Noname manuscript No.
(will be inserted by the editor)

Optimizing Partition Granularity, Membership Function Parameters, and Rule Bases of Fuzzy Classifiers for Big Data by a Multi-objective Evolutionary Approach

IMPORTANT: The final publication is available at link.springer.com

Marco Barsacchi · Alessio Bechini · Pietro Ducange · Francesco Marcelloni

Abstract *Background/Introduction* - Classical data mining algorithms are considered inadequate to manage the volume, variety, velocity, and veracity aspects of Big Data. The advent of a number of open source cluster computing frameworks has opened new interesting perspectives for handling the volume and velocity features. In this context, thanks to their capability of coping with vague and imprecise information, distributed fuzzy models appear to be particularly suitable for handling the variety and veracity features of big data. Moreover, the interpretability of fuzzy models may assume a particular relevance in the context of big data mining.

Methods - In this work, we propose a novel approach for generating, out of big data, a set of fuzzy rule-based classifiers characterized by different optimal trade-offs between accuracy and interpretability. We extend a state-of-the-art distributed multi-objective evolutionary learning scheme, implemented under the Apache Spark environment. In particular, we exploit a recently proposed distributed fuzzy decision tree learning approach for generating an initial rule base that serves as input to the evolutionary process. Furthermore, we integrate the evolutionary learning scheme with an ad-hoc strategy for the granularity learning of the fuzzy partitions, along with the optimization of both the rule base and the fuzzy set parameters.

Results and Conclusions - Experimental investigations show that the proposed approach is able to generate fuzzy rule-based classifiers that are significantly less complex than the ones generated by the original multi-objective evolutionary learning scheme, while keeping the same accuracy levels.

Keywords Big Data Mining · Multi-objective Evolutionary Fuzzy Systems · Fuzzy Classification Models · Distributed Algorithms

Marco Barsacchi, Alessio Bechini, Francesco Marcelloni
Dipartimento di Ingegneria dell'Informazione: University of Pisa, Pisa, Italy
E-mail: {marco.barsacchi, alessio.bechini, francesco.marcelloni}@unipi.it

Pietro Ducange
SMART Engineering Solutions & Technologies (SMARTEST) Research Centre
eCAMPUS University, Novedrate, Italy
E-mail: pietro.ducange@uniecampus.it

1 Introduction

Data mining allows extracting useful knowledge from data. In the last decades, data mining has been considerably investigated and a huge number of different techniques have been proposed for generating, for instance, descriptive models in clustering and frequent pattern analysis, and predictive models in classification and regression tasks [35]. Data mining is closely related to cognitive computation. Indeed, as discussed in [20], cognitive computation helps improve human decision-making: data mining models are often adopted for decision making issues, such as image recognition [16], disease identification [3], and managing the energy consumption in wireless sensor networks [38].

We are currently experiencing the *Big Data Era* [45] and classical data mining algorithms appear to be inadequate to manage Big Data. Indeed, Big Data are characterized by the four "V"s, namely volume, variety, velocity and veracity: large *volumes* of data, which are often produced at very high speed and need to be elaborated in almost real time (*velocity)*, are generated by different sources and may have different formats (*variety)* [11] and trustworthiness (*veracity*). These data represent a very important source of *added-values* in several contexts, such as in marketing strategies [23], industrial applications [55], and Internet of Things [2].

Big Data are of prominent importance also for their relationship with cognitive computing systems [1]: they naturally learn from people and from the huge amount of data they are involved with, typically by exploiting computational intelligence and machine learning algorithms. In the last years several researchers have introduced data mining approaches purposely designed and implemented for Big Data [48, 58]. Most of these approaches have employed specific distributed frameworks, such as Apache Hadoop [57] and Apache Spark [59], which have been recently proposed with the aim of dealing with data storage and elaboration of Big Data. Further, most of the recent contributions in the field exploit the MapReduce paradigm [22] for implementing both descriptive and predictive models, with the additional benefit of the possibility to exploit computing resources on the Cloud [31].

As an example, in [39] and in [42], authors have proposed a distributed version of two famous clustering algorithms, i.e. DB-SCAN and Fuzzy C-Means, respectively, developed using the Apache Hadoop framework. A fuzzy version of Random Forests has been implemented over the same framework as well [15].

Recently, in [14] and [43], Apache Spark implementations of associative classification models and of the KNN classifier, respectively, have been discussed. Also big social data analysis has taken advantage of the use of such distributed computing frameworks [47]. A recent work highlights the main advances, challenges and objectives in designing, developing and using data mining and machine learning algorithms for Big Data [60].

In the context of predictive models, in the last years, a number of contributions employing Fuzzy Models (FMs) for handling Big Data have been proposed [25, 28, 32, 41, 44, 51, 52, 53]. As stated in [29], FMs are particularly suitable for handling the variety and veracity of Big Data. This is mainly due to their good capability of coping with vague, imprecise, and uncertain concepts. It is worth underlining that fuzzy logic has been recognized also as an important tool to keep the fidelity of psychological interpretation of emotion [12], opening up new ways to analyze the sentiment contents of huge amounts of data available from web sources. From a more technical point of view, the use of overlapped fuzzy labels ensures a good coverage of the problem space. This issue is especially relevant when dealing with very large datasets that may be divided into a number of heterogeneous chunks, such as in the MapReduce programming paradigm. Actually, the different chunks may influence in a different way the parameter learning process of the classification model.

To the best of our knowledge, the first proposal regarding FM for Big Data classification is the Chi-FRBCS-BigData [51] model. It is a fuzzy rule-based classifier (FRBC), developed according to the MapReduce paradigm and based on the approach described by Chi et al. [17]. In the work discussed in [41], the Chi-FRBCS-BigData algorithm has been adapted for handling imbalanced big datasets, and the effects of the granularity of fuzzy partitions, when using the same algorithm, have been studied in [30]. Recently, in the work published in [25], the CHI-BD algorithm has been introduced: it is novel distributed version of the Chi et al's approach [17], with improved results with respect to Chi-FRBCS-BigData. Basically, CHI-BD is the exact distributed implementation of the Chi et al's approach, whereas Chi-FRBCS-BigData is an approximated implementation. More details can be found in [25] and [51], respectively. Notably, efficient MapReduce solutions based on the CHI-BD algorithm have been proposed also for the Prototype Reduction problem [26]

Similarly to the aforementioned approaches, most of the contributions proposed so far focus on the design and development of FMs in a distributed environment, especially considering the accuracy of the models. Also two recent works discuss the good results obtained by two novel fuzzy classification models for Big Data. The two models are respectively based on Fuzzy Associative Classifiers [52] and Distributed Multi-Way Fuzzy Decision Trees (DMFDTs) [53]. Even if the accuracy of the obtained classifiers is good, the complexity of the relative models, in terms of both number of rules and number of decision nodes, is very high. The greater the complexity, the lower the interpretability of the FMs. However, the interpretability is a very important feature that characterizes FMs, and is particularly relevant in the context of Big Data as well [29, 56]. New methods to generate both accurate and interpretable FMs are currently investigated in the research community on fuzzy models [24].

Interpretability is a subjective concept: it is hard to find a worldwide agreed definition and consequently a universal measure of interpretability. A taxonomy of interpretability measures for fuzzy rule-based models has been proposed [33] by considering the two distinct dimensions of semantics and complexity, at the rule base (RB) and data base (DB) levels. As regards the DB, the semantic interpretability is usually evaluated in terms of integrity of the fuzzy partitions, whereas the complexity is evaluated in terms of number of fuzzy sets. As regards the RB, the interpretability is mostly analyzed in terms of complexity and one of the most used metrics is the Total Rule Length (*TRL*) [18, 36, 37], that is, the total number of conditions used in the RB. In the learning of interpretable FMs, the importance of other factors like *rule relevance* has been experimentally studied as well [49].

In the framework of "classical" FMs, multi-objective evolutionary algorithms (MOEAs) have been widely used with the aim of generating models characterized by good trade-offs between accuracy and interpretability [7, 27]. Independently of the approach used to generate the DB and the RB of the fuzzy rule-based systems, the computation of the accuracy of each individual generated in the evolutionary process requires the scan of the overall training set. When the size of the dataset is very large, such as in the context of Big Data, the application of MOEA-based approaches to the FMs generation is very critical. Thus, the natural way for managing very large datasets would be to adopt a solution to speed up the computation: this can be done by exploiting a distributed implementation on a computer cluster.

Although some evolutionary-based methods for learning FMs for Big Data have been recently proposed [28, 44], in 2017, we have introduced the first distributed implementation of an MOEA to learn concurrently the RB and DB of FRBCs, by maximizing accuracy and minimizing complexity [32]. We have named our algorithm as DPAES-RCS: it is an Apache Spark distributed implementation of the PAES-RCS approach discussed in [9, 10]. PAES-RCS learns the RB through a rule and condition selection strategy, which selects a reduced number of rules from a heuristically generated set of candidate rules and a reduced number of conditions, for each selected rule, during the evolutionary process. Moreover, the parameters of the fuzzy sets are learnt concurrently with the RB. PAES-RCS has proven to be very efficient in obtaining satisfactory approximations of the Pareto front using a limited number of iterations [9].

In this paper, we propose an extension of DPAES-RCS that includes two main novel aspects. First, we generate the initial set of rules using the distributed FDT learning approach introduced in [53] rather than the distributed version of the C4.5 algorithm [21]. We highlight that we adopt the same learning algorithm proposed in [53], but we do not employ fuzzy partitions generated by the distributed fuzzy discretizer (proposed in [53] as well) and leaves labelled with different classes. Similar to [32], once the FDT has been generated, we extract the rules by surfing the tree from the root to each leaf. Second, we introduce a strategy for learning the most suitable number of fuzzy sets (*granularity learning*), for each linguistic variable, concurrently to the learning of the RB and the parameters of the fuzzy sets. To this aim, we adopt the *virtual partition* method introduced in [5] and recently used in [7] in the context of MOEA-based fuzzy models. We experiment the proposed extension of DPAES-RCS, named DPAES-FDT-GL, on 8 benchmark datasets for Big Data classification. We compare DPAES-FDT-GL with DPAES-RCS and with a simplified version of DPAES-FDT-GL, named DPAES-FDT, which exploits the FDT for generating the initial rule set, but does not employ the granularity learning during the evolutionary process. This last comparison is performed to evaluate whether both the extensions of DPAES-RCS included in DPAES-FDT-GL produce valuable effects.

We present that the accuracies achieved by the three approaches are statistically equivalent, while the complexity of the FRBCs generated by DPAES-FDT-GL and DPAES-FDT are much lower than the one of the FRBCs generated by DPAES-RCS. Thus, we conclude that DPAES-FDT-GL and DPAES-FDT are definitely able to generate more interpretable models than DPAES-RCS. However, even though DPAES-FDT-GL and DPAES-FDT are statistically equivalent in terms of complexity, results show that in most of the cases DPAES-FDT-GL generates the most compact solutions, characterized by the lowest number of rules, conditions, and fuzzy sets.

The paper is organized as follows. In Section 2 some background concepts are introduced. Section 3 describes the overall approach. We report the results of our experimental analysis in Section 4 and draw some final conclusion in Section 5.

2 Preliminaries

The term "classification" refers to the action of assigning a class C_m, out of a given set $C = \{C_1, \dots, C_K\}$ of K classes, to an unlabeled instance. A generic instance is described by a set $\mathbf{X} = \{X_1, \dots, X_F\}$ of attributes with cardinality F. Each attribute can be either *categorical* or *numerical.* In case of categorical attributes, X_f takes values out of a set $L_f = \{L_{f,1}, \dots, L_{f,T_f}\}$ of T_f distinct values.

For numerical attributes, the universe U_f of X_f can always be considered as a bounded interval in $\mathbb{R}$. With the aim of defining fuzzy rules, a fuzzy partition is defined on each of these intervals. Referring to a generic numeric attribute X_f, let $P_f = \{A_{f,1}, \dots, A_{f,T_f}\}$ be a partition over the relative universe U_f, where T_f is the number of fuzzy sets in the partition. A label $L_{f,j}$ is then assigned to each fuzzy set $A_{f,j}$, thus letting us work with linguistic variables and deal with both categorical and numerical attributes in a homogeneous way.

In this paper, we adopt triangular fuzzy sets and, therefore, each fuzzy set $A_{f,j}$ is identified by the tuples $(a_{f,j}, b_{f,j}, c_{f,j})$, where $a_{f,j}$ and $c_{f,j}$ correspond to the left and right extremes of the support, respectively, and $b_{f,j}$ to the core. Since we use strong partitions, $a_{f,1} = b_{f,1}, b_{f,T_f} = c_{f,Tf}$ and, for $j = 2, ..., T_f - 1$, $b_{f,j} = c_{f,j-1}$ and $b_{f,j} = a_{f,j+1}$.

The number T_f of fuzzy sets to be used in the partition for the attribute X_f can be regarded as a measure of the *granularity* used for the definition of linguistic variables over U_f. We can use the notation $P_f(T_f)$ to emphasize the granularity level for P_f, because clearly T_f has a direct impact on the accuracy and interpretability of the derived classification models.

