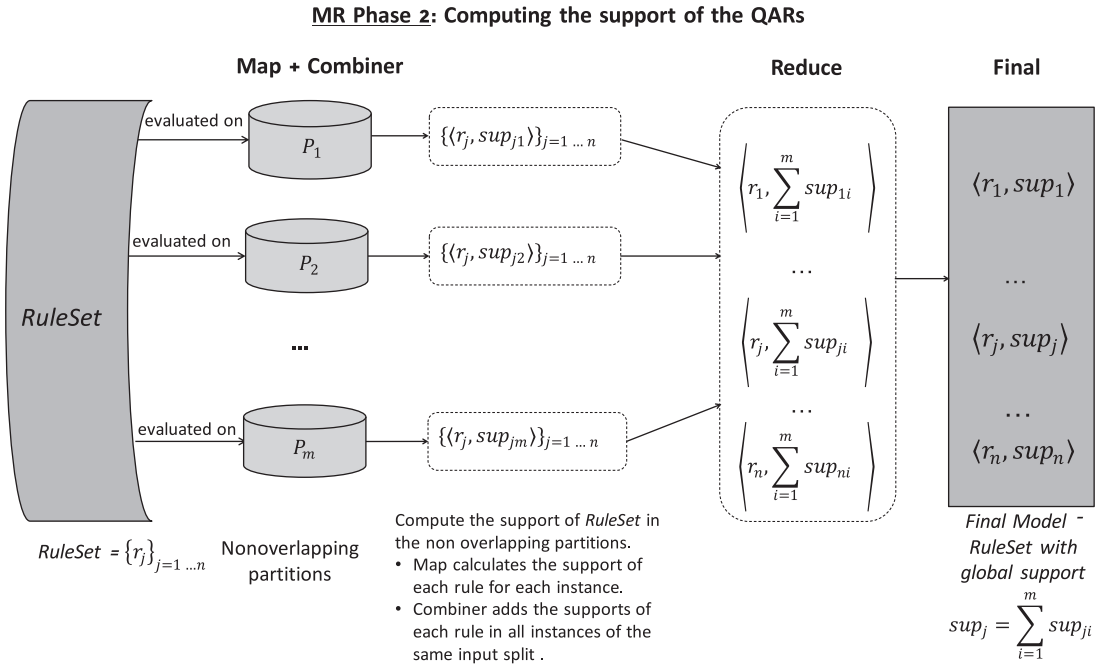


Fig. 4. Flowchart of how the support of QARs is computed.

4. Experimental study

4.1. Experimental set-up

The main features of the big datasets used in the experimental study are presented in this section. The analyzed algorithms and the configuration parameters used are also introduced. Finally, a description of the hardware and software used in this study are provided.

4.1.1. Datasets

The following big datasets have been analyzed to validate the performance of the proposed framework MRQAR:

- Susy dataset has been taken from UCI Machine Learning Repository [60], where it was deposited by the authors of the work proposed in [61]. Bache et al. specified that this dataset has been produced using Monte Carlo simulations. It consists of 5,000,000 instances and 18 attributes. The first 8 features are kinematic properties measured by the particle detectors in the accelerator. The last ten features are functions of the first 8 features. The task is to distinguish between a process where new supersymmetric particles are produced and a background process with the same detectable particles but fewer invisible particles and distinct kinematic features.
- Higgs dataset has been taken from UCI Machine Learning Repository [60], where it was deposited by the authors of the work proposed in [61]. Bache et al. specified that this dataset has been produced using Monte Carlo simulations. It has 11,000,000 instances and 28 attributes. The first 21 features are kinematic properties measured by the particle detectors in the accelerator. The last seven features are functions of the first 21 features. The task is to distinguish between a signal process where new theoretical Higgs bosons are produced and a background process with the identical decay products but distinct kinematic features.
- Epsilon dataset has been taken from LIBSVM [62]. This dataset was artificially created for the Pascal Large Scale Learning [63] in 2008.
- ECBDL14 dataset [64,65]. This dataset was used as a reference at the ML competition of the Evolutionary Computation for Big Data and Big Learning held on July 14, 2014, under the international conference GECCO-2014. It consists of 631 characteristics (including both numerical and categorical attributes) and 32 million instances. It is a binary classification problem where the class distribution is highly imbalanced: 2% of positive instances. For this work, we have used the complete dataset (ECBDL14 complete) and furthermore, we have randomly selected a subset of 12 million instances and 90 characteristics (ECBDL14 reduced).

The main features of the three big data problems are briefly described in Table 2. In particular, the first column indicates the name of

1: **Input:** *RuleSet* a set of rules discovered by the association rule algorithm
2: **Output:** (key, value) pair, where key indicates ruleId of each QAR of RuleSet evaluated in the instance corresponding to the offset key and value contains the support of the antecedent (*supA*), support of the consequent (*supC*) and support of the rule (*supR*) of each rule in such instance.
3: $supports \leftarrow$
4: **map partitions** $instances \in data$
5: {*instances* will contain all instances in each partition}
6: **for each** $Rule \in RuleSet$ **do**
7: $\{ruleId, \{supA, supC, supR\}\} \leftarrow computeSupports(instances, Rule)$
8: **end for**
9: **end map**

Algorithm 2. Function of the MR Phase 2 of the MRQAR framework that computes the support of each association rule:

1: **Input:** *supports* a (key, value) pair, where key is the id of a rule and value is the support of the antecedent (*supA*), support of the consequent (*supC*) and support of the rule (*supR*) of such rule for each instance.
2: **Output:** (key, value) pair, where key is the id of each rule of *RuleSet* and value contains the global support of the antecedent, consequent and the rule considering all the instances.
3: $Rules \leftarrow supports.reduce\{(a, b) =>$
4: $\quad a.supA+ = b.supA$
5: $\quad a.supC+ = b.supC$
6: $\quad a.supR+ = b.supR$
7: }

Algorithm 3. *Reduce* phase for the MRQAR framework for computing the support of the model:

the problems, the second column represents the number of attributes of each attribute type (Real/Integer/Nominal) in the problems, the column "Examples" shows the number of instances of each problem and the column "Size" is the size of memory expressed in GB. Note that the size of the datasets used is significant enough to undertake the study of our proposal since most of the existing non-parallel algorithms are unable to deal with these datasets or they have difficulties to obtain good results.

4.1.2. Case study: MOPNAR algorithm

MOPNAR [10] is a recent MOEA that mines a reduced set of QARs with low computational cost extending the MOEA/D-DE algorithm [66]. This kind of MOEAs decomposes the multiobjective optimization problem into N scalar optimization subproblems and uses an EA process to optimize these subproblems simultaneously. As stated in previous sections, the MR Phase 1 execute an algorithm to obtain association rules in each map partition of the dataset used as training data. In the experimentation performed, MOPNAR is applied as many times as the number of partitions of the training data. Then, a different set of QARs is obtained for each map partition as if they were independent datasets. The main features of MOPNAR algorithm are detailed as follows:

- MOPNAR obtains interesting and comprehensible rules with a good trade-off between the number of rules, support, and coverage of the dataset. To accomplish that, this proposal maximizes the following three objectives: comprehensibility, interestingness, and performance.
- MOPNAR performs an evolutionary algorithm to learns the most suitable intervals of the attributes belonging to the rules in a self-adaptive way without performing a discretization process of the variables domain of the problem so that the information loss is avoided.
- A rule is coded as a array of genes that represent the attributes and their intervals. Each gene represents an attribute that is composed of 4 parts as follows: the membership of the attribute in the rule, interval positive or negative and the interval bounds of the attributes.
- MOPNAR also introduces two new components into the evolutionary model: an external population (EP) and a method to restart the population. On one hand, the EP keeps the nondominated solutions discovered which are updated with the new rules generated by the genetic operators after deleting the redundant rules. The number of solutions of the EP is not bounded, which allows us to obtain a larger number of rules of the Pareto front regardless of the size of the population and reduce the size of the population, following a dataset independent approach. On the other hand, the population is restarted when the number of new solutions obtained in each generation does not satisfy a minimum threshold. This process introduces diversity in the population and prevents the algorithm from being trapped in local optimums. MOPNAR ends when a maximum number of evaluations is satisfied.