2.1 Classification by Means of Fuzzy Rules

In FRBCs, the output value for an unlabelled instance is inferred from the fuzzy rules that compose the RB. In the present work, we assume the following structure for the generic m-th rule R_m in RB:

$$\begin{aligned} R_m : &\textbf{ if } X_1 \textbf{ is } L_{1,j_{m,1}} \textbf{ and } \dots \textbf{ and } X_F \textbf{ is } L_{F,j_{m,F}} \\ &\textbf{ then } Y \textbf{ is } C_{k_m} \end{aligned} \tag{1}$$

where Y is the FRBC output, whose value in the consequent of rule R_m is C_{k_m}, and $j_{m,f} \in [1, T_f]$ identifies the index of the label that has been selected for X_f in rule R_m, i.e. $L_{f,j_{m,f}}$. Depending on the nature of X_f (either numerical or categorical), such a label may refer to either a fuzzy set in partition P_f or a categorical value. In general, it may happen that in one rule the value assumed by an attribute provides no indications in choosing the outcome. This situation can be plainly dealt with by introducing an additional fictitious label $L_{f,0}$ for each attribute X_f, and using it to express that X_f does not contribute to the classification. Formally, this "dummy" label corresponds to a set with unitary membership across the whole attribute universe: thus, $L_{f,0}$ lets us keep the generic structure of (1) also for rules where the outcome actually depends only on a subset of the attributes.

Let $TR = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ be the training set that contains N instances. In this notation, ($\mathbf{x}_n$, y_n) indicates the n-th input-output pair, where $\mathbf{x}_n$ is the input vector with F values (each, either numerical or categorical, for the relative attribute), and y_n is the classification label.

The *matching degree* of the rule R_m with the input $\mathbf{x}_n$ represents the strength of activation of the rule. It is calculated according to the following equation:

$$w_m(\mathbf{x}_n) = \prod_{f=1}^{F} \mu_{L_{f,j_{m,f}}}(x_{n,f}) \tag{2}$$

where $\mu_{L_{f,j_{m,f}}}(x_{n,f})$ is, in case of numerical attributes, the membership value of $x_{n,f}$ to the fuzzy set $A_{f,j_{m,f}}$ represented by label $L_{f,j_{m,f}}$ and, in case of categorical attributes, is either 0 or 1.

As discussed in [33], the complexity of a rule base can be measured in different ways, but a simple yet effective index is *TRL*. In the approach proposed in this paper, *TRL* is used to quantify the model complexity (and, indirectly, its interpretability) and it is taken as one of the objectives for the evolutionary algorithm that shapes the final solutions.

Finally, we adopt the "maximum matching method" as *reasoning method*: the class of an unlabelled instance is determined by the consequent of the rule with the maximum matching degree for such an instance. In case of tie, among the equally-matching rules, the first one is chosen. If no rule is fired, the instance is classified with the most frequent class.

3 The Proposed Approach

The overall proposed approach, named DPAES-FDT-GL, is structured according to the scheme reported in Fig. 1. In the first place, it is necessary to obtain a very good *candidate rule base* as the starting point for the successive optimization process, aimed at producing a set of final FRBCs with different trade-offs between accuracy and interpretability. We denote the first phase as *Candidate RB Generation*, and the second one as *Multi-Objective Evolutionary Learning*. It is important to underline that dealing with Big Data asks for particularly efficient algorithms, and that consistent speed-ups can be achieved by adopting distributed computing solutions. For this reason, along all the successive steps in the proposed approach, distributed algorithms have been employed whenever possible using the Apache Spark framework [59], which is able to *implicitly* deal with data distribution by means of a predefined container type (known as RDD, Resilient Distributed Dataset).

For the Candidate RB Generation, differently from the previous solution discussed in [32], we adopt a distributed FDT learning approach. Thus, the candidate RB is directly obtained from the learned FDT. The rationale for using an FDT learning algorithm instead of C4.5 is based on the intuition that learning an initial candidate fuzzy RB by means of a procedure, which is completely managed by using fuzzy sets, can produce a fuzzy model more appropriate to undergo the subsequent fuzzy manipulations; moreover, in practical contexts the RBs generated by traversing all the paths from the root down to each leaf of an FDT have proven to

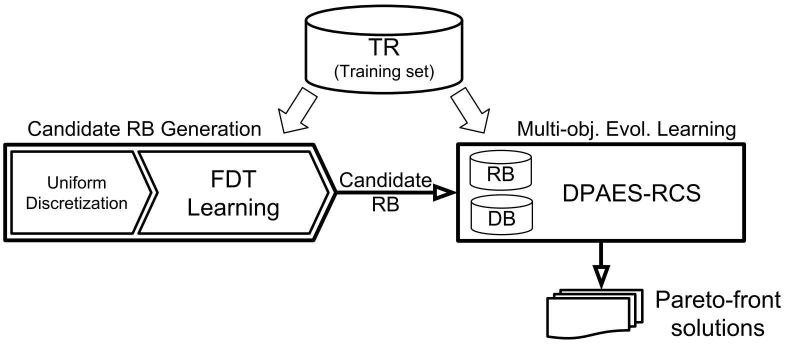

Fig. 1 General scheme of the proposed approach. Upon the construction of an initial candidate RB by means of a distributed FDT learning algorithm applied to uniformly partitioned attributes, a multi-objective evolutionary algorithm finds optimal FRBCs with different tradeoffs between accuracy and complexity.

be compact and interpretable [50]. As a preliminary step, a uniform discretization is performed. The FDT learning is carried out by means of a distributed algorithm [53].

The algorithm chosen for the Multi-Objective Evolutionary Learning (MOEL) is designed according to the well-known Pareto Archive Evolutionary Strategy (PAES) [19], and in particular it is based on DPAES-RCS [32], which is a distributed version over Apache Spark of the PAES-RCS algorithm [9]. Such a choice is motivated by its fast convergence rate, which lets us reduce the number of iterations to get to a set of satisfying solutions. This is a crucial point with Big Data, because fitness evaluations over very extensive datasets represent the most time-consuming task. An important characteristic of this phase is that the granularity of the attribute partitions is directly taken into account, and learned throughout the evolutionary process.

The following subsections give a description of the algorithms used in the two phases of the proposed approach; further details can be found in [32] and [53].

3.1 The Distributed Candidate RB Generation

The Candidate RB Generation phase used in this work exploits a distributed FDT learning approach suitable for dealing with Big Data, recently proposed by Segatori et al. [53]. The multi-way version of the FDT has been chosen (instead of the binary one) because it makes particularly simple and efficient the subsequent rule extraction to build the candidate RB: rules are obtained in the form described in (1) through a tree traversal, deriving one rule for each possible path from the root down to a leaf.

Differently from [53], which employs fuzzy partitions generated by a distributed fuzzy discretizer (proposed in [53] as well), and leaves labelled with different classes, the tree learning used in this paper relies on a preliminary *uniform* discretization for all the numerical features, with T_{max} evenly-spaced triangular fuzzy sets that make up strong partitions. This operation requires nothing more than knowing the lowest and highest values in TR for each numerical feature (that is, the endpoints for each universe U_f), and this can be easily accomplished in a distributed fashion.

Each FDT node corresponds to a subset of TR, and the root corresponds to the whole TR. The FDT construction starts from the root and follows a top-down approach; unless a termination condition is not satisfied, a newly-generated node gives rise to T_{max} child nodes according to the fuzzy partition of the attribute chosen for that specific splitting. In this procedure, an attribute can be considered only once in the same path from the root to a leaf. The attribute that drives the splitting is selected as the one that yields the best *fuzzy information gain*, which will be defined below. The termination conditions are the following:

1. all the instances in the node belong to the same class;
2. the number of instances in the node is lower than a fixed threshold λ;
3. the tree has reached a maximum fixed depth β;
4. the value of the fuzzy information gain is lower than a fixed threshold ε. In our experiments, we set $\varepsilon = 10^{-6}$.

More formally, given a parent node PN, let CN_j indicate the generic j-th child node, $j = 1, \ldots, T_{max}$. The subset of TR in CN_j contains only the instances belonging to the support of the fuzzy set $A_{f,j}$. Let S_f be the set of instances in the parent node, and $S_{f,j}$ be the set of instances for CN_j, i.e. the support of $A_{f,j}$. Each node CN_j is characterized by a fuzzy set G_j, whose cardinality is defined as

$$|G_j| = \sum_{i=1}^{N_j} \mu_{G_j}(\mathbf{x}_i) = \sum_{i=1}^{N_j} TN(\mu_{A_{f,j}}(x_{f,i}), \mu_G(\mathbf{x}_i)) \tag{3}$$

where N_j is the number of instances in set $S_{f,j}$, $\mu_G(\mathbf{x}_i)$ is the membership degree of instance $\mathbf{x}_i$ to parent node PN (for the root node, $\mu_G(\mathbf{x}_i) = 1$), and the operator TN is a T-norm.

The fuzzy information gain $FGain$ used for selecting the splitting attribute is computed, for a generic attribute X_f with partition P_f, as

$$FGain(P_f; I_G) = FEnt(G) - WFEnt(P_f; I_G) \tag{4}$$

where I_G is the support of fuzzy set G. The Fuzzy Entropy $FEnt(G)$ is defined as

$$FEnt(B_{f,j}) = \sum_{m=1}^{M} -\frac{|B_{f,j,C_m}|}{|B_{f,j}|} log_2(\frac{|B_{f,j,C_m}|}{|B_{f,j}|}) \tag{5}$$

where fuzzy cardinality $|B_{f,j,C_m}|$ is computed on the set of instances in $S_{f,j}$ with class label C_m. The weighted fuzzy entropy $WFEnt(P_{I_f}, I_f)$ of partition P_{I_f} is defined as

$$WFEnt(P_{I_f}; I_f) = \sum_{j=1}^{K_{P_{I_f}}} \frac{|B_{f,j}|}{|S_f|} FEnt(B_{f,j}) \tag{6}$$

where $|B_{f,j}|$ is the fuzzy cardinality of fuzzy set $B_{f,j}$, $|S_f|$ is the cardinality of set S_f and $FEnt(B_{f,j})$ is the fuzzy entropy of $B_{f,j}$.

In case of categorical attributes, we split the parent node into a number of child nodes CN_j equal to the number of possible values for the attribute. Each node CN_j is characterized by a fuzzy set G_j, whose cardinality is

$$|G_j| = \sum_{i=1}^{N_j} \mu_{G_j}(\mathbf{x}_i) = \sum_{i=1}^{N_j} TN(1, \mu_G(\mathbf{x}_i)) \tag{7}$$

Fig. 2 summarizes the distributed implementation of the candidate rule generation phase. We have highlighted the distribution of the operations across the

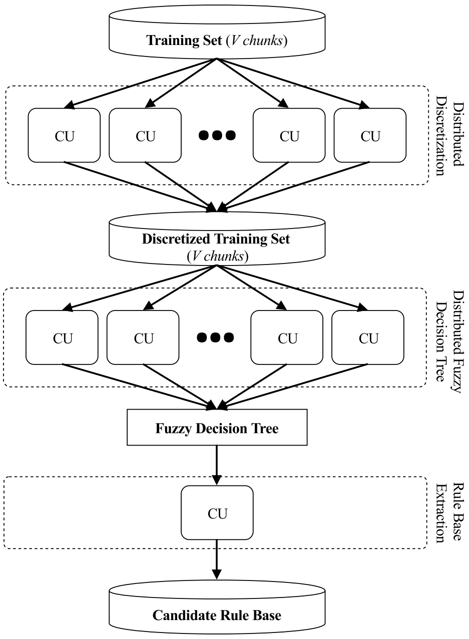

Fig. 2 Outline of the distributed candidate rule generation phase.

cluster of Computing Units (CUs). The adopted distribution strategy is aimed at reducing as much as possible the scans over TR, which is the very bottleneck in the overall computation. To this aim, the computation of the best split for each node is spread across the CUs. The FDT learning consists in the iterative execution of a MapReduce step: the mapping phase takes care of computing the figures (over V chunks for TR) to decide *how* to split, and the reduce phase is in charge of completing (if it is the case) the node splitting. The nodes to be possibly split are kept in a list, where in each iteration at most $MaxY$ elements at a time are extracted to be processed in a MapReduce step.

It is worth underlining that the chosen distributed FDT has a very good CU utilization, with extremely satisfactory values for the speed-up, thus yielding good scalability figures with respect to the number of CUs [53]. The most critical aspect relates to the system requirements for the used cluster of computers. In particular, the maximum number of nodes $MaxY$ that can be processed in parallel depends on the amount of RAM available on the cluster. Of course, more memory resources can be provided so to achieve the required parallelism [53].

3.2 The Distributed Multi-Objective Evolutionary Learning

3.2.1 The Evolutionary Process

The multi-objective evolutionary learning phase aims to produce solutions that maximize accuracy and minimize TRL, the index chosen to express the complexity of the learned FRBCs. This is obtained through a *distributed* multi-objective evolutionary algorithm that takes in input the candidate RB produced in the previous phase. The overall structure of the learning stage is based on PAES-RCS for classification models, introduced in [9] and then extended to the distribued framework in [32]. As multi-objective evolutionary algorithm, we use (2+2)M-PAES, introduced in [18] and also successfully employed in our previous works [4, 6, 7]. (2+2)M-PAES is a modified version of the well-known (2+2)PAES [40] and is a steady-state evolutionary algorithm that uses two current solutions and stores the non-dominated solutions in an archive. Unlike the classical (2+2)PAES, which maintains the current solutions until they are not replaced by solutions with particular characteristics, we randomly extract, at each iteration, the current solutions. Unlike the PAES-RCS adopted in [9, 32], which considers uniform fuzzy partitions with a pre-fixed number of fuzzy sets, the granularity of each partition here is learned as well. Thus, the chromosome coding has to accommodate this additional requirement.

The evolutionary process operates over three different aspects:

1. selection of a subset of rules out of the initial set of candidate rules, and contextual activation/deactivation of the relative conditions,
2. modification of the fuzzy partitions by properly moving the cores of the composing triangular fuzzy sets, and
3. selection of the granularity level, i.e. the number T_f of partitions (or *fuzzy sets*), in the range $[T_{min}, T_{max}]$.

We recall that the *initial* set of candidate rules is obtained by considering uniform strong fuzzy partitions for each numeric variable X_f, each containing T_{max} fuzzy sets. Also the learning of the RBs, using the evolutionary rule and condition selection procedure, and the optimization of the parameters of the fuzzy sets are performed considering such partitions, computed at the very beginning of the first phase. Indeed, we deal with *virtual RBs* [4] and *virtual partitions* [5]. The actual granularity is used only in the computation of the objectives. In practice, although during the evolutionary process we generate RBs, denoted as virtual RBs, and tune the fuzzy set parameters by using such virtual partitions, each time we need to evaluate the fitness, the evaluation is performed on the *actual* number of fuzzy sets used to partition the single variables. This process requires the definition of proper mapping strategies, both for the RB and for the fuzzy set parameters. Thus, the execution of crossover and mutation operators is not affected by the actual number of fuzzy sets.

3.2.2 RB Mapping Strategy

To map the virtual RB defined on partitions with T_{max} fuzzy sets onto a concrete RB defined on variables partitioned with T_f fuzzy sets, we adopt the simple procedure proposed in [4, 5]. Let us consider the following general proposition in a rule of the virtual RB: X_f **is** $\widehat{A}_{f,h}$, $h \in [1, T_{max}]$. It will be mapped to X_f **is** $\widetilde{A}_{f,s}$, with $s \in [1, T_f]$, where $\widetilde{A}_{f,s}$ is the fuzzy set *most similar* to $\widehat{A}_{f,h}$ out of the T_f fuzzy sets $\widehat{A}_{f,h}$ defined on X_f. In dealing with triangular fuzzy sets, defined as discussed in Section 2, a trivial yet effective similarity measure is the distance between the centers of the cores of the two fuzzy sets. If two fuzzy sets exist in the partition with centers of the cores at the same distance from the center of the core of $\widehat{A}_{f,h}$, we operate a random choice between them.

It can be noted that, at a certain granularity level, it is possible that distinct fuzzy sets defined on the partitions of the virtual RB do map onto the same fuzzy set on the partitions used in the concrete RB. Thus, different rules of the virtual RB may correspond to the same rule in the concrete RB. For this reason, duplicates in the concrete RB are searched and possibly removed. Of course, operating at a different granularity level with the same virtual RB, a different situation may arise also with respect to the presence of duplicates for concrete rules. Thus, the concept of virtual RB allows us to explore the search space and, at the same time, exploit the (sub)optimal solutions found during the evolutionary process.

3.2.3 Fuzzy Set Parameter Mapping Strategy

As regards the fuzzy set parameter tuning, we approach the problem by using a piecewise, non-decreasing linear transformation [5]. We start from an initial partition of the input variables, and tune the parameters of the fuzzy sets that compose the partition by applying such a transformation. Let $\widetilde{P}_f = \left\{\widetilde{A}_{f,1}, \ldots, \widetilde{A}_{f,T_f}\right\}$ and $P_f = \left\{A_{f,1}, \ldots, A_{f,T_f}\right\}$ be the initial and the transformed partitions, respectively. In the following, we assume that the two partitions have the same universe (i.e. $\widetilde{U}_f \equiv U_f$), considering also each variable normalized in the interval $[0, 1]$.

Let $t(x_f) : U_f \rightarrow \widetilde{U}_f$ be the piecewise linear transformation. We have that $A_{f,j}(x_f) = \widetilde{A}_{f,j}\left(t\left(x_f\right)\right) = \widetilde{A}_{f,j}\left(\widetilde{x}_f\right)$, where $\widetilde{A}_{f,j}$ and $A_{f,j}$ are two generic fuzzy sets from the initial and transformed partitions, respectively. We define the piecewise linear transformation by considering as "representative" for each fuzzy set the corresponding center of the core. The sequence of representatives indicates the change of slopes of the piecewise linear transformation $t(x_f)$ for each variable X_f. Let $\widetilde{b}_{f,1}, \ldots, \widetilde{b}_{f,T_f}$ and $b_{f,1}, \ldots, b_{f,T_f}$ be the representatives of $\widetilde{A}_{f,1}, \ldots, \widetilde{A}_{f,T_f}$ and $A_{f,1}, \ldots, A_{f,T_f}$, respectively. In each interval $b_{f,j-1} \leq x_f \leq b_{f,j}, j = 1 \ldots T_f$, the transformation $t(x_f)$ is defined as:

$$t(x_f) = \frac{\widetilde{b}_{f,j} - \widetilde{b}_{f,j-1}}{b_{f,j} - b_{f,j-1}} \cdot (x_f - b_{f,j-1}) + \widetilde{b}_{f,j-1} \tag{8}$$

Given $t(x_f)$, it can be used for the transformation of all the parameters that define the fuzzy sets.

Fig. 3 shows an example of the case where $t(x_f)$, as per Eq.(8), is defined assuming a uniform initial partition and a maximum granularity $T_{max} = 7$. Clearly, $b_{f,1}$ and b_{f,T_f} coincide with the extremes of the universe U_f of X_f, thus $t(x_f)$ depends on $T_f - 2$ parameters, that is $t(x_f) = t\left((x_f; b_{f,2}, \ldots, b_{f,T_f-1})\right)$ [5]. Once the values $b_{f,2}, \ldots, b_{f,T_f-1}$ are given, the partition P_f can be obtained by transforming the three points $(\widetilde{a}_{f,j}, \widetilde{b}_{f,j}, \widetilde{c}_{f,j})$ that describe the generic triangular fuzzy set $\widetilde{A}_{f,j}$ into $(a_{f,j}, b_{f,j}, c_{f,j})$ by applying $t^{-1}(\widetilde{x}_f)$.

It is worth noticing that, during the learning of the granularity, the transformation is applied only to the parameters that describe a fuzzy set, thus obtaining again triangular fuzzy sets. Fig. 4 shows an example of this transformation for granularity $T_f = 5$ by using the piecewise linear transformation in Fig. 3, which is defined for $T_{max} = 7$.

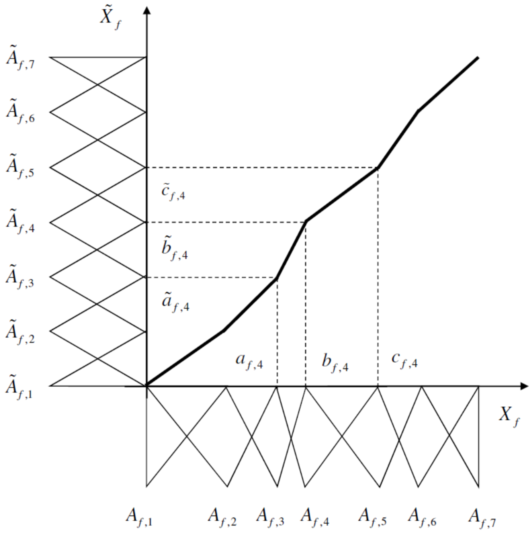

Fig. 3 An example of piecewise linear transformation.

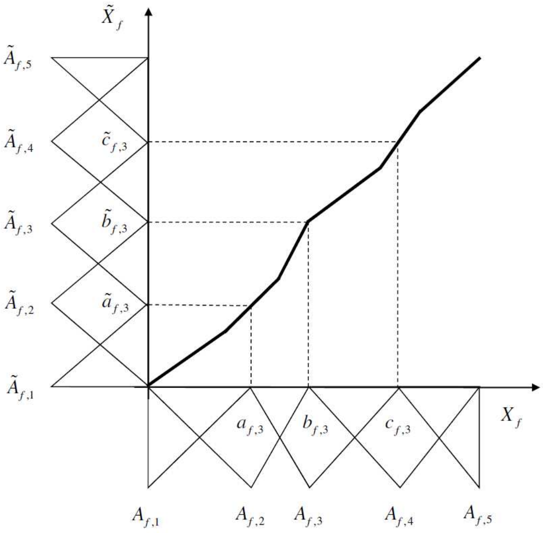

Fig. 4 Use of the transformation $t(x_f)$ of Fig. 3 on partitions with granularity $T_f = 5$ different from $T_{max} = 7$.

3.2.4 Objective Functions, Chromosome Coding, and Mating Operators

In the designed MOEL scheme, each chromosome is associated with a bi-dimensional *objective vector*. The first element accounts for the model complexity in terms of *TRL* for the relative actual RB. The second element assesses the model accuracy through its classification rate, as computed over the training set.

Within the evolutionary procedure, the chromosome C is composed of three different portions (C_R, C_G, C_T): C_R defines the virtual RB, C_G the number of fuzzy sets, and C_T the virtual partitions.

Let J_{FDT} be the initial virtual RB obtained in the first phase, and M_{FDT} the corresponding number of rules. We underline that the initial RBs are generated considering T_{max} fuzzy sets for each partition. As we are interested in getting compact RBs, we constrain the virtual RB to contain no more than M_{max} rules.

The C_R part is a vector of M_{max} pairs $\mathbf{p}_m = (k_m, \mathbf{v}_m)$, where $k_m \in [0, M_{FDT}]$ identifies the index of the selected rule in J_{FDT}, and $\mathbf{v}_m = [v_{m,1}, \ldots, v_{m,F}]$ is a "mask" boolean vector whose generic element $v_{m,f}$ indicates, for attribute X_f, whether to consider or not the relative condition in the rule (if not, it becomes a "don't care" condition). As we want to be able to generate RBs with a number of rules lower than M_{max}, k_m is set to 0 if the m-th rule must be excluded from the RB.

C_G is a vector that specifies the number of fuzzy sets to be used for each attribute. Thus, its f-th element contains the number $T_f \in [2, T_{max}]$ of fuzzy sets to be used in the actual partition $P_f(T_f)$. T_{max} is an input parameter for the algorithm, and such a value applies to all the variables. As discussed in Section 3.2.2, the values contained in C_G are used to generate the concrete RB from the virtual RB coded in C_R.

C_T is aimed at describing the placement of the T_{max} distinct fuzzy sets within each strong fuzzy partition for all the F attributes; thus, it is a vector of F vectors, each containing $T_{max} - 2$ real numbers. The f-th vector $[b_{f,2}, \ldots, b_{f,T_{max}-1}]$ indicates the positions of the cores of the triangular fuzzy sets: it also contains the information to define the shape of the piecewise linear transformation $t(x_f)$ (and consequently t^{-1}) used to determine the position of T_f fuzzy sets, if $T_f < T_{max}$ holds. To make sure that $b_{f,i} < b_{f,i+1}$, $\forall i \in [2, T_{max} - 1]$, and to avoid an excessive departure of the cores with respect to the uniform partition, the value for the generic $b_{f,j}$ is restricted to vary in the interval

$$\left[\widetilde{b}_{f,j} - \frac{\widetilde{b}_{f,j} - \widetilde{b}_{f,j-1}}{2}, \widetilde{b}_{f,j} + \frac{\widetilde{b}_{f,j+1} - \widetilde{b}_{f,j}}{2}\right], \forall j \in [2, T_{max} - 1]\,. \tag{9}$$

For the generation of the offspring populations, the MOEL makes use of both crossover and mutation operations. We apply independently the two-point crossover to C_R, the one-point crossover to C_G and the BLX-α-crossover, with $\alpha = 0.5$, to C_T. As regards C_R, the crossover points are always placed between two rules. In the case of C_G, the crossover point is a random position in $[1, F]$.

The algorithm includes two mutation operators for C_R, one for C_G, and another for C_T. As regards C_R, both operators start by randomly choosing a rule (actually, a pair $\mathbf{p}_m$) in the chromosome. The first operator replaces the rule in $\mathbf{p}_m$ with another rule randomly chosen out of the candidate rule base. The second operator modifies the rule in $\mathbf{p}_m$ by going through each position $\mathbf{v}_{m,f}$ of the condition mask, and performing its complement with a probability equal to P_{cond} ($P_{cond} = \frac{2}{F}$ in the experiments).