A detailed explanation of the MOEA/D-DE scheme and MOPNAR algorithm can be found in [10,66], respectively.

4.1.3. Algorithms and parameters considered for comparison

As stated Section 1, the performance of the proposed framework is validated using the MOPNAR algorithm (see Section 4.1.2) as a case study in the first phase of MRQAR. Henceforth, we denote as MRQAR-MOPNAR to the proposed framework MRQAR using MOPNAR in the first phase. Then, several experiments have been carried out to compare

1: **Input:** (key, value) pair, where key is the id of each rule of *RuleSet* and value contains the global support of the antecedent, consequent and the rule considering all the instances.
2: **Output:** *GlobalRulePool* the external set of rules that contains the non-dominated solutions found for the entire dataset.
3: **for** $RuleRS \in RuleSet$ **do**
4: $\quad$ **if** *RuleRS* satisfies *Minimum Thresholds* **then**
5: $\quad\quad$ **for** $RuleGP \in GlobalRulePool$ **do**
6: $\quad\quad\quad$ **if** *ruleRS* dominates *ruleGB* **then**
7: $\quad\quad\quad\quad$ remove *ruleGB* from *GlobalRulePool*
8: $\quad\quad\quad\quad$ add *ruleRS* to *GlobalRulePool*
9: $\quad\quad\quad$ **end if**
10: $\quad\quad\quad$ **if** *ruleRS* and *ruleGB* are non-dominated solutions **then**
11: $\quad\quad\quad\quad$ add *ruleRS* to *GlobalRulePool*
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end for**
14: $\quad$ **end if**
15: **end for**
16: $GlobalRulePool \leftarrow removeRedundant(GlobalRulePool)$

Algorithm 4. Function of the Sequential Phase 3 of the MRQAR framework that update the *GlobalRulePool*:

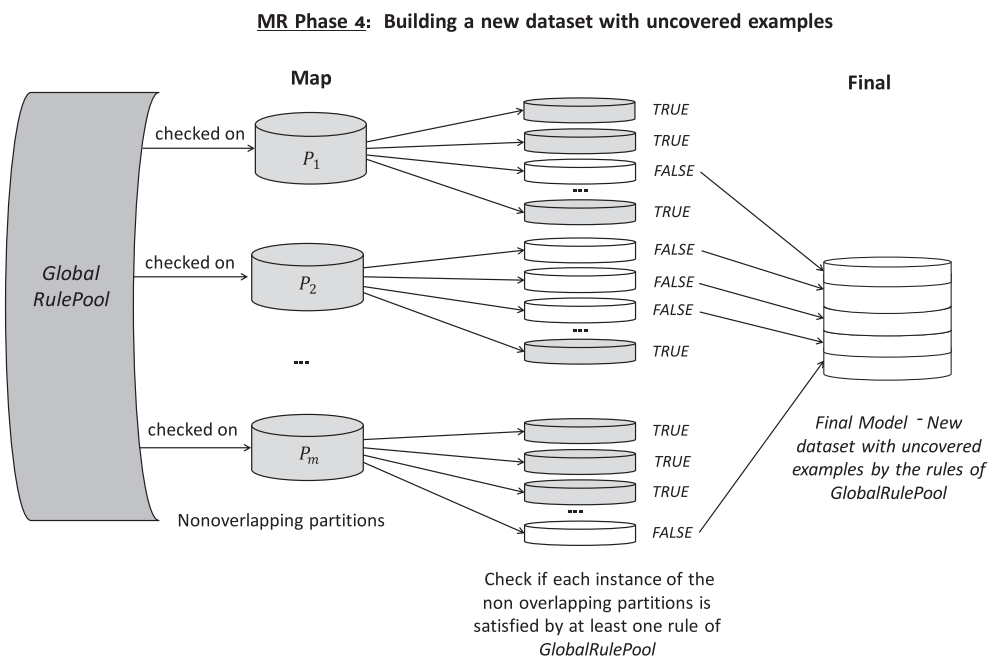


Fig. 5. Flowchart of how the new dataset is composed by the uncovered instances by *GlobalRulePool*.

MOPNAR with respect to MRQAR-MOPNAR to analyze the behavior and the inability of a sequential association rule algorithms to handle big data problems when the proposed framework is not used.

The scalability analysis of MRQAR-MOPNAR was performed by setting the amount of maps depending on the size of the problem. The number of instances has been fixed for each map taking into account previous works in which the methods for mining association rules converge properly.

Notice that the computational power of the available hardware will be used depending on the problem size. In a standard distributed model, the number of map is determined by the number of nodes of the machine. Nevertheless, the MapReduce model is hardware independent.

On the other hand, we have also compared the results obtained by MRQAR-MOPNAR with the PFP algorithm [11], which is implemented in Spark [41], available in MLlib [9].

Table 3 provides the parameters used in the comparative study of the algorithms. We have select the standard common parameters that provide a good performance avoiding very specific values for the framework proposed. Moreover, the selection of the parameters for the other algorithms has been carried out taking into account the considerations of their authors.

Table 2
Big Data problems used for the experiments.

Names	*#Atributes (R/I/N)*	*Examples*	*Size(GB)*
susy	(18/0/1)	5,000,000	2
higgs	(28/0/1)	11,000,000	6
epsilon	(2000/0/0)	500,000	11
ECBDL14 (reduced)	(18/72/0)	12,000,000	5.25
ECBDL14 (complete)	(77/462/92)	34,890,838	62

Table 3
Parameters considered for the comparison.

Algorithms	*Parameters*
MOPNAR	$N_{eval}=100000$, $H=13$, $m=3$, PopSize $=N^{m-1}_{H+m-1}$, T $=10$, $\delta=0.9$, $\eta_r=2$, $\gamma=2$, $P_{mut}=0.1$, $\alpha=5\%$
PFP	*minSup* $=0.1$, minConf $=$ *0.5, 0.8*
MRQAR-MOPNAR	$N_{eval-Global}=100000$, $N_{eval}=25000$, $H=9$, trNotCover $=2\%$, $m=3$, PopSize $=N^{m-1}_{H+m-1}$, T $=7$, $\delta=0.9$, $\eta_r=2$, $\gamma=2$, $P_{mut}=0.1$, $\alpha=5\%$

4.1.4. Hardware and software used

The experiments have been executed in the research group's cluster composed of 16 nodes. Each one contains two Intel Xeon E5-2620 microprocessors with 2.00 GHz and 15 B cache. The nodes are connected via a 40 Gb/s Infiniband network. All nodes have 64 GB of main

```
1:  Input: data that represents the dataset, each row is an Array of values.
2:  Output: newData a new dataset with all uncovered instances.
3:  newData ← data.filter{instance =>
4:  keep ← TRUE
5:  for Rule ∈ GlobalRulePool do
6:      if instance is satisfied by Rule then
7:          keep ← FALSE
8:      end if
9:  end for
10: keep
11: }
```