The mutation for C_G attempts to modify the granularity for one single variable X_f: it consists of randomly choosing a gene $f \in [1, F]$ and randomly either incrementing or decrementing it by one. If the new value is out of the range $[2, T_{max}]$, no modification is actually performed.

The mutation for C_T operates on $b_{f,j}$, by first randomly choosing f in $[1, F]$ and j in $[2, T_{max} - 1]$: the new value for it is randomly selected in the allowed interval (Eq. 9). The described mating operators have experimentally shown a good balance between exploration and exploitation, thus being suitable for driving the evolutionary algorithm towards good approximations of the Pareto fronts.

3.2.5 MOEL Distributed Implementation

The characteristics of the possible strategies for parallel/distributed MOEAs have been extensively studied before the widespread use of cloud computing facilities [54]. Anyway, recently the opportunity to exploit cloud resources in a simple way by means of efficient frameworks like Apache Spark has made more attractive some solutions than others. Thus, the "master-slave" paradigm [54] inherent in typical Spark programs has been chosen in particular for its ability to deal with big datasets, providing good scalability with respect to the size of the training set. Other paradigms, like the "islands" and "diffusion" ones, are much more suited with other distributed computing frameworks [19, 54], and in these cases the obtained accuracy may be affected by the number of used CUs, while we would make the results independent of the underlying platform.

The distributed implementation of the MOEL phase for the proposed approach is described in the schematic view of Fig. 5: here, it is explicitly shown the workload distribution across a cluster of CUs. It can be noted that distributed computations can be used both in the initialization of the archive required by the genetic algorithm (upper part of the figure), and in the evolutionary procedure itself (lower part).

The overall MOEL algorithm is driven by the master task: it is in charge of the main control flow and, at each single iteration, of the *TRL* part of the fitness computation (which, given the limited size of the rule base, is really effortless). Instead, the evaluation of the accuracy asks for a scan of the whole TR, which is typically very large. Thus, it is advantageous to split TR in V chunks to be separately scanned by slave tasks on the cluster CUs.

This way to exploit the CUs in the cluster is clearly indicated in Fig. 5.

This scheme can be easily accommodated by developing a single procedure to be executed just for this purpose by all the slave CUs, taking as input the two solutions to be evaluated and returning, for each of them, the number of successful classifications over the target chunk of *TR*. It is up to the driver task to sum up all the contributions to the overall accuracy value.

Notably, it has been shown that the scalability of the adopted MOEL, with respect to the used CUs, is almost linear [32]. This means that, whenever needed, additional CUs can be used so to effectively deliver reduced runtimes. Finally, we can underline that the effort required by the accuracy evaluation depends on the complexity of the rule base. As such a complexity typically becomes smaller and smaller as the population evolves, the execution time for each iteration significantly decreases as the algorithm proceeds towards its completion.

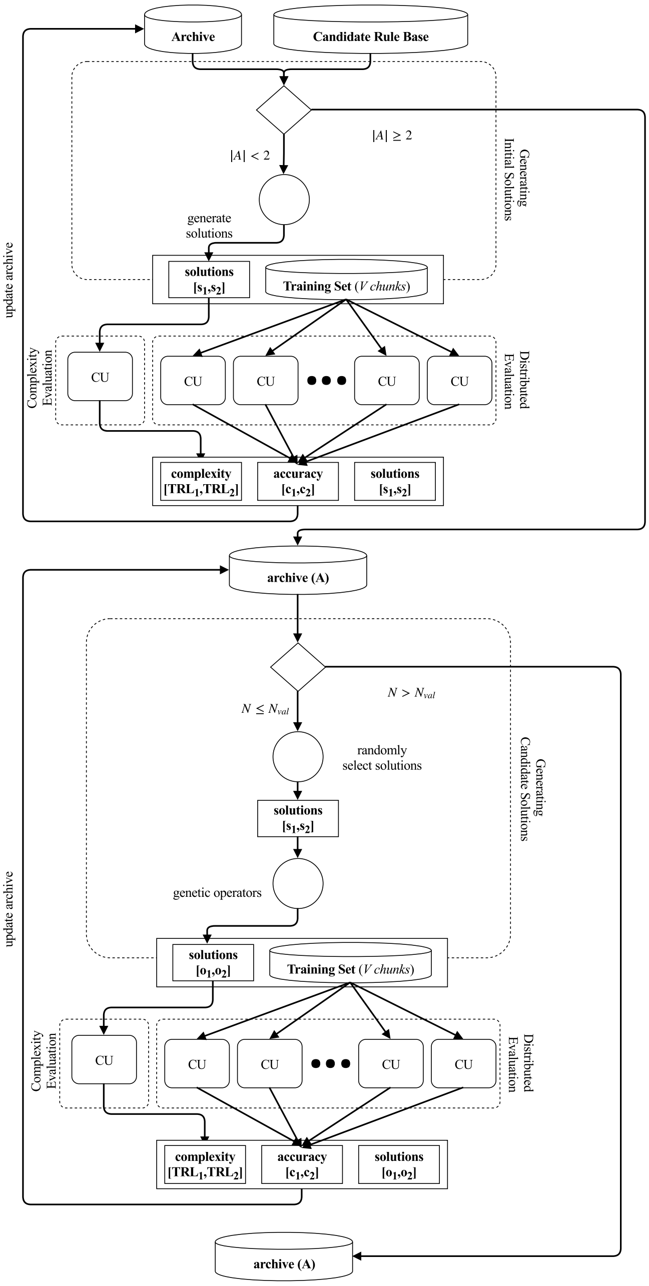

Fig. 5 Outline of the distributed computation approach adopted in the evolutionary procedure.

4 Experimental Results

In this section we show the results of an experimental study for the evaluation of DPAES-FDT-GL. In the following we take two main aspects into consideration: (1) evaluation of the solutions provided by the algorithm in terms of classification accuracy, complexity, and interpretability; (2) comparison of the algorithm with the original DPAES-RCS. Moreover, in order to disentangle the contribution of the FDT from that of the granularity learning, we performed a comparison with DPAES-FDT, which adopts the FDT for generating the initial rule set, but no granularity learning during the evolutionary process.

The eight datasets used in the experiments are listed in Table 1, along with their number of instances, attributes (both numerical and categorical), classes, and their size. The datasets have been collected from the UCI[1] and the LIBSVM[2] repositories.

Table 1 Datasets used in the experiments: n and c denote numerical and categorical, respectively.

Name	# Instances	# Attributes	# Classes	Size
Covertype 2 (COV_2)	581 012	54 (n:10, c:44)	2	75.2 MB
Covertype 7 (COV_7)	581 012	54 (n:10, c:44)	7	75.2 MB
eCO (ECO)	4 178 504	16 (n:16)	10	534 MB
eME (EME)	4 178 504	16 (n:16)	10	535.2 MB
Higgs (HIG)	11 000 000	28 (n:28)	2	8.04 GB
Kddcup 2 (KDD_2)	4 856 151	41 (n:26, c:15)	2	476 MB
PokerHand (POK)	1 025 010	10 (c:10)	10	24.5 MB
Susy (SUS)	5 000 000	18 (n:18)	2	2.40 GB

For each dataset, we performed a five-fold cross validation. The experiments have been carried out using Apache Spark 2.2.0 over a small computer cluster; we used up to 7 machines, one master node and up to 6 workers. Both the *master* and the *workers* are supplied with 4 vCPU, 8GB of RAM, and 160 GB Hard Drive. All the machines run Ubuntu 14.04. The datasets are stored on the Hadoop Distributed File System. In all the experiments we used the standalone cluster manager provided by Apache Spark.

The parametrization used for DPAES-FDT-GL and DPAES-FDT is reported in Table 2, and the values have been devised starting from the experience with DPAES-RCS [32]. Regarding the number of evaluations, for most of the datasets, we experimentally verified that the evolutionary optimization process has a similar behavior as the one discussed in [9], where we showed that 50,000 fitness evaluations allow obtaining Pareto fronts statistically equivalent to the ones achieved after 1 million evaluations. For the sake of brevity, we do not report this analysis in the paper. Since each iteration of the (2+2)M-PAES requires two fitness evaluations, it follows that 50,000 fitness evaluations correspond to 25,000 iterations. Regarding granularity learning, we let the number of fuzzy sets per partition vary between $T_{min} = 3$ and $T_{max} = 7$. In fact, considering the employed strong triangu-

[1] Available at `https://archive.ics.uci.edu/ml/datasets.html`

[2] Available at `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

lar fuzzy partitioning scheme, the first and last fuzzy sets are tied to the ends of the universe, thus a meaningful learned partitioning requires at least three fuzzy sets. Furthermore, according to psychologists, to preserve interpretability, the number of linguistic terms per variable should be in the range 7±2. This is due to a limit of the human information processing capability [46]. Thus, for the sake of simplicity, we set $T_{max} = 7$. Moreover, in our first work on granularity learning, discussed in [4], we showed that using 7 or 9 as T_{max} yields similar results.

Table 2 Values of the parameters used in the experiments for DPAES-FDT-GL and DPAES-FDT.

Parameter	Description	Value
N_{val}	Total number of fitness evaluations	50000
AS	(2+2)M-PAES archive size	64
M_{max}	Maximum number of rules in a virtual RB	100
T_f	Number of fuzzy sets for each continous attribute X_f	7
P_{C_R}	Probability of applying crossover operator to C_R	0.6
P_{C_T}	Probability of applying crossover operator to C_T	0.5
P_{C_G}	Probability of applying crossover operator to C_G	0.5
P_{MRB_1}	Probability of applying first mutation operator to C_R	0.1
P_{MRB_2}	Probability of applying second mutation operator to C_R	0.7
P_{M_T}	Probability of applying mutation operator to C_T	0.6
P_{M_G}	Probability of applying mutation operator to C_G	0.2
T_{max}	Maximum number of fuzzy sets for each linguistic variable	7
T_{min}	Minimum number of fuzzy sets for each linguistic variable	3

As previously described, for each dataset the initial RB has been obtained exploiting the multi-way version of the distributed FDT algorithm [52], along with a uniform discretization with $T_{max} = 7$ linguistic values for each numeric attribute. As suggested in [52], we limited the minimum number of instances per leaf to 0.1% of the total number of instances. Moreover, we set the maximum tree depth β to 10 so to generate sufficiently complex rules, yet limiting their total number. The average number of *generated* rules, as well as the average number of selected features, are reported in Table 3.

Table 3 Values of the parameters used for the distributed FDT algorithm, and average numbers of rules and attributes in the RBs extracted from the generated FDTs.

Dataset	min # inst. per leaf	$\overline{\text{Rules}}$	$\overline{\text{Attributes}}$
COV_2	1	13 392.8	12.0
COV_7	1	18 176.2	53.0
ECO	334	6 683.0	13.0
EME	334	9 226.6	16.0
HIG	880	4 138.0	21.0
KDD_2	391	451.6	22.2
POK	80	28 561.0	5.0
SUS	400	10 770.0	18.0

4.1 Experimental Evaluation of DPAES-FDT-GL

In this section, we discuss the results of an experimental evaluation of DPAES-FDT-GL. In order to provide a thorough evaluation of the proposed algorithm, we analyzed the Pareto front approximations generated during the optimization process by means of previously proposed methods [32]. First, for each fold, we extracted and sorted the Pareto front by decreasing accuracy; then, we selected three solutions: FIRST, MEDIAN and LAST, as the most accurate, the median solution in the set and the least accurate, respectively, with respect to accuracy.

Table 4 shows, for each dataset and for each representative solution, the average values and the standard deviations of the accuracy achieved on both the training (Acc_{Tra}) and test (Acc_{Tst}) sets, of the average values and the standard deviations of the complexity (TRL) and of the number (N_{NDS}) of non-dominated solutions contained in the archive at the end of the evolutionary process.

First, in Table 4, we observe highly competitive results (a comparison with the state-of-the-art is provided in Section 4.2), while the TRL is still reasonable. Thus, we can conclude that DPAES-FDT-GL is able to generate both accurate and interpretable systems. Furthermore, by comparing the accuracies obtained on the training and test sets, we observed that little or no overtraining occurs.

In order to better characterize the interpretability of the provided solutions, in Table 5 we report M, $\widehat{F}$, and $\#F_{set}$ for the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL. Here, M is the average number of rules, $\widehat{F}$ is the average number of selected features, and $\#F_{set}$ represents the average number of fuzzy sets obtained via granularity learning for each selected numerical feature. Interestingly, both the number of selected rules and the number of features are quite low, suggesting that the learned RB is highly interpretable. Moreover, the mean number of fuzzy sets per feature is lower than 5, suggesting that the granularity learning does indeed help in producing more intepretable systems.

Finally, comparing the number of rules M with TRL we observe that the average rule length (not shown here) is typically very low, suggesting that the RBs are mostly composed by generic rules.

Table 6 shows, for each dataset, the average execution time for DPAES-FDT-RCS (in seconds) as well as its standard deviation. Execution times have been measured on a cluster with 6 slaves, equipped with 4 cores each, for a total of 24 cores. Both the total execution time and the runtime for the distributed evolutionary optimization phase (DEO) are reported. We observe that the DEO phase is the most time consuming one. The runtime is primarily affected by two factors: the number of instances in the dataset, and the TRL of the evaluated solutions.

To give an example of the results provided by DPAES-FDT-GL, we show the MEDIAN solution obtained on the first fold of SUSY[3] dataset. As reported in Table 4, the MEDIAN solution performs well both in terms of accuracy and TRL. SUSY is a binary classification problem to distinguish between a signal process that produces supersymmetric particles and a background process that does not. The data have been produced using Monte Carlo simulations, and are characterized by 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; these are high-level features derived by physicists to help discrim-

[3] The SUSY dataset is available at `https://archive.ics.uci.edu/ml/datasets/SUSY`

Table 4 Average values and standard deviations of the accuracy on the training and test sets and of TRL of the FIRST, MEDIAN and LAST solutions and average values and standard deviations of the number of non-dominated solutions generated by DPAES-FDT-GL.

	FIRST			**MEDIAN**			**LAST**			
Dataset	Acc_{Tra}	Acc_{Tst}	TRL	Acc_{Tra}	Acc_{Tst}	TRL	Acc_{Tra}	Acc_{Tst}	TRL	N_{NDS}
COV_2	75.843 ± 0.002	75.767 ± 0.003	107.4 ± 23.6	75.378 ± 0.002	75.320 ± 0.003	43.4 ± 11.8	66.153 ± 0.097	66.203 ± 0.096	12.2 ± 7.5	37.0 ± 7.5
COV_7	67.649 ± 0.012	67.618 ± 0.012	11.4 ± 3.0	67.611 ± 0.011	67.582 ± 0.012	8.8 ± 1.8	67.172 ± 0.007	67.157 ± 0.007	5.8 ± 0.8	7.0 ± 2.8
ECO	76.261 ± 0.005	76.266 ± 0.005	101.2 ± 24.9	74.069 ± 0.007	74.074 ± 0.007	34.6 ± 10.3	56.816 ± 0.089	56.801 ± 0.089	5.0 ± 0.0	37.8 ± 8.6
EME	81.225 ± 0.005	81.193 ± 0.005	136.8 ± 23.0	78.997 ± 0.007	78.981 ± 0.007	50.2 ± 26.4	62.793 ± 0.039	62.793 ± 0.039	5.8 ± 1.8	44.8 ± 6.6
HIG	65.040 ± 0.003	65.035 ± 0.004	48.4 ± 23.2	63.625 ± 0.007	63.610 ± 0.007	22.0 ± 1.7	58.718 ± 0.003	58.697 ± 0.003	6.2 ± 1.5	24.2 ± 7.0
KDD_2	99.886 ± 0.008	99.886 ± 0.010	24.6 ± 6.5	99.883 ± 0.008	99.882 ± 0.009	13.6 ± 2.5	94.423 ± 0.094	94.415 ± 0.094	5.4 ± 0.5	15.0 ± 4.8
POK	61.778 ± 0.011	61.806 ± 0.001	90.2 ± 7.1	56.061 ± 0.008	55.989 ± 0.008	37.6 ± 5.7	49.870 ± 0.009	49.822 ± 0.009	9.2 ± 3.3	55.2 ± 4.3
SUS	78.628 ± 0.004	78.608 ± 0.004	63.0 ± 17.1	78.362 ± 0.006	78.361 ± 0.006	28.2 ± 8.3	72.525 ± 0.052	72.521 ± 0.052	7.4 ± 3.8	28.0 ± 5.3

Table 5 Average values and standard deviations of the number of rules (M), the number of attributes ($\widehat{F}$), and the average number of fuzzy sets in the partition ($\#F_{set}$) for the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL. Please note that the dataset POK has no numerical feature.

	FIRST			**MEDIAN**			**LAST**		
Dataset	M	$\widehat{F}$	$\#F_{set}$	M	$\widehat{F}$	$\#F_{set}$	M	$\widehat{F}$	$\#F_{set}$
COV_2	20.8 ± 3.3	11.4 ± 0.5	4.4 ± 0.2	12.4 ± 2.4	9.4 ± 0.5	4.5 ± 0.3	7.6 ± 3.1	6.4 ± 2.7	4.0 ± 0.3
COV_7	7.0 ± 1.2	9.2 ± 2.4	3.7 ± 0.2	6.8 ± 1.1	7.2 ± 1.9	3.9 ± 0.3	5.8 ± 0.8	4.6 ± 0.9	3.8 ± 0.3
ECO	21.2 ± 3.3	11.6 ± 0.5	4.6 ± 0.2	10.2 ± 2.3	10.2 ± 0.9	4.5 ± 0.2	5.0 ± 0.0	3.6 ± 0.5	4.7 ± 0.3
EME	28.6 ± 4.7	14.2 ± 0.8	4.8 ± 0.3	14.4 ± 5.1	11.0 ± 2.3	4.7 ± 0.2	5.4 ± 0.9	2.8 ± 1.9	4.7 ± 0.3
HIG	14.0 ± 1.7	12.2 ± 2.1	3.7 ± 0.2	9.0 ± 1.7	9.4 ± 0.6	3.7 ± 0.1	6.4 ± 1.5	5.4 ± 0.6	3.7 ± 0.2
KDD_2	10.8 ± 1.8	9.8 ± 2.2	3.9 ± 0.3	7.2 ± 1.1	7.6 ± 1.7	3.9 ± 0.3	5.4 ± 0.5	4.2 ± 0.4	3.9 ± 0.3
POK	41.6 ± 3.1	5.0 ± 0.0	-	19.4 ± 3.0	5.0 ± 0.0	-	6.6 ± 1.3	4.2 ± 0.4	-
SUS	14.6 ± 2.7	12.4 ± 0.9	4.3 ± 0.2	9.0 ± 1.6	10.2 ± 1.3	4.2 ± 0.2	6.6 ± 2.5	4.8 ± 1.5	4.2 ± 0.2

Table 6 Average computation times (in seconds) and standard deviations for the distributed evolutionary optimization (DEO) phase and the overall algorithm (Tot).

	Execution Time (s)	
Datasets	DEO	Tot
COV_2	$6\,245 \pm 1\,115$	$7\,165 \pm 1\,191$
COV_7	$4\,965 \pm 718$	$5\,147 \pm 671$
ECO	$23\,895 \pm 6\,449$	$24\,836 \pm 6\,416$
EME	$27\,189 \pm 3\,060$	$28\,088 \pm 3\,047$
HIG	$53\,749 \pm 9\,780$	$54\,821 \pm 9\,805$
KDD_2	$13\,470 \pm 1\,033$	$14\,310 \pm 1\,033$
POK	$3\,935 \pm 422$	$3\,964 \pm 421$
SUS	$32\,010 \pm 6\,697$	$32\,611 \pm 6\,690$

inate between the two classes. In the following the features will be labelled as: lepton 1 p_T, lepton 1 η, lepton 1 ϕ, lepton 2 p_T, lepton 2 η, lepton 2 ϕ, missing energy magnitude, missing energy ϕ, MET_rel, Axial MET, M_R, M_TR_2, R, M_{T2}, $\sqrt{\hat{S}_R}$, M_{Δ_R}, $\Delta\Phi_{R^\beta}$ and $cos(\theta_{R+1})$. More information about the attributes can be retrieved in the original manuscript [13].

Fig. 6 shows, for each continuous attribute, both the original uniform fuzzy partition (dashed line), and the learned fuzzy partition (solid line) of the MEDIAN solution. The corresponding RB is shown in Fig. 7. Here, the labelling of the fuzzy sets depends on the granularity of the partitioning: for 3 fuzzy sets we used *low*, *medium* and *high*, while for 7 fuzzy sets we used *very_low*, *low*, *medium-low*, *medium*, *medium-high*, *high* and *very_high*. The labelling for 4, 5 and 6 fuzzy sets has been obtained by interpolating in between. It is worth noticing that the RB is composed of only 7 rules, with a maximum of 3 antecedents each.

4.2 Comparison of DPAES-FDT-GL with DPAES-FDT and DPAES-RCS

In this section, we experimentally compare the performances of DPAES-FDT-GL with DPAES-FDT and DPAES-RCS, the baseline MOEL scheme from which DPAES-FDT-GL and DPAES-FDT have been derived. We underline that in [32] it has been shown that DPAES-RCS is highly effective when compared to other state-of-the-art algorithms, such as distribute decision trees and the Chi-FRBCS-BigData.

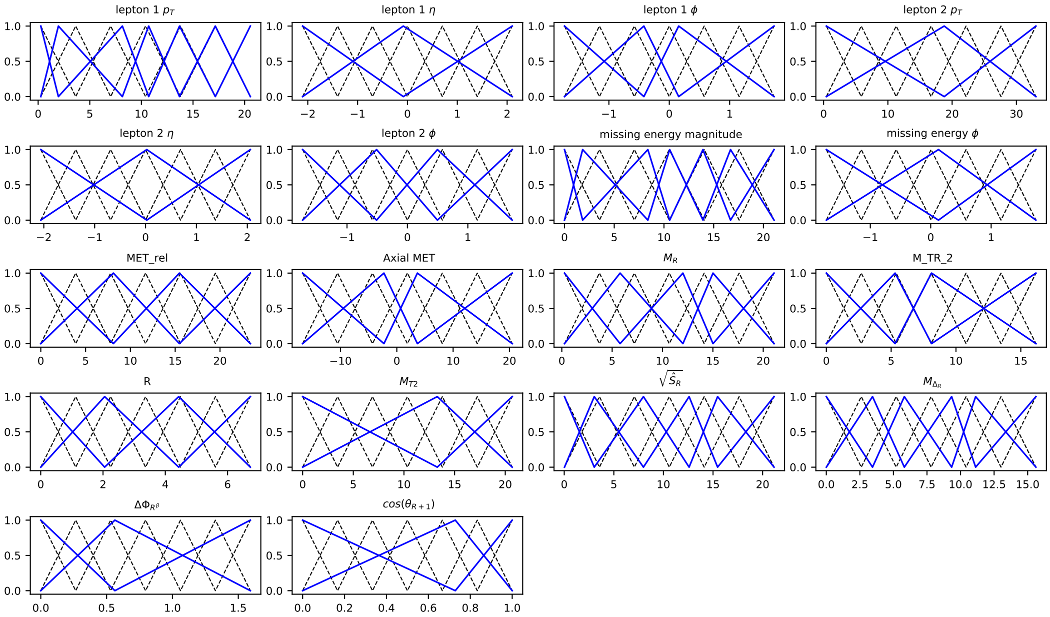

Fig. 6 Uniform fuzzy partitions (dashed line) and learned fuzzy partitions (solid line) of the attributes of the MEDIAN solution obtained at the end of the evolutionary process on the SUSY dataset.

IF missing energy magnitude IS 'very_low' AND M_R IS 'very_high' AND cos(θ_{R+1}) IS 'low' THEN Y IS TYPE_1
IF lepton 1 p_T IS 'very_low' AND missing energy magnitude IS 'very_low' AND M_{Δ_R} IS 'very_high' THEN Y IS TYPE_1
IF lepton 1 p_T IS 'low' AND lepton 1 η IS 'low' AND lepton 2 η IS 'low' THEN Y IS TYPE_2
IF Axial MET IS 'high' AND $\sqrt{\hat{S}_R}$ IS 'very_high' AND $cos(\theta_{R+1})$ IS 'low' THEN Y IS TYPE_2
IF lepton 1 p_T IS 'very_low' AND missing energy magnitude IS 'very_low' AND $cos(\theta_{R+1})$ IS 'medium' THEN Y IS TYPE_1
IF R IS 'high' AND $cos(\theta_{R+1})$ IS 'high' THEN Y IS TYPE_2
IF missing energy magnitude IS 'medium-low' THEN Y IS TYPE_2

Fig. 7 RB of the MEDIAN solution obtained on the first fold of SUSY. The RB, composed of 7 rules, and characterized by a TRL of 18, achieved a classification accuracy of $\sim$ 78.776% on the test set.

Hereafter, the reported results achieved by DPAES-RCS are taken from the tables in [32]. In Table 7 we list the average values with standard deviations of the accuracy on the training (Acc_{Tra}) and test (Acc_{Tst}) sets for the FIRST, MEDIAN, and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and DPAES-RCS. We also report M and TRL in Table 8.

We observe that the accuracy values obtained by the three algorithms are generally comparable across all the three solutions analyzed here. Moreover, the

Table 7 Average accuracies ± standard deviations achieved by the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and by DPAES-RCS.