Algorithm 5. Function of the MR Phase 4 of the MRQAR framework that builds a new dataset with the uncovered instances:

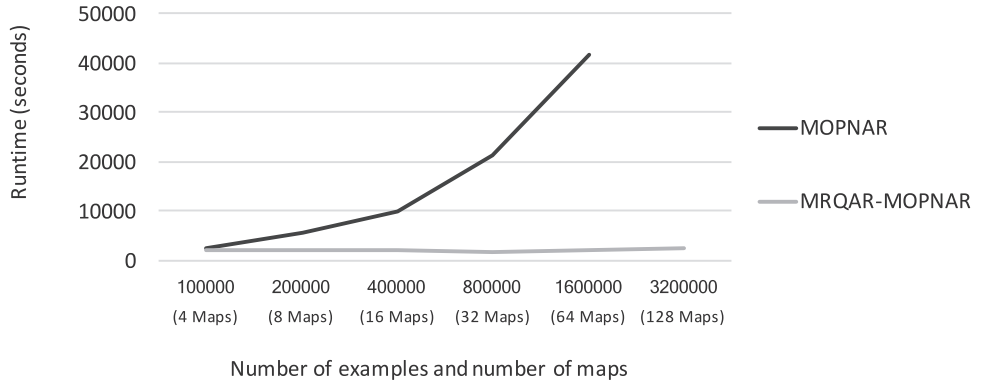


Fig. 6. Runtime expended for MOPNAR and MRQAR-MOPNAR according to number of instances executed.

Table 4
Results of the measures for MOPNAR and MRQAR-MOPNAR over the higgs subsets.

Algorithm	*#R*	*Support*	*Confidence*	*Lift*	*Conviction*	*CF*	*Netconf*	*YulesQ*	*Attributes*	*%Trans*
	higgs (100000 examples)									
MOPNAR	62.60	0.39	0.84	**3.89**	∞	0.74	0.47	0.92	2.93	99.31
MRQAR-MOPNAR	**66**	**0.52**	**0.94**	2.08	∞	**0.9**	**0.52**	**0.96**	**1.91**	**99.84**
	higgs (200000 examples)									
MOPNAR	**159.20**	0.30	0.82	**3.21**	∞	0.75	0.52	0.91	2.71	97.49
MRQAR-MOPNAR	118	**0.4**	**0.88**	2.53	86.36	**0.8**	**0.6**	**0.95**	**1.75**	**99.86**
	higgs (400000 examples)									
MOPNAR	**211.20**	0.41	0.87	**4.79**	∞	0.80	0.50	0.94	2.64	96.12
MRQAR-MOPNAR	107	**0.57**	**0.93**	2.4	76.36	**0.87**	**0.59**	**0.97**	**1.35**	**99.92**
	higgs (800000 examples)									
MOPNAR	**224.60**	0.32	0.86	**6.44**	∞	0.74	0.53	0.91	3.11	99.44
MRQAR-MOPNAR	140	**0.49**	**0.94**	3.17	91.10	**0.91**	**0.61**	**0.98**	**1.43**	**99.85**
	higgs (1600000 examples)									
MOPNAR	**575.80**	**0.54**	**0.92**	2	∞	0.78	0.33	0.89	2.75	98.21
MRQAR-MOPNAR	178	0.53	**0.96**	**3.87**	79.94	**0.93**	**0.61**	**0.99**	**1.49**	**99.91**

Table 5
Analysis of the performance for MRQAR-MOPNAR depending on the number of maps on the susy, higgs, epsilon, ECBDL14 (reduced) and ECBDL14 (complete) datasets.

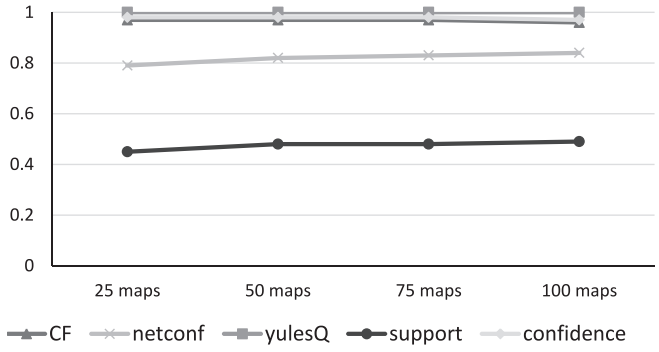
Datasets	*Maps*	*ExSplit*	*#R*	*Support*	*Confidence*	*Lift*	*Conviction*	*CF*	*Netconf*	*YulesQ*	*Attributes*	*%Trans*
susy	25	200,000	469	0.45	**0.98**	3.5	2587.3	**0.97**	0.79	**1**	1.65	**99.99**
	50	100,000	489	0.48	**0.98**	4.59	1601.8	**0.97**	0.81	**1**	**1.41**	**99.99**
	75	66,666	440	0.48	**0.98**	**7.99**	2773.8	**0.97**	0.83	**1**	1.64	**99.99**
	100	50,000	**515**	**0.49**	0.97	7.3	∞	0.96	**0.84**	**1**	1.45	**99.99**
higgs	50	220,000	401	**0.61**	0.96	2.88	**144.24**	0.93	0.57	**0.99**	**1.48**	99.95
	100	110,000	**510**	0.57	0.97	1.87	73.78	**0.95**	**0.61**	**0.99**	1.53	99.87
	150	73,333	383	0.55	**0.98**	2.44	96.19	**0.95**	**0.61**	**0.99**	1.85	**99.99**
	200	55,000	403	0.56	0.96	**2.96**	68.39	0.94	**0.61**	**0.99**	1.53	**99.99**
epsilon	10	50,000	27	0.22	0.88	2.93	198.00	0.82	0.70	0.94	5.93	93.86
	15	33,333	34	**0.25**	0.87	2.80	16.11	0.80	0.60	0.94	**4.50**	95.68
	20	25,000	31	0.23	**0.92**	4.19	75.93	**0.89**	**0.78**	**0.98**	4.68	96.75
	25	20,000	**35**	0.23	0.89	**4.53**	**419.75**	0.86	0.77	0.95	7.68	**98.05**
ECBDL14 reduced	60	200,000	132	0.34	0.93	5.96	∞	0.90	0.82	**0.99**	**1.7**	**100**
	120	100,000	179	0.33	**0.95**	**9.44**	∞	0.92	**0.84**	**0.99**	1.78	**100**
	180	66,666	173	0.35	**0.95**	6.96	∞	**0.93**	0.81	**0.99**	1.71	**100**
	240	50,000	**196**	**0.35**	0.93	4.63	∞	0.90	0.81	**0.99**	1.74	**100**
ECBDL14 (complete)	800	43,613	**19**	**0.42**	0.91	2.37	6.34	0.81	**0.70**	**0.96**	**1.20**	**99.99**
	960	36,345	16	0.36	0.89	**2.43**	6.79	0.79	0.63	0.95	1.41	99.88
	1120	31,153	18	0.40	**0.93**	2.16	**9.54**	**0.84**	0.63	0.95	1.63	**99.99**
	1280	27,258	14	0.41	0.89	2.31	6.13	0.78	0.68	**0.96**	1.40	99.95

memory except the main node that have 96 GB. The operating system used is Linux CentOS 6.9. Hadoop 2.6.0 (Cloudera CDH5.8.0) is run in the cluster. The NameNode and ResourceManager are assigned to the head node. DataNodes and NodeManager are configured in the remaining nodes. Moreover, the cluster is configured with Spark 2.1.1, where the head node is configured as master and NameNode, and the rest are workers and DataNodes.

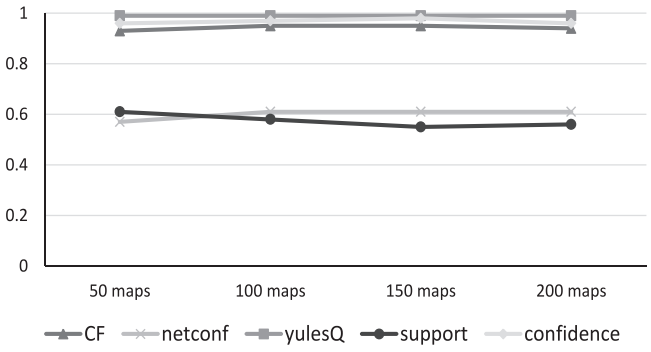
4.2. Analysis of MOPNAR with respect to MRQAR-MOPNAR

In this subsection, we have been carried out several experiments to illustrate the behavior of MOPNAR with respect to MRQAR-MOPNAR. The experiments were performed using different subsets of the higgs dataset.

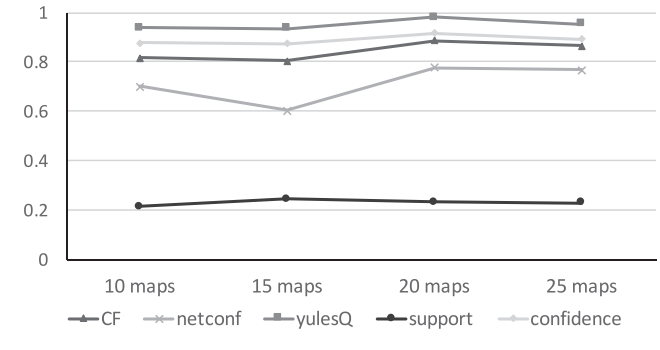
Fig. 6 presents the number of examples of each subset, the number of split or maps computed by MRQAR-MOPNAR for each version of the



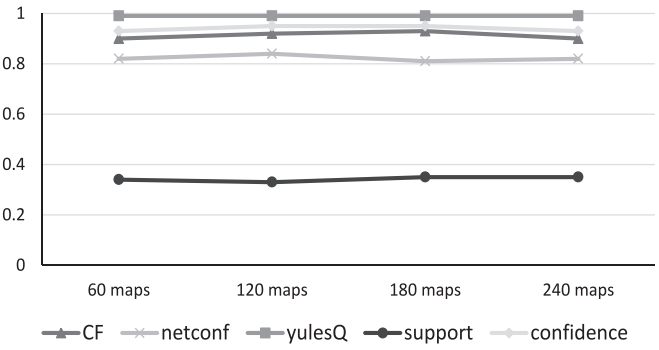
(a) Performance in susy



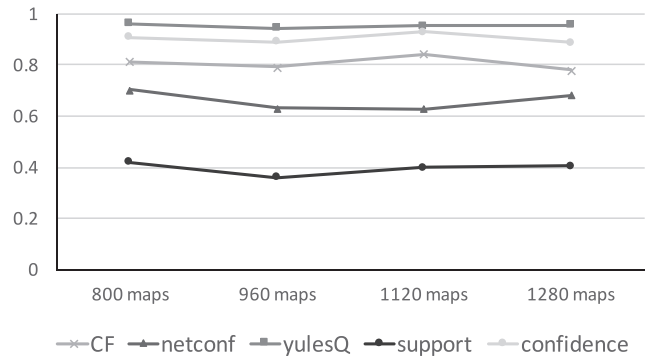
(b) Performance in higgs



(c) Performance in epsilon



(d) Performance in ECBLD14 (reduced)



(e) Performance in ECBLD14 (complete)

Fig. 7. Average results of the support, confidence, CF, netconf and yulesQ measures when the number of maps increase with the susy, higgs, epsilon and ECBDL14 (reduced and complete) datasets for MRQAR-MOPNAR.

dataset ("#Maps") and the runtime expended by MOPNAR and MRQAR-MOPNAR. The number of examples is the result of multiplying a fixed dataset size by the number of maps specified in order to analyze how the runtime is evolved according to an increasing number of examples and a fixed number of instances per map or partition. The number of instances for each map should be selected considering a problem size in which standard methods can converge properly. That is, when the dimensionality curve allows us to address these problems in a reasonable time and therefore the execution times in each MapReduce phase are suitable. In particular, we have considered 25,000 examples per split (map) for MRQAR-MOPNAR. The number of iterations of MRQAR-MOPNAR can be calculated dividing the $N_{eval-global}$ parameter by N_{eval} parameter. Considering the parameters provided in Table 3, the number of iterations of MRQAR-MOPNAR for all the experimental results presented in Fig. 6 is 100000/25000 = 4.

We can highlight how MOPNAR scales exponentially when the number of examples increase. Notice that, MOPNAR was not able to manage the subset with 3,200,000 examples. Moreover, the time difference between MRQAR-MOPNAR and MOPNAR is higher while the size of the problem increase, being MRQAR-MOPNAR eighteen times faster than MOPNAR with 1,600,000 examples.

Several measures that evaluate the rules according to different quality properties have been considered to analyze the performance of the MOPNAR and MRQAR-MOPNAR. These metrics solve the optimization problems of support and confidence measures as stated Section 2.1 and guarantees the quality of the rules found. Note that the

Table 6
Average runtime in seconds for MRQAR-MOPNAR depending on the number of maps on the poker, susy and higgs datasets.

Datasets	*Maps*	second	hh:mm:ss
susy	25	16,208	4:30:08
	50	7962	2:12:42
	75	6336	1:45:36
	100	5256	**1:27:36**
higgs	50	27,009	7:30:09
	100	15,702	4:21:42
	150	11,478	3:11:18
	200	10,160	**2:49:20**
epsilon	10	55,700	15:28:20
	15	48,722	13:32:02
	20	23,328	6:28:48
	25	16,745	**4:39:05**
ECBDL14 reduced	60	30,043	8:20:43
	120	17,320	4:48:40
	180	13,508	3:45:08
	240	12,112	**3:21:52**
ECBDL14 complete	800	69,613	19:20:13
	960	45,234	12:33:54
	1120	29,736	8:17:10
	1280	29,830	**8:15:36**

quality measures presented for MRQAR-MOPNAR have been calculated for all the instances of the input dataset. Table 4 shows the average of the results obtained by MOPNAR and MRQAR-MOPNAR over the subsets of the higgs dataset that MOPNAR was able to be executed. The number of QARs discovered is indicated in column #*R*; the average support, confidence, lift, CF, netconf and yules'Q values are presented in columns *Support, Confidence, Lift, Conviction, Certain Factor* (CF), *Netconf* and *YulesQ*, respectively. the average number of attributes that belong to the rules is shown in column *Attributes*; and the last column %*Trans* represents the percentage of instances that the discovered QARs satisfy. The value 100 - *trNotCover* threshold could be a lower bound of the percentage of instances of the dataset covered by the resulting QARs. This fact can only happen in the iterations in which the percentage of instances not covered is less than *trNotCover*. In other case, it cannot be ensured due to the $N_{eval-Global}$ parameter can be achieved before the *trNotCover* parameter is reached, therefore, a higher value $N_{eval-Global}$ could be needed.