	DPAES-FDT-GL (FIRST)		**DPAES-FDT (FIRST)**		**DPAES-RCS (FIRST)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	75.843 ± 0.002	75.767 ± 0.003	**75.999 ± 0.002**	**75.988 ± 0.001**	75.753 ± 0.004	75.732 ± 0.003
COV_7	67.649 ± 0.012	67.618 ± 0.012	68.156 ± 0.006	68.232 ± 0.007	**72.383 ± 0.003**	**72.374 ± 0.003**
ECO	76.261 ± 0.005	76.266 ± 0.005	**78.498 ± 0.005**	**78.493 ± 0.005**	77.133 ± 0.004	77.115 ± 0.004
EME	81.225 ± 0.005	81.193 ± 0.005	**82.882 ± 0.004**	**82.855 ± 0.004**	80.600 ± 0.008	80.570 ± 0.008
HIG	**65.040 ± 0.003**	**65.035 ± 0.004**	65.013 ± 0.003	65.010 ± 0.003	65.008 ± 0.012	64.998 ± 0.012
KDD99_2	99.886 ± 0.008	99.886 ± 0.010	99.866 ± 0.000	99.865 ± 0.000	**99.948 ± 0.012**	**99.947 ± 0.012**
POK	**61.778 ± 0.011**	**61.806 ± 0.001**	60.535 ± 0.010	60.504 ± 0.011	60.233 ± 0.006	60.221 ± 0.006
SUS	**78.628 ± 0.004**	**78.608 ± 0.004**	77.968 ± 0.003	77.954 ± 0.003	78.123 ± 0.001	78.110 ± 0.001
AVERAGE	75.789 ± 0.006	75.772 ± 0.005	76.115 ± 0.004	76.113 ± 0.004	76.148 ± 0.006	76.133 ± 0.006
	DPAES-FDT-GL (MEDIAN)		**DPAES-FDT (MEDIAN)**		**DPAES-RCS (MEDIAN)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	**75.378 ± 0.002**	75.320 ± 0.003	75.375 ± 0.003	**75.335 ± 0.004**	74.968 ± 0.005	74.909 ± 0.005
COV_7	67.611 ± 0.011	67.582 ± 0.012	67.925 ± 0.006	67.988 ± 0.006	**71.940 ± 0.004**	**71.924 ± 0.004**
ECO	74.069 ± 0.007	74.074 ± 0.007	**76.634 ± 0.008**	**76.635 ± 0.007**	74.995 ± 0.011	74.984 ± 0.011
EME	78.997 ± 0.007	78.981 ± 0.007	**80.982 ± 0.010**	**80.957 ± 0.010**	78.221 ± 0.010	78.201 ± 0.010
HIG	63.625 ± 0.007	63.610 ± 0.007	63.711 ± 0.005	63.710 ± 0.005	**64.389 ± 0.008**	**64.370 ± 0.008**
KDD_2	99.883 ± 0.008	99.882 ± 0.009	99.865 ± 0.000	99.864 ± 0.000	**99.933 ± 0.008**	**99.934 ± 0.008**
POK	56.061 ± 0.008	55.989 ± 0.008	55.428 ± 0.010	55.360 ± 0.010	**58.423 ± 0.008**	**58.430 ± 0.009**
SUS	**78.362 ± 0.006**	**78.361 ± 0.006**	77.729 ± 0.003	77.707 ± 0.003	77.658 ± 0.003	77.659 ± 0.003
AVERAGE	74.248 ± 0.007	74.225 ± 0.007	74.706 ± 0.006	74.694 ± 0.006	75.066 ± 0.007	75.051 ± 0.007
	DPAES-FDT-GL (LAST)		**DPAES-FDT (LAST)**		**DPAES-RCS (LAST)**	
Datasets	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}	Acc_{Tra}	Acc_{Tst}
COV_2	66.153 ± 0.097	66.203 ± 0.096	70.907 ± 0.035	70.955 ± 0.035	**72.708 ± 0.007**	**72.681 ± 0.006**
COV_7	67.172 ± 0.007	67.157 ± 0.007	**67.243 ± 0.005**	**67.299 ± 0.006**	57.921 ± 0.106	57.907 ± 0.106
ECO	56.816 ± 0.089	56.801 ± 0.089	**59.864 ± 0.032**	**59.851 ± 0.032**	56.228 ± 0.078	56.244 ± 0.078
EME	**62.793 ± 0.039**	**62.793 ± 0.039**	62.233 ± 0.043	62.230 ± 0.043	61.407 ± 0.061	61.391 ± 0.061
HIG	58.718 ± 0.003	58.697 ± 0.003	58.524 ± 0.016	58.530 ± 0.015	**59.825 ± 0.017**	**59.849 ± 0.017**
KDD_2	94.423 ± 0.094	94.415 ± 0.094	94.351 ± 0.094	94.347 ± 0.094	**98.508 ± 0.017**	**98.514 ± 0.017**
POK	**49.870 ± 0.009**	**49.822 ± 0.009**	48.313 ± 0.012	48.331 ± 0.012	48.772 ± 0.031	48.749 ± 0.032
SUS	**72.525 ± 0.052**	**72.521 ± 0.052**	71.377 ± 0.051	71.414 ± 0.051	68.131 ± 0.083	68.128 ± 0.082
AVERAGE	66.059 ± 0.049	66.051 ± 0.049	66.601 ± 0.036	66.620 ± 0.036	65.438 ± 0.050	65.433 ± 0.050

solutions generated by DPAES-FDT-GL and DPAES-FDT are always, except for the FIRST solution of COV_2 dataset, more compact than those produced by DPAES-RCS. However, it is worth noting that DPAES-FDT-GL solutions are characterized, in most cases, by a lower *TRL* and fewer rules than DPAES-FDT solutions.

To statistically assess the differences among the achieved accuracies and complexities, we generate, for each comparison algorithm and on all datasets, a distribution consisting of the average accuracy values obtained on the test set, and a distribution consisting of the average complexity values. Then, we apply non-parametric statistical tests. In particular, we perform the Friedman test to compute a ranking among the distributions, and the Iman and Davenport test to evaluate whether there exists a statistical difference among the distributions. If the Iman and Davenport p-value is lower than the level of significance α (it is assumed the standard threshold value $\alpha = 0.05$), we can reject the null hypothesis and affirm that there exist statistical differences among the multiple distributions. Otherwise, no statistical difference exists. In case of statistical difference, we apply a post-hoc procedure, namely the Holm test. This test allows detecting effective statistical

Table 8 Average M and TRL ± standard deviations achieved by the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL, DPAES-FDT, and by DPAES-RCS.

	DPAES-FDT-GL (FIRST)		**DPAES-FDT (FIRST)**		**DPAES-RCS (FIRST)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	**20.8 ± 3.3**	107.4 ± 23.6	22.4 ± 3.8	113.8 ± 36.5	33.6 ± 8.4	**74.4 ± 23.0**
COV_7	7.0 ± 1.2	**11.4 ± 3.0**	**6.6 ± 1.9**	11.4 ± 4.4	36.2 ± 7.3	145.0 ± 37.0
ECO	**21.2 ± 3.3**	**101.2 ± 24.9**	28.0 ± 3.5	136.6 ± 21.4	54.0 ± 16.5	168.4 ± 79.6
EME	**28.6 ± 4.7**	**136.8 ± 23.0**	34.2 ± 4.0	165.2 ± 35.3	58.6 ± 5.7	187.4 ± 39.8
HIG	**14.0 ± 1.7**	**48.4 ± 23.2**	18.6 ± 3.4	77.2 ± 22.3	30.2 ± 8.2	125.2 ± 40.2
KDD99_2	**10.8 ± 1.8**	24.6 ± 6.5	11.4 ± 0.5	**21.8 ± 1.3**	21.8 ± 4.1	35.4 ± 8.0
POK	41.6 ± 3.1	90.2 ± 7.1	**39.0 ± 3.0**	**83.8 ± 6.5**	50.0 ± 4.6	113.2 ± 13.3
SUS	**14.6 ± 2.7**	**63.0 ± 17.1**	18.0 ± 6.0	73.0 ± 35.6	28.0 ± 8.6	80.4 ± 33.4
AVERAGE	19.825 ± 2.725	72.875 ± 16.050	22.275 ± 3.262	85.350 ± 20.412	39.050 ± 7.925	116.175 ± 34.287

	DPAES-FDT-GL (MEDIAN)		**DPAES-FDT (MEDIAN)**		**DPAES-RCS (MEDIAN)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	12.4 ± 2.4	43.4 ± 11.8	**11.2 ± 0.8**	**37.2 ± 6.0**	21.7 ± 7.3	38.7 ± 17.3
COV_7	6.8 ± 1.0	8.8 ± 1.8	**6.2 ± 1.6**	**7.8 ± 2.5**	29.4 ± 6.8	84.2 ± 25.1
ECO	**10.2 ± 2.3**	**34.6 ± 10.3**	15.0 ± 2.3	51.0 ± 11.2	45.4 ± 17.3	117.7 ± 73.4
EME	**14.4 ± 5.1**	**50.2 ± 26.4**	15.4 ± 1.5	56.8 ± 13.2	48.1 ± 5.9	112.0 ± 27.2
HIG	**9.0 ± 1.7**	**22.0 ± 1.7**	10.6 ± 0.9	26.8 ± 6.0	25.8 ± 6.8	78.7 ± 28.6
KDD_2	**7.2 ± 1.1**	13.6 ± 2.5	7.4 ± 0.9	**13.4 ± 0.5**	13.2 ± 2.5	19.5 ± 4.3
POK	19.4 ± 3.0	37.6 ± 5.7	**17.2 ± 2.9**	**33.8 ± 6.3**	35.2 ± 6.3	68.1 ± 11.8
SUS	**9.0 ± 1.6**	28.2 ± 8.3	9.6 ± 2.8	**26.6 ± 12.3**	19.9 ± 7.7	45.6 ± 25.5
AVERAGE	11.050 ± 2.275	29.800 ± 8.562	11.575 ± 1.712	31.675 ± 7.250	29.838 ± 7.575	70.562 ± 26.650

	DPAES-FDT-GL (LAST)		**DPAES-FDT (LAST)**		**DPAES-RCS (LAST)**	
Datasets	*M*	*TRL*	*M*	*TRL*	*M*	*TRL*
COV_2	7.6 ± 3.1	12.2 ± 7.5	**5.2 ± 0.4**	**6.4 ± 1.7**	9.2 ± 2.6	10.0 ± 3.2
COV_7	**5.8 ± 0.8**	**5.8 ± 0.8**	5.8 ± 1.8	5.8 ± 1.8	28.0 ± 6.4	58.2 ± 19.9
ECO	**5.0 ± 0.0**	**5.0 ± 0.0**	6.4 ± 2.1	7.6 ± 4.2	35.2 ± 10.9	54.4 ± 24.2
EME	**5.4 ± 0.9**	**5.8 ± 1.8**	**5.4 ± 0.9**	6.2 ± 2.7	44.6 ± 4.6	75.2 ± 17.3
HIG	6.4 ± 1.5	6.2 ± 1.5	**5.6 ± 1.3**	**5.8 ± 1.3**	23.2 ± 7.2	48.6 ± 21.4
KDD_2	**5.4 ± 0.5**	**5.4 ± 0.5**	5.4 ± 0.9	5.6 ± 1.3	8.0 ± 1.4	8.2 ± 1.3
POK	6.6 ± 1.3	9.2 ± 3.3	**5.6 ± 1.3**	**6.0 ± 2.2**	25.4 ± 3.1	34.2 ± 8.4
SUS	**6.6 ± 2.5**	**7.4 ± 3.8**	7.0 ± 1.2	7.6 ± 1.9	15.0 ± 6.9	22.0 ± 14.0
AVERAGE	6.100 ± 1.325	7.125 ± 2.400	5.800 ± 1.238	6.375 ± 2.138	23.575 ± 5.388	38.850 ± 13.713

differences between the *control approach*, i.e. the one with the lowest Friedman rank, and the remaining approaches. Details on the aforementioned tests may be found in [34].

Table 9 shows the results of the application of the Friedman and of the Iman and Davenport tests on the accuracy values obtained over the test set. The null hypothesis for the Iman and Davenport test can never be rejected (the p-values are always greater than 0.05). Thus, we can conclude that the three algorithms are statistically equivalent in terms of accuracy. On the other hand, it is worth noticing that DPAES-FDT-GL and DPAES-FDT achieve the hightest ranks for the FIRST solutions.

Table 10 shows the results of the application of the Friedman and of the Iman and Davenport tests on the compexities. In this case, the null hypothesis associated with the Iman and Davenport test is always rejected (the p-values are always lower than 0.05). Thus, we performed the Holm post-hoc procedure by considering DPAES-FDT-GL, DPAES-FDT and PDAES-FDT-GL as control approaches for the FIRST, MEDIAN and LAST solutions, respectively. By analyzing Table 11 we can conclude that the DPAES-RCS solutions are always statistically more complex

Table 9 Results of the Friedman and of the Iman and Davenport tests on the accuracy computed on the test set.