The conclusions described below can be drawn from the analysis performed in Table 4:

The rule sets discovered by MRQAR-MOPNAR and MOPNAR present similar values for the quality measures, being better for MRQAR-MOPNAR when the size of the problem increase. The framework proposed is able to obtain a reduced set of quality QARs that cover more than 99% of instances in all the datasets. These rules provide interesting knowledge and different features of the problem. MRQAR-MOPNAR mines smaller number of rules than MOPNAR without consequences in the increasing of the average support (almost less than 0.5 in all datasets).

4.3. Analysis of the MRQAR-MOPNAR behavior

This section presents a set of experiments to illustrate the behavior of MRQAR-MOPNAR over the datasets considered in the study. Five different values for the number of maps for each dataset have been used to facilitate the comprehensibility of the performed analysis. The number of maps selected is different between the datasets to process a similar number of examples per split approximately.

The results obtained by MRQAR-MOPNAR are shown in Table 5, this kind of table was described in Section 4.2, but in this case we have added the number of maps (*Maps*) and the number of examples per split (*ExSplit*). The best value obtain for each measure is highlighted in bold for each big data problem. Fig. 7 shows the average results of the support, confidence, CF, netconf and yulesQ measures for MRQAR-MOPNAR when the number of maps increase for the susy, higgs, epsilon and ECBDL14 reduced and complete datasets, respectively. Notice that we have not showed the lift and conviction measures in the figure because their range is not bounded, which makes them difficult to represent with the rest of the measures. As can be observed in Table 5 and Fig. 7, the interestingness measures start with high values that remain almost unchanged when the method uses a larger number of maps. MRQAR-MOPNAR obtains set of rules that cover different areas of the problem, which allow us to achieve almost the 100% of covered instances of the datasets for all the number of maps used.

On the other hand, we analyze the runtime expended by MRQAR-MOPNAR using different number of maps for the datasets analyzed in the experimental study. Table 6 shows the runtime expended in seconds. Furthermore, the runtime is also presented using the hh:mm:ss structure (where *hh, mm* and *ss* represent the hours, minutes and seconds spent, respectively) by our proposal when the number of maps increases. The best average runtime is highlighted in bold for each dataset considered. Fig. 8 shows the relationship between the runtime and the number of maps.

In a first glance, it can be observed that the runtime necessary to execute the proposed approach is admissible. MRQAR-MOPNAR improves its runtime as the number of mappers are enlarged, obtaining a good speed gain for all the different number of mappers. However, this speed gain is less impressive as the number of mappers used is higher.

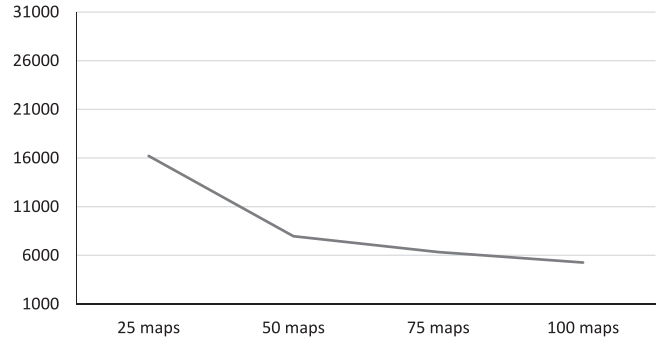
Therefore, analyzing the MRQAR-MOPNAR behavior presented in this section, we will use the maximum number of maps for all datasets (100, 200, 25, 240 and 1280 maps for susy, higgs, epsilon and ECBLD14 reduced and complete datasets, respectively) in the following experimental study due to our proposal tends to obtain a reduction on the runtime and more precise results when a higher number of mappers are used.

4.4. Comparison with PFP-Spark algorithm for big data problems

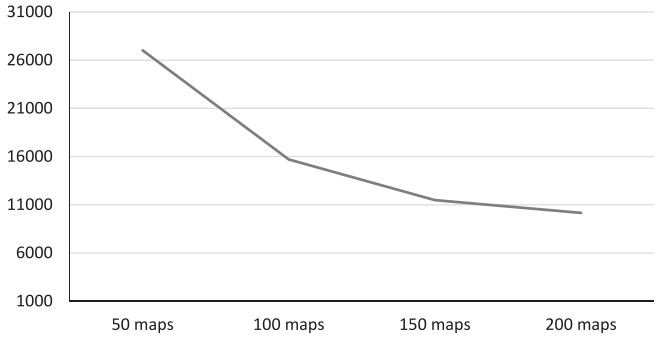
This section is devoted to compare the results obtained by MRQAR-MOPNAR against another existing approaches that deal with Big Data problems to mine association rules. Specifically, we have used a distributed adaptation of the well-known FP-Growth named PFP algorithm implemented in Spark. The PFP algorithm presents a MapReduce approach to parallelize the FP-Growth algorithm [67]. This method only finds frequent itemsets from datasets with binary or discrete values then, several procedures have been carried to make this algorithm comparable. In particular, we have used a procedure afterwards to generate association rules and we have discretized the datasets using a Spark implementation [68] of the Fayyad's discretized [69] based on Minimum Description Length Principle.

Table 7 shows the average of the results obtained by MRQAR-MOPNAR and PFP over the datasets analyzed. Note that no results are shown for epsilon and ECBDL14 (complete) datasets because FPF was not able to work with these datasets due to scalability issues in spite of using several parameter configuration settings. The structure of the table was described in Section 4.2 but in this case we have also included the runtime devoted in seconds by each method analyzed (*Runtime*(*s*)) and by the discretization method (*Discretize*(*s*)). The real attributes of each dataset have been discretized into 25 intervals. The results achieved by PFP using a minimum confidence threshold (*MinConf*) of 0.5 and 0.8, in addition to a minimum support thresholds (*MinSup*) ranging from 0.2 to 0.4 are also presented. Note that PFP algorithm was unable to be executed using a minimum support threshold of 0.2 for the ECBDL14 (reduced) problem due to the huge amount of rules generated.

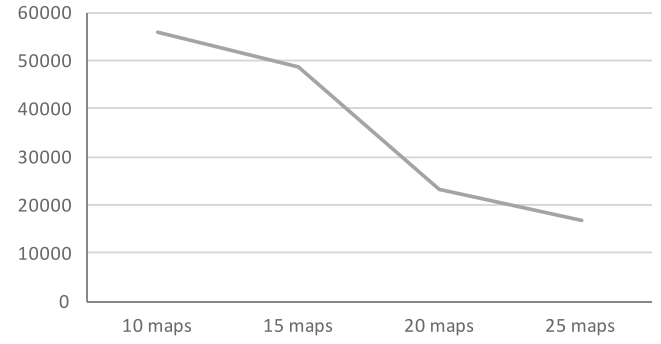
Several conclusions can be drawn after analyzing the reported results. MRQAR-MOPNAR overcomes the PFP algorithm in all the interestingness measures considered. The coverage of the datasets achieved for the rules obtained by MRQAR-MOPNAR for all the problems is



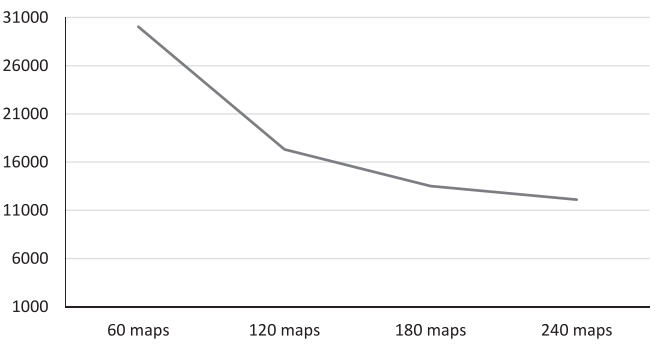
(a) Runtime (seconds) in susy



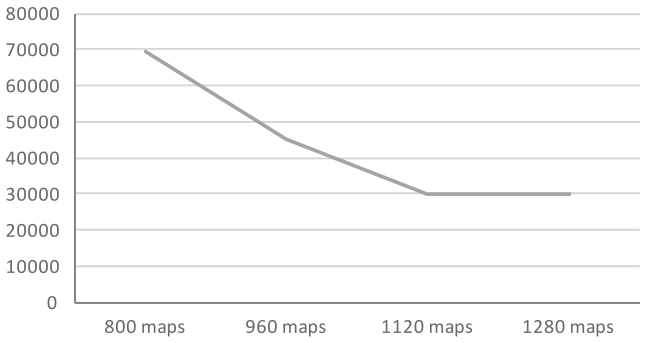
(b) Runtime (seconds) in higgs



(c) Runtime (seconds) in epsilon



(d) Runtime (seconds) in ECBLD14 (reduced)



(e) Runtime (seconds) in ECBLD14 (complete)

Fig. 8. The runtime (seconds) over the susy, higgs epsilon, ECBLD14 (reduced and complete) datasets for MRQAR-MOPNAR.

almost 100%. Additionally, *Certain Factor* and *YulesQ* measures are almost 1, which is the highest possible value. Indeed, the conviction value is ∞ for susy and ECBDL14 (reduced) problems which means perfect implications.

In the case of susy problem any rule was obtained by PFP using the configurations *MinSup* 0.3 and *MinConf* 0.8, in addition to *MinSup* 0.4. For the remaining configurations, it can be observed that a very low number of rules are obtained by PFP. This fact shows the inability of this method of obtaining good quality rules for big data problems. The quality measures obtained for *MinSup* 0.2 and 0.3 using *MinConf* 0.5 are quite similar, although the number of rules and the percentage of instances is higher for the first configuration. For higgs problem, despite the fact that the percentage of instances, support and confidence values are good values, PFP obtains values near 0 for *Certain Factor, Netconf* and *YulesQ* measures that represent independence between antecedent and consequent and denote the poor quality of the rules obtained. In fact, the configuration *MinSup* 0.3 and *MinConf* 0.5 reports negative values for that measures that means that antecedent and consequent are independent. In contrast to these problems, PFP reports better results for ECBDL14 (reduced) problem.

Finally, the runtime required for the PFP algorithm in all the studied datasets is better than MRQAR-MOPNAR. However, that algorithm requires a previous discretization process to deal with quantitative datasets, so that the total time required for PFP is highly increased when the discretization is considered. Note that the discretization runtime is quite higher for susy and higgs problems in contrast to ECBDL14 (reduced) due to the huge amount of possible values that their real attributes have. In conclusion, MRQAR-MOPNAR performs better than PFP regardless of the data set used without prior discretization process nor minimum thresholds of the measures to increase the quality of the rules

Table 7
Comparison of PFP with MRQAR-MOPNAR.

Algorithm	*MinSup.*	*MinConf.*	*#R*	*Support*	*Confidence*	*Lift*	*Conviction*	*CF*	*Netconf*	*YulesQ*	*Attributes*	*%Trans*	*Runtime(s) + discretize(s)*
			susy										
PFP	0.2	0.5	37	0.27	0.63	1.12	1.23	0.14	0.11	0.22	2.24	95.78	23 + 5627
PFP	0.2	0.8	2	0.26	0.84	1.38	2.43	0.59	0.33	0.67	2.5	62.7	20 + 5627
PFP	0.3	0.5	8	0.36	0.64	1.13	1.22	0.15	0.15	0.28	2	63.88	14 + 5627
PFP	0.3	0.8	0	0	0	0	0	0	0	0	0	0	16 + 5627
PFP	0.4	0.5	0	0	0	0	0	0	0	0	0	0	19 + 5627
PFP	0.4	0.8	0	0	0	0	0	0	0	0	0	0	15 + 5627
MRQAR-MOPNAR			**515**	**0.49**	**0.97**	**7.30**	∞	**0.96**	**0.84**	**0.99**	**1.45**	**99.99**	**5256 + 0**
			higgs										
PFP	0.2	0.5	33	0.28	0.67	1.01	1.01	0.01	0.01	0.01	2.21	95.72	61 + 8222
PFP	0.2	0.8	13	0.29	0.80	1	1	0	0	0	2.23	80.29	59 + 8222
PFP	0.3	0.5	9	0.39	0.74	0.99	0.98	–0.01	–0.01	–0.03	2	80.29	25 + 8222
PFP	0.3	0.8	6	0.37	0.80	1	1	0	0	0	2	80.29	**17 + 8222**
PFP	0.4	0.5	4	0.44	0.67	1	1	0	0	0	2	64.46	**17 + 8222**
PFP	0.4	0.8	2	0.44	0.8	1	1	0	0	0	2	64.46	22+ 8222
MRQAR-MOPNAR			**403**	**0.56**	**0.96**	**2.96**	**68.39**	**0.94**	**0.61**	**0.99**	**1.53**	**99.99**	10160 + 0
			epsilon										
MRQAR-MOPNAR			35	0.23	0.89	4.53	419.75	0.86	0.77	0.95	7.68	98.05	16745 + 0
			ECBDL14 reduced										
PFP	0.3	0.5	**443**	0.35	0.87	1.55	9.22	0.57	0.46	0.69	3.26	99.75	2471 + 107
PFP	0.3	0.8	336	0.35	0.92	1.59	9.22	0.62	0.48	0.70	3.42	99.75	2128 + 107
PFP	0.4	0.5	83	**0.44**	0.90	1.44	2.74	0.47	0.43	0.59	2.43	99.59	216 + 107
PFP	0.4	0.8	80	**0.43**	0.92	1.46	2.81	0.49	0.44	0.61	2.45	99.59	**168 + 107**
MRQAR-MOPNAR			196	0.35	**0.93**	**4.63**	∞	**0.90**	**0.81**	**0.99**	**1.74**	**100**	12112 + 0
			ECBDL14 complete										
MRQAR-MOPNAR			14	0.41	0.89	2.31	6.13	0.78	0.68	**0.96**	1.40	99.95	29,736 + 0

obtained.

We have applied a statistical analysis in order to assess whether significant differences exist among the results obtained in MRQAR-MOPNAR and PFP. We have considered the use of Bayesian tests due to the few datasets used in the experimental analysis. The use of this type of tests has experienced a great growth in the machine learning community in the last years due to the shortcomings of frequentist reasoning, and especially of the null hypothesis significance tests (NHST) [70].

Regarding the datasets used in the statistical analysis, susy, higgs and ECBDL14 (reduced) have been only considered since PFP was unable to return rules for datasets ECBDL14 (complete) and epsilon.

According to PFP, we have selected the configurations where the measures of this algorithm have had better values. In particular, MinSup. 0.2 and MinConf. 0.8 for susy and higgs datasets in addition to MinSup. 0.3 and MinConf. 0.8 for ECBLD14 (reduced).

In order to compare the two algorithms statistically, we have considered multiple quality measures. We have selected confidence, lift, certainty factor (CF), NetConf and YulesQ (see Table 1). Note that conviction measure has not been used because MRQAR-MOPNAR obtains infinity values in some datasets.

The mean values of these measures have been adopted to *MeanS* because all of them are not in the same domain. *MeanS* takes values in the domain [0,1]. The independence value for each measure is represented by the worst value, which means that it does not provide new knowledge to the user.