	Algorithm	*Friedman rank*	*Iman and Davenport p-value*	*Hypothesis*
	DPAES-FDT-GL	1.875		
FIRST	DPAES-FDT	1.875	0.7145	Not Rejected
	DPAES-RCS	2.25		
	DPAES-FDT	1.875		
MEDIAN	DPAES-RCS	2	0.714	Not Rejected
	DPAES-FDT-GL	2.25		
	DPAES-FDT-GL	1.75		
LAST	DPAES-FDT	2.125	0.7145	Not Rejected
	DPAES-RCS	2.125		

Table 10 Results of the Friedman and of the Iman and Davenport tests on the complexity.

	Algorithm	*Friedman rank*	*Iman and Davenport p-value*	*Hypothesis*
	DPAES-FDT-GL	1.437		
FIRST	DPAES-FDT	1.812	0.0139	Rejected
	DPAES-RCS	2.75		
	DPAES-FDT	1.375		
MEDIAN	DPAES-FDT-GL	1.75	0.0013	Rejected
	DPAES-RCS	2.875		
	DPAES-FDT-GL	1.562		
LAST	DPAES-FDT	1.562	0.0025	Rejected
	DPAES-RCS	2.875		

than those of the control algorithms. On the other hand, the complexity of the solutions generated by DPAES-FDT-GL and DPAES-FDT are always statistically equivalent. In conclusion, both DPAES-FDT-GL and DPAES-FDT outperform DPAES-RCS in terms of complexity. It is worth noticing that, for the FIRST and the LAST solutions, DPAES-FDT-GL achieves the best Friedman rank. Indeed, as discussed above, in most of the cases, the complexities of the DPAES-FDT-GL solutions are lower than those generated by DPAES-FDT.

Table 11 Results of the Holm post hoc procedures on the complexity for $\alpha = 0.05$

	i	**algorithm**	z**-value**	p**-value**	**alpha/**i	**Hypothesis**
FIRST	2	DPAES-RCS	2.625	0.0086	0.025	Rejected
	1	DPAES-FDT	0.75	0.4532	0.05	Not Rejected
MEDIAN	2	DPAES-RCS	3	0.0027	0.025	Rejected
	1	DPAES-FDT-GL	0.75	0.4532	0.05	Not Rejected
LAST	2	DPAES-RCS	2.62	0.0086	0.025	Rejected
	1	DPAES-FDT	0	1	0.05	Not Rejected

For an easier visual comparison of the widths of the Pareto front approximations obtained by DPAES-FDT-GL, DPAES-FDT and DPAES-RCS, in Fig. 8 we plot, on the classification rate/TRL plane, the average values achieved by the three representative solutions, for all the datasets, on both the training and test sets. Here the solutions generated by DPAES-FDT-GL, DPAES-FDT and DPAES-RCS are reported as blue diamond, empty black circle and red plus markers, respectively. The results provided in Table 7 and 8 can thus be visually evaluated, and the trends of the results previously discussed can be easily identified.

In conclusion, we can state that employing the distributed FDT, rather than a distributed version of the C4.5, allows the MOEL process to generate more compact FRBCs. Moreover, even though we cannot find statistical differences between the complexities of the FRBCs generated by DPAES-FDT-GL and DPAES-FDT, the activation of the granularity learning allows us to reduce, in most of the cases, the number of rules and the *TRL* of the generated classifiers. The good behaviour of DPAES-FDT-GL can be mainly attributed to the following considerations. First of all, the FDT learning algorithm generates fuzzy decision trees directly from fuzzy partitions. Thus, the tree is tuned to the fuzzy partitions. On the other hand, the C4.5 learning algorithm used in DPAES-RCS generates decision trees from crisp partitions. Indeed, the fuzzy partitions are actually considered crisp for the execution of the learning algorithm: each fuzzy set is approximated by using a crisp set that corresponds to the α-cut with $\alpha = 0.5$, preserving the same label as in the corresponding fuzzy set. Once the tree is learned, then the rules are extracted from the tree and the labels re-assigned to the original fuzzy sets. Thus, the decision tree (differently from FDT) is not tuned to the final fuzzy partitions. Second, the granularity learning process allows reducing the number of fuzzy sets for each linguistic variable. The lower the number of fuzzy sets that describe each partition, the lower the number of combinations that can be obtained for generating classification rules. These two aspects mainly contribute to the good behaviour exhibited by DPAES-FDT-GL, which achieves more compact solutions than DPAES-RCS. This result is achieved thanks to the synergy among the initial set of fuzzy rules extracted from the FDT, granularity learning, rule and condition selection, and fuzzy set parameter learning. Indeed, the membership function parameter learning allows adapting the fuzzy partitions to the dataset, also when using a low number of fuzzy sets for each linguistic attribute. Thus, the number of rules can decrease and the accuracy increase during the evolutionary process.

As a final remark, we briefly compare the results achieved by DPAES-FDT-GL with those obtained by a distributed multi-way fuzzy decision tree (DMFDT) learning algorithm [53], and a distributed fuzzy associative classifier for big data (DFAC-FFP) [52]. We highlight that DMFDT exploits the same FDT learning algorithm used to generate the initial set of fuzzy rules in DPAES-FDT-GL, but employs fuzzy partitions generated by a distributed fuzzy discretizer, leaves labelled with different classes and a weighted voting inference strategy. To this aim, we selected HIGGS, KDD_2 and SUSY datasets. We chose only these three datasets since the relative results are the unique ones available in both [53] and [52], where DMFDT and DFAC-FFP were proposed. Furthermore, HIGGS and SUSY are the largest datasets in terms of memory occupancy.

On HIGGS, DMFDT achieves the highest accuracy; the lower complexity of both DPAES-FDT-GL and DFAC-FFP is balanced by a lower classification accuracy. Furthermore, while the accuracies of DPAES-FDT-GL and DFAC-FFP are comparable, the model complexities are different by about 2 order of magnitudes. On KDD_2, the three algorithms achieve more or less the same accuracy, but the complexity of DPAES-FDT-GL is one order of magnitude smaller than the one

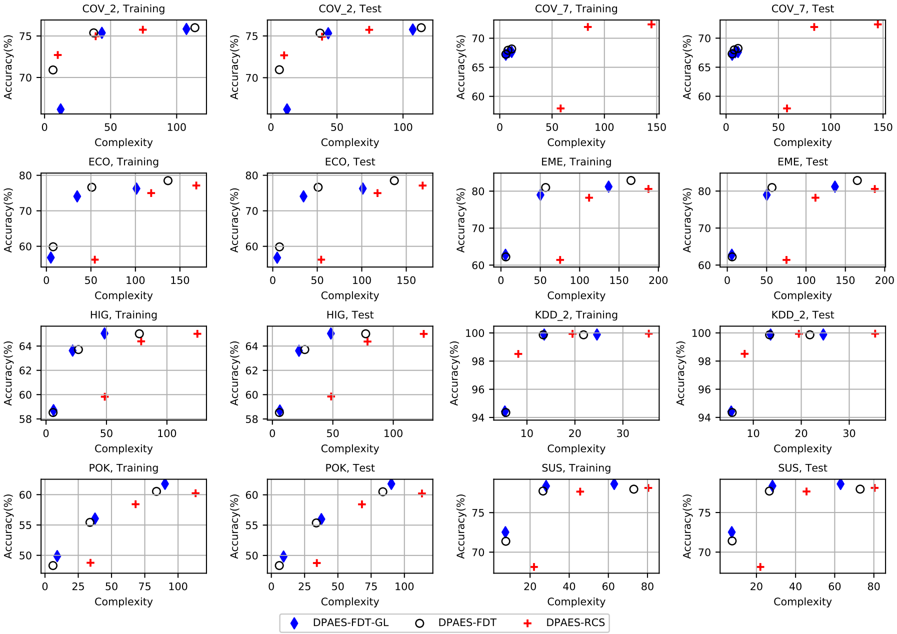

Fig. 8 Plots of the average accuracy on the training and test sets and average *TRL* of the FIRST, MEDIAN and LAST solutions generated by DPAES-FDT-GL (blue diamond markers), DPAES-FDT (empty black circle markers) and DPAES-RCS (red plus symbol markers).

Table 12 Comparison of the average accuracies on the test set and average complexities for DPAES-FDT-GL, DMFDT and DFAC-FFP. Complexity is measured as average number (M) of rules and average *TRL* for DPAES-FDT-GL, average number of nodes and leaves for DMFDT, and average number (M) of rules for DFAC-FFP.

	DPAES-FDT-GL			**DMFDT**			**DFAC-FFP**	
Dataset	Acc_{Tst}	M	TRL	Acc_{Tst}	$\#Leaves$	$\#Nodes$	Acc_{Tst}	M
HIG	65.035 ± 0.004	14.0	48.4	71.253 ± 0.029	920,942	972,779	66.005 ± 0.078	9,365
KDD_2	99.886 ± 0.010	10.8	24.6	99.986 ± 0.005	703	630	99.998 ± 0.001	890
SUS	78.608 ± 0.004	14.6	63.0	79.639 ± 0.016	758,064	805,076	78.267 ± 0.050	10,970

of the two comparison algorithms. More interestingly, DMFDT achieves a classification accuracy of $\sim 79.6\%$ on the SUSY dataset; it is $\sim 1.1\%$ higher of that achieved by DPAES-FDT-GL, yet it has been obtained with $805,076$ nodes and $758,064$ leaves, thus with a system of 4 orders of magnitude more complex than the one generated by DPAES-FDT-GL. Finally, it is worth noticing that DPAES-FDT-GL achieves better results than DFAC-FFP, with a complexity smaller by two orders of magnitude.

5 Conclusions and Future Work

In this paper, we have presented a novel approach, denoted as DPAES-FDT-GL, for generating sets of fuzzy rule-based classifiers with different optimal trade-offs between accuracy and interpretability from big data. The approach extends DPAES-RCS, a distributed multi-objective evolutionary algorithm recently proposed on the Apache Spark framework. The extensions regard two main aspects. First, the initial set of candidate rules used in the multi-objective evolutionary learning is extracted from a fuzzy decision tree (FDT) rather than a crisp decision tree. The FDT is generated by a distributed learning algorithm recently proposed by one of the authors. Second, the granularity of each numerical attribute is determined during the evolutionary process. We have executed DPAES-FDT-GL on 8 big datasets and have compared the results to the ones obtained by DPAES-RCS. Although the accuracy achieved by the fuzzy rule-based classifiers generated by DPAES-FDT-GL is statistically comparable to the one obtained by the classifiers generated by DPAES-RCS, the models generated by DPAES-FDT-GL are characterized by the lowest number of rules, conditions, and fuzzy sets. We can conclude that DPAES-FDT-GL represents an important step forward in getting interpretable fuzzy classifiers in the context of big data. Since there exists a number of real applications that require not only high accuracy, but also high interpretability, we strongly believe that DPAES-FDT-GL can be a very interesting and promising approach for such applications.

In order to disentangle the contribution of the FDT from that of the granularity learning, we also performed a comparison with DPAES-FDT, a version of DPAES-FDT-GL, which adopts the FDT for generating the initial rule set, but no granularity learning during the evolutionary process. We observed that, when extracting the initial set of rules from an FDT, we obtain models that are always statistically less complex. Moreover, even though we cannot find statistical differences between the complexities of the FRBCs generated by DPAES-FDT-GL and DPAES-FDT, we observed that the activation of the granularity learning allows reducing, in most of the cases, the number of rules and the TRL of the generated classifiers.

Future works will address the problem, in the specific setting of the described approach, of bounding the size of the training set without experiencing losses in the achieved accuracy. This aspect is crucial in dealing with Big Data, and effective solutions can extend the practical applicability of DPAES-FDT-GL to extremely big dataset, with no significant additional penalties in the runtimes for the learning phase. Indeed, the main problem we have to cope with when using EFS with Big Data is the computation of the accuracy on the overall training set. This computation depends on the number of instances in the training set and on the dimensionality of each instance. In Big Data generally both these numbers are high and then require long runs before achieving satisfactory solutions. Thus, techniques for reducing the number of attributes and the numerosity of the datasets, preserving the accuracy achieved by the models, are very appealing. As regards attribute reduction, our approach already performs a selection of attributes when we apply the FDT algorithm for generation the initial set of rules: As regards attribute reduction, our approach already performs a selection of attributes when we apply the FDT algorithm for generating the initial set of rules: indeed, the attributes that are considered in no decision node are removed. Furthermore, during the evolutionary process of RCS, attributes that are included in no rule can be eliminated. Nevertheless, we would like to investigate an appropriate chromosome coding for performing explicitly attribute selection during the evolutionary optimization. The reduction of the instance numerosity can be performed with the amount of approaches that have been proposed in the literature, but that need to be adequately tuned to the specific setting of the proposed algorithm. Further, in the past we proposed a co-evolutionary approach for instance selection [8], which should be adapted to manage big data.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with the active participation of humans. Furthermore, this article does not contain any studies on animals. The data collected and processed will be solely used for research related to this work and it will be ensured that they will not allow to identify any of the authors of such data.

Funding This work was been partially supported by the University of Pisa under grant PRA_2017 "IoT e Big Data: metodologie e tecnologie per la raccolta e l'elaborazione di grosse moli di dati." Moreover, the work carried out in implementing the described approach is part of the efforts for the development of the projects "SIBILLA" and "TALENT," co-financed by Regione Toscana under the framework POR-FESR 2014-2020 - Bando 2.

References

1. Abdullah, A., Hussain, A., Khan, I.H.: Introduction: Dealing with Big Data - lessons from cognitive computing. Cognitive Computation **7**(6), 635–636 (2015). DOI 10.1007/s12559-015-9364-6
2. Al-Ali, A., Zualkernan, I.A., Rashid, M., Gupta, R., Alikarar, M.: A smart home energy management system using IoT and Big Data analytics approach. IEEE Transactions on Consumer Electronics **63**(4), 426–434 (2017). DOI 10.1109/TCE.2017.015014
3. Aljarah, I., Al-Zoubi, A.M., Faris, H., Hassonah, M.A., Mirjalili, S., Saadeh, H.: Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cognitive Computation **10**(3), 478–495 (2018). DOI 10.1007/s12559-017-9542-9
4. Antonelli, M., Ducange, P., Lazzerini, B., Marcelloni, F.: Learning concurrently partition granularities and rule bases of Mamdani fuzzy systems in a multi-objective evolutionary framework. International Journal of Approximate Reasoning **50**(7), 1066–1080 (2009). DOI 10.1016/j.ijar.2009.04.004
5. Antonelli, M., Ducange, P., Lazzerini, B., Marcelloni, F.: Multi-objective evolutionary learning of granularity, membership function parameters and rules

of Mamdani fuzzy systems. Evolutionary Intelligence **2**(1-2), 21–37 (2009). DOI 10.1007/s12065-009-0022-3

6. Antonelli, M., Ducange, P., Lazzerini, B., Marcelloni, F.: Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity. Soft Computing **15**(12), 2335–2354 (2011). DOI 10.1007/s00500-010-0665-0
7. Antonelli, M., Ducange, P., Lazzerini, B., Marcelloni, F.: Multi-objective evolutionary design of granular rule-based classifiers. Granular Computing **1**(1), 37–58 (2016)
8. Antonelli, M., Ducange, P., Marcelloni, F.: Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach. IEEE Transactions on Fuzzy Systems **20**(2), 276–290 (2012). DOI 10.1109/TFUZZ.2011.2173582
9. Antonelli, M., Ducange, P., Marcelloni, F.: A fast and efficient multi-objective evolutionary learning scheme for fuzzy rule-based classifiers. Information Sciences **283**, 36–54 (2014). DOI 10.1016/j.ins.2014.06.014
10. Antonelli, M., Ducange, P., Marcelloni, F.: Multi-objective evolutionary design of fuzzy rule-based systems. In: HANDBOOK ON COMPUTATIONAL INTELLIGENCE: Volume 2: Evolutionary Computation, Hybrid Systems, and Applications, pp. 635–670. World Scientific (2016)
11. Anuradha, J., et al.: A brief introduction on Big Data 5Vs characteristics and Hadoop technology. Procedia computer science **48**, 319–324 (2015). DOI 10.1016/j.procs.2015.04.188
12. Ayesh, A., Blewitt, W.: Models for computational emotions from psychological theories using type I fuzzy logic. Cognitive Computation **7**(3), 285–308 (2015). DOI 10.1007/s12559-014-9287-7
13. Baldi, P., Sadowski, P., Whiteson, D.: Searching for exotic particles in high-energy physics with deep learning. Nature Communications **5** (2014). DOI 10.1038/ncomms5308
14. Bechini, A., Marcelloni, F., Segatori, A.: A MapReduce solution for associative classification of big data. Information Sciences **332**, 33–55 (2016). DOI 10.1016/j.ins.2015.10.041
15. Bechini, A., Matteis, A.D.D., Marcelloni, F., Segatori, A.: Spreading fuzzy random forests with MapReduce. In: 2016 IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC), pp. 2641–0646 (2016). DOI 10.1109/SMC.2016.7844638
16. Cai, Z., Shao, L.: RGB-D scene classification via multi-modal feature learning. Cognitive Computation (2018). DOI 10.1007/s12559-018-9580-y
17. Chi, Z., Yan, H., Phạm, T.: Fuzzy algorithms: with applications to image processing and pattern recognition, *Advances in Fuzzy Systems - Applications and Theory*, vol. 10. World Scientific (1996). DOI 10.1142/3132
18. Cococcioni, M., Ducange, P., Lazzerini, B., Marcelloni, F.: A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. Soft Computing **11**(11), 1013–1031 (2007). DOI 10.1007/s00500-007-0150-6
19. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary algorithms for solving multi-objective problems, vol. 5, 2nd edn. Springer (2007). DOI 10.1007/978-0-387-36797-2
20. Contreras, D., Salamó, M.: A cognitively inspired clustering approach for critique-based recommenders. Cognitive Computation (2018). DOI 10.1007/

s12559-018-9586-5

21. Dai, W., Ji, W.: A MapReduce implementation of C4.5 decision tree algorithm. Int'l Journal of Database Theory and Application **7**(1), 49–60 (2014). DOI 10.14257/ijdta.2014.7.1.05
22. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Comm. of the ACM **51**(1), 107–113 (2008). DOI 10.1145/1327452.1327492
23. Ducange, P., Pecori, R., Mezzina, P.: A glimpse on big data analytics in the framework of marketing strategies. Soft Computing **22**(1), 325–342 (2018). DOI 10.1007/s00500-017-2536-4
24. Duţu, L.C., Mauris, G., Bolon, P.: A fast and accurate rule-base generation method for Mamdani fuzzy systems. IEEE Transactions on Fuzzy Systems **26**(2), 715–733 (2018). DOI 10.1109/TFUZZ.2017.2688349
25. Elkano, M., Galar, M., Sanz, J., Bustince, H.: CHI-BD: A fuzzy rule-based classification system for Big Data classification problems. Fuzzy Sets and Systems **348**, 75–101 (2018). DOI 10.1016/j.fss.2017.07.003
26. Elkano, M., Galar, M., Sanz, J., Bustince, H.: CHI-PG: A fast prototype generation algorithm for Big Data classification problems. Neurocomputing **287**, 22–33 (2018). DOI 10.1016/j.neucom.2018.01.056
27. Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., Herrera, F.: A review of the application of multi-objective evolutionary fuzzy systems: Current status and further directions. IEEE Transactions on Fuzzy Systems **21**(1), 45–65 (2013). DOI 10.1109/TFUZZ.2012.2201338
28. Fernández, A., Almansa, E., Herrera, F.: Chi-Spark-RS: an Spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6. IEEE (2017). DOI 10.1109/FUZZ-IEEE.2017.8015520
29. Fernández, A., Carmona, C.J., del Jesus, M.J., Herrera, F.: A view on fuzzy systems for Big Data: Progress and opportunities. Int'l Journal of Computational Intelligence Systems **9**(sup1), 69–80 (2016). DOI 10.1080/18756891.2016.1180820
30. Fernández, A., del Río, S., Bawakid, A., Herrera, F.: Fuzzy rule based classification systems for big data with MapReduce: granularity analysis. Advances in Data Analysis and Classification **11**(4), 711–730 (2017). DOI 10.1007/s11634-016-0260-z
31. Fernández, A., del Río, S., López, V., Bawakid, A., del Jesus, M.J., Benítez, J.M., Herrera, F.: Big data with cloud computing: an insight on the computing environment, MapReduce, and programming frameworks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **4**(5), 380–409 (2014). DOI 10.1002/widm.1134
32. Ferranti, A., Marcelloni, F., Segatori, A., Antonelli, M., Ducange, P.: A distributed approach to multi-objective evolutionary generation of fuzzy rule-based classifiers from big data. Information Sciences **415**, 319–340 (2017). DOI 10.1016/j.ins.2017.06.039
33. Gacto, M.J., Alcalá, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. Information Sciences **181**(20), 4340–4360 (2011). DOI 10.1016/j.ins.2011.02.021
34. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case

study on the cec'2005 special session on real parameter optimization. Journal of Heuristics **15**(6), 617–644 (2009)

35. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd ed. edn. Data Management Systems. Morgan Kaufmann (2012). DOI 10.1016/C2009-0-61819-5
36. Ishibuchi, H., Nakashima, T., Murata, T.: Three-objective genetics-based machine learning for linguistic rule extraction. Information Sciences **136**(1-4), 109–133 (2001)
37. Ishibuchi, H., Yamamoto, T.: Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy sets and systems **141**(1), 59–88 (2004)
38. Kim, S.S., McLoone, S., Byeon, J.H., Lee, S., Liu, H.: Cognitively inspired artificial bee colony clustering for cognitive wireless sensor networks. Cognitive Computation **9**(2), 207–224 (2017)
39. Kim, Y., Shim, K., Kim, M.S., Lee, J.S.: DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce. Information Systems **42**, 15–35 (2014). DOI 10.1016/j.is.2013.11.002
40. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary computation **8**(2), 149–172 (2000). DOI 10.1162/106365600568167
41. López, V., del Río, S., Benítez, J.M., Herrera, F.: Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets and Systems **258**, 5–38 (2015). DOI 10.1016/j.fss.2014.01.015
42. Ludwig, S.A.: MapReduce-based fuzzy C-means clustering algorithm: implementation and scalability. International journal of machine learning and cybernetics **6**(6), 923–934 (2015). DOI 10.1007/s13042-015-0367-0
43. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: kNN-IS: An iterative Spark-based design of the k-nearest neighbors classifier for big data. Knowledge-Based Systems **117**, 3–15 (2017). DOI 10.1016/j.knosys.2016.06.012
44. Márquez, A., Márquez, F., Peregrín, A.: A scalable evolutionary linguistic fuzzy system with adaptive defuzzification in big data. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6. IEEE (2017). DOI 10.1109/FUZZ-IEEE.2017.8015753
45. Mayer-Schönberger, V., Cukier, K.: Big data: A revolution that will transform how we live, work, and think. Eamon Dolan/Houghton Mifflin Harcourt (2013)
46. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review **63**(2), 81 (1956). DOI 10.1037/h0043158
47. Oneto, L., Bisio, F., Cambria, E., Anguita, D.: Semi-supervised learning for affective common-sense reasoning. Cognitive Computation **9**(1), 18–42 (2017). DOI 10.1007/s12559-016-9433-5
48. Ramírez-Gallego, S., Fernández, A., García, S., Chen, M., Herrera, F.: Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce. Information Fusion **42**, 51–61 (2018). DOI 10.1016/j.inffus.2017.10.001
49. Rey, M., Galende, M., Fuente, M., Sainz-Palmero, G.: Multi-objective based fuzzy rule based systems (FRBSs) for trade-off improvement in accuracy and interpretability: A rule relevance point of view. Knowledge-Based Systems

127, 67 – 84 (2017). DOI 10.1016/j.knosys.2016.12.028

50. Ricatto, M., Barsacchi, M., Bechini, A.: Interpretable CNV-based tumour classification using fuzzy rule based classifiers. In: Proc. of the 33rd ACM Symposium on Applied Computing, SAC 18. ACM, New York, NY, USA (2018). DOI 10.1145/3167132.3167135
51. del Río, S., López, V., Benítez, J.M., Herrera, F.: A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. Int'l Journal of Computational Intelligence Systems **8**(3), 422–437 (2015). DOI 10.1080/18756891.2015.1017377
52. Segatori, A., Bechini, A., Ducange, P., Marcelloni, F.: A distributed fuzzy associative classifier for big data. IEEE Transactions on Cybernetics (2017). DOI 10.1109/TCYB.2017.2748225
53. Segatori, A., Marcelloni, F., Pedrycz, W.: On distributed fuzzy decision trees for big data. IEEE Transactions on Fuzzy Systems **26**(1), 174–192 (2018). DOI 10.1109/TFUZZ.2016.2646746
54. Van Veldhuizen, D.A., Zydallis, J.B., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation **7**(2), 144–173 (2003). DOI 10.1109/TEVC.2003.810751
55. Wan, J., Tang, S., Li, D., Wang, S., Liu, C., Abbas, H., Vasilakos, A.V.: A manufacturing big data solution for active preventive maintenance. IEEE Transactions on Industrial Informatics **13**(4), 2039–2047 (2017). DOI 10.1109/TII.2017.2670505
56. Wang, H., Xu, Z., Pedrycz, W.: An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities. Knowledge-Based Systems **118**, 15–30 (2017). DOI 10.1016/j.knosys.2016.11.008
57. White, T.: Hadoop: The definitive guide. O'Reilly Media, Inc. (2012)
58. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. IEEE Transactions on Knowledge and Data Engineering **26**(1), 97–107 (2014). DOI 10.1109/TKDE.2013.109
59. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, vol. 10, p. 10 (2010)
60. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: Opportunities and challenges. Neurocomputing **237**, 350–361 (2017). DOI 10.1016/j.neucom.2017.01.026