The *MeanS* for each measure is defined as:

- For the measures CF, NetConf and YulesQ, whose domain is $[-1,1]$:

$$MeanS = \begin{cases} \frac{|meanValue|}{2} & meanValue \leq 0 \\ \frac{|meanValue|}{2} + 0.5 & otherwise \end{cases} \quad (1)$$

- For the measure Lift, whose domain is [0, inf]:

$$MeanS = \begin{cases} 1 - \frac{0.5}{|meanValue|} & meanValue > 1 \\ 0.5 - \frac{|meanValue|}{2} & 0 \leq meanValue \leq 1 \end{cases} \quad (2)$$

- *MeanS* of confidence measure has not been adopted since the domain is [0,1].

where *meanValue* is the mean value obtained for each measure in a dataset.

In order to compare the results, we have used the De Campos's test [71] in which the dominance statement for two given algorithms *A, B* is defined as $D^{(BA)} = [\ <, \ldots, >,\ <\ ]$, where $D_i^{(BA)}$ is $<$ if *B* gets a score less than *A* for the measure $i = 1, \ldots, m$. The aim of this test is to calculate the probability of each of the 2^m possible configurations. To do that, this test initially considers a probability distribution that assigns each configuration the same probability and then adjusts the probabilities based on the algorithm's results. The test results show that the probability of MRQAR being better than PFP in all analyzed measures is 0.944.

From these results we can assure that MRQAR is better than PFP with a high probability, presenting better results in all analyzed datasets. In addition, it is more scalable and capable of handling large datasets at an appropriate computational cost.

5. Conclusions

A new incremental parallel and general framework named MRQAR has been presented in this paper to discover QARs in Big Data problems. Our proposal is designed following a MapReduce scheme using Apache Spark, a well-known solution to face Big Data problems. MRQAR is a general purpose framework that can execute any sequential association rule algorithm to obtain quantitative association rules in Big Data, obtaining the best rules in terms of quality for the entire dataset. This proposal must be used when the standard algorithm to find QARs is not able to be executed in big data scenarios, or it has difficulty to obtain good results due to the size of the search space. As a particular case study, the non-distributed MOPNAR algorithm is integrated within the

Map of the first phase of MRQAR named MRQAR-MOPNAR.

The results obtained show that our proposal using MOPNAR in the first phase discovers reduced sets of high quality QARs from the Big Data problems analyzed. Furthermore, the rules generated present a high coverage of the datasets (more than 99% in most cases), providing with interesting knowledge of the whole datasets to the user. The runtime expended by our proposal is adequate in all datasets. When comparing the results obtained with another algorithm to mine association rules from Big Data problems, it can be observed that MRQAR-MOPNAR overcomes the PFP algorithm in terms of better values for interesting measures and coverage of the dataset.

Acknowledgments

This work has been partially funded by the Ministry of Economy and Competitiveness under the TIN2017-89517-P, TIN2014-55894-C2-1-R and TIN2017-88209-C2-2-R projects.

References

[1] M. Martínez-Ballesteros, J. Bacardit, A. Troncoso, J. Riquelme, Enhancing the scalability of a genetic algorithm to discover quantitative association rules in large-scale datasets, Integr. Comput. Aided Eng. 22 (1) (2015) 21–39.
[2] A. Bechini, F. Marcelloni, A. Segatori, A MapReduce solution for associative classification of big data, Inf. Sci. (Ny) 332 (2016) 33–55.
[3] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[4] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, F. Herrera, Big data: tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce, Inf. Fus. 42 (2018) 51–61.
[5] A. Fernández, S. del Río, V. López, A. Bawakid, M.J. del Jesus, J.M. Benítez, F. Herrera, Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks, Wiley Interdiscip. Rev. Data Mining Knowl. Disc. 4 (5) (2014) 380–409.
[6] C.P. Chen, C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: a survey on big data, Inf. Sci. (Ny) 275 (2014) 314–347.
[7] Apache Spark: lightning-fast cluster computing, 2017, [Online; consulted in July 2017] (http://spark.apache.org/).
[8] H. Karau, A. Konwinski, P. Wendell, M. Zaharia, Learning Spark: Lightning-Fast Big Data Analytics, first, O'Reilly Media, Inc., 2015.
[9] MLlib website, 2017, [Online; consulted in July 2017] (https://spark.apache.org/mllib/).
[10] D. Martín, A. Rosete, J. Alcalá-Fdez, F. Herrera, A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules, IEEE Trans. Evol. Comput. 18 (1) (2014) 54–69.
[11] H. Li, Y. Wang, D. Zhang, M. Zhang, E. Chang, PFP: parallel FP-growth for query recommendation, Proceedings of the ACM Conference on Recommender Systems, RecSys '08, ACM, 2008, pp. 107–114.
[12] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, Proceedings of the ACM SIGMOD Inter. Conference on Management of Data, (1993), pp. 207–216.
[13] L. Geng, H. Hamilton, Interestingness measures for data mining: a survey, ACM Comput. Surv. 38 (3) (2006) 1–42.
[14] M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, J. Riquelme, Obtaining optimal quality measures for quantitative association rules, Neurocomputing 176 (2016) 36–47.
[15] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: generalizing association rules to correlations, Proceedings of the ACM SIGMOD, 26 (1997), pp. 265–276.
[16] S. Brin, R. Motwani, J.D. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, Proceedings of the ACM SIGMOD, (1997), pp. 265–276.
[17] E. Shortliffe, B. Buchanan, A model of inexact reasoning in medicine, Math. Biosci. 23 (1975) 351–379.
[18] K.-I. Ahn, J.-Y. Kim., Efficient mining of frequent itemsets and a measure of interest for association rule mining. J. Inform. Know. Manag. 3 (3) (2004) 245–257.
[19] P. Tan, V. Kumar, Interestingness measures for association patterns: a perspective, Proceedings of the KDD-2000 workshop on post processing in machine learning and data mining, (2000).
[20] D. Adhikary, S. Roy, Trends in quantitative association rule mining techniques, Proceedings of the IEEE Second International Conference on Recent Trends in Information Systems (ReTIS), (2015), pp. 126–131.
[21] T. Fukuda, Y. Morimoto, S. Morishita, T. Tokuyama, Mining optimized association rules for numeric attributes, J. Comput. Syst. Sci. 58 (1) (1999) 1–12.
[22] R. Srikant, R. Agrawal, Mining quantitative association rules in large relational tables, Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '96, ACM, New York, NY, USA, 1996, pp. 1–12.
[23] Y. Guo, J. Yang, Y. Huang, An effective algorithm for mining quantitative association rules based on high dimension cluster, Proceedings of the Fourth International Conference on Wireless Communications, Networking and Mobile Computing, (2008), pp. 1–4.
[24] J. Yang, Z. Feng, An effective algorithm for mining quantitative associations based on subspace clustering, Proceedings of the International Conference on Networking and Digital Society, 1 (2010), pp. 175–178.
[25] Y. Aumann, Y. Lindell, A statistical theory for quantitative association rules, J. Intel. Inf. Syst. 20 (3) (2003) 255–283.
[26] H. Ishibuchi, Y. Nojima, Optimization of scalarizing functions through evolutionary multiobjective optimization, Lect. Notes Comput. Sci. 4403 (2006) 51–65.
[27] H. Zheng, J. He, G. Huang, Y. Zhang, Optimized fuzzy association rule mining for quantitative data, Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), (2014), pp. 396–403.
[28] X. Yan, C. Zhang, S. Zhang, Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support, Expert Syst. Appl. 36 (2) (2009) 3066–3076.
[29] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, J. Riquelme, Mining quantitative association rules based on evolutionary computation and its application to atmospheric pollution, Integr. Comput. Aided Eng. 17 (2010) 227–242.
[30] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, J. Riquelme, An evolutionary algorithm to discover quantitative association rules in multidimensional time series, Soft Comput. 15 (10) (2011) 2065–2084.
[31] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evolut. Comput. 3 (4) (1999) 257–271.
[32] S. Ventura, J. Luna, Multiobjective Approaches in Pattern Mining, Springer International Publishing, Cham, pp. 119–139.
[33] D. Martín, A. Rosete, J. Alcalá-Fdez, F. Herrera, QAR-CIP-NSGA-II: a new multiobjective evolutionary algorithm to mine quantitative association rules, Inform. Sci. 258 (2014) 1–28.
[34] M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, J.C. Riquelme, Improving a multi-objective evolutionary algorithm to discover quantitative association rules, Knowl. Inf. Syst. 49 (2) (2016) 481–509.
[35] M. Almasi, M.S. Abadeh, Rare-pears: a new multi objective evolutionary algorithm to mine rare and non-redundant quantitative association rules, Knowl.-Based Syst. 89 (2015) 366–384.
[36] D. Martín, J. Alcalá-Fdez, A. Rosete, F. Herrera, Nicgar: a niching genetic algorithm to mine a diverse set of interesting quantitative association rules, Inf. Sci. (Ny) 355 (2016) 208–228.
[37] J. Li, X. Ye, Study on linked list-based algorithm for metarule-guided mining of multidimensional quantitative association rules, Proceedings of the Third International Conference on Natural Computation (ICNC 2007), 1 (2007), pp. 300–304.
[38] J. Dean, S. Ghemawat, Mapreduce: a flexible data processing tool, Commun. ACM 53 (1) (2010) 72–77.
[39] Apache hadoop, 2017, [Online; consulted in July 2017] (http://hadoop.apache.org/).
[40] T. White, Hadoop: The Definitive Guide, First, O'Reilly Media, Inc., 2009.
[41] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing, Proceedings of the Ninth USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2012. 2–2
[42] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, X. D., R. Xin, J. Franklin, R. Zadeh, M. Zaharia, A. Talwalkar, MLLib: machine learning in apache spark, J. Mach. Learn. Res. 17 (34) (2016) 1–7.
[43] D. Adhikary, S. Roy, Issues in Quantitative Association Rule Mining: A Big Data Perspective, Springer Singapore, Singapore, pp. 377–385.
[44] S. Ventura, J. Luna, Scalability in Pattern Mining, Springer International Publishing, Cham, pp. 177–190.
[45] A. Saabith, E. Sundararajan, A. Bakar, Parallel implementation of apriori algorithms on the hadoop-mapreduce platform–an evaluation of literature, J. Theor. Appl. Inf. Technol. 85 (3) (2016) 321–351.
[46] F. Kovács, J. Illés, Frequent itemset mining on hadoop, Proceedings of the IEEE Ninth International Conference on Computational Cybernetics (ICCC), (2013), pp. 241–245.
[47] .R.M. Hazarika M., Mapreduce based eclat algorithm for association rule mining in datamining: mr_eclat, Int. J. Comput. Sci. Eng. 3 (1) (2014) 19–28.
[48] S. Moens, E. Aksehirli, B. Goethals, Frequent itemset mining for big data, Proceedings of the IEEE International Conference on Big Data, (2013), pp. 111–118.
[49] M.J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, New algorithms for fast discovery of association rules, Proceedings of the Third International Conference on Knowledge Discovery and Data mining (KDD'97), AAAI Press, 1997, pp. 283–286.
[50] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, Proceedings of the International Conference on Very Large Databases, (1994), pp. 478–499.
[51] L. Greeshma, G. Pradeepini, Input split frequent pattern tree using mapreduce paradigm in hadoop, J. Theor. Appl. Inf. Technol. 84 (2) (2016) 260–271.
[52] S. Salah, R. Akbarinia, F. Masseglia, A highly scalable parallel algorithm for maximally informative k-itemset mining, Knowl. Inf. Syst. 50 (1) (2017) 1–26.
[53] X. Yusheng, C. Zhengzhang, D. Palsetia, G. Trajcevski, A. Agrawal, A. Choudhary, Silverback+: scalable association mining via fast list intersection for columnar social data, Knowl. Inf. Syst. 50 (3) (2017) 969–997.
[54] M. Lin, P. Lee, S. Hsueh, Apriori-based frequent itemset mining algorithms on MapReduce, Proceedings of the Sixth International Conference on Ubiquitous Inform. Management and Communication, ICUIMC '12, ACM, 2012, pp. 1–8.
[55] L. Wang, L. Feng, J. Zhang, P. Liao, An efficient algorithm of frequent itemsets mining based on mapreduce, J. Inform. Comput. Sci. 11 (8) (2014) 2809–2816.
[56] H. Qiu, R. Gu, C. Yuan, Y. Huang, Yafim: a parallel frequent itemset mining

algorithm with spark, Proceedings of the IEEE International Parallel Distributed Processing Symposium Workshops, (2014), pp. 1664–1671.
[57] F. Zhang, M. Liu, F. Gui, W. Shen, A. Shami, Y. Ma, A distributed frequent itemset mining algorithm using spark for big data analytics, Clust. Comput. 18 (4) (2015) 1493–1501.
[58] K.K. Sethi, D. Ramesh, Hfim: a spark-based hybrid frequent itemset mining algorithm for big data processing, J. Supercomput. (2017) 1–17.
[59] H.V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J.M. Patel, R. Ramakrishnan, C. Shahabi, Big data and its technical challenges, Commun. ACM 57 (7) (2014) 86–94.
[60] P. Baldi, P. Sadowski, D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nat. Commun. 5 (2014).
[61] K. Bache, M. Lichman, Uci machine learning repository, 2017, [Online; consulted in July 2017] (http://archive.ics.uci.edu/ml).
[62] Epsilon in the libsvm website, 2016, [Online; consulted in December 2017] (http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary.html#epsilon).
[63] Pascal large scale learning challenge, 2008, [Online; consulted in December 2017] (http://largescale.ml.tu-berlin.de).
[64] Evolutionary computation for big data and big learning workshop. data mining competition 2014: Self-deployment track., 2014, (http://cruncher.ncl.ac.uk/bdcomp/).
[65] I. Triguero, S. del Ro, V. Lpez, J. Bacardit, J.M. Bentez, F. Herrera, Rosefw-rf: the winner algorithm for the ecbdl14 big data competition: an extremely imbalanced big data bioinformatics problem, Knowl. Based Sys.t 87 (Supplement C) (2015) 69–79. Computational Intelligence Applications for Data Science
[66] H. Li, Q. Zhang, Multiobjective optimization problems with complicated pareto sets, MOEA/d and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.
[67] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Mining Know. Disc. 8 (1) (2004) 53–87.
[68] S. Ramírez-Gallego, S. García, H. Mourio-Talin, D. Martínez-Rego, V. Boln, A. Alonso-Betanzos, J. Benitez, F. Herrera, Distributed entropy minimization discretizer for big data analysis under apache spark, Proceedings of the IEEE BigDataSE Conference, (2015).
[69] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of the IJCAI, (1993), pp. 1022–1029.
[70] A. Benavoli, G. Corani, J. Demsar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis, J. Mach. Learn. Res. 18 (2017) 2653–2688.
[71] C.P. de Campos, A. Benavoli, Joint analysis of multiple algorithms and performance measures, New Gen. Comput. 35 (1) (2017) 69–86.