A Genetic Fuzzy System for Large-scale Regression

I. Rodríguez-Fdez*, M. Mucientes*, and A. Bugarín*
*Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela
Santiago de Compostela, Spain
Email: ismael.rodriguez@usc.es, manuel.mucientes@usc.es, alberto.bugarin.diz@usc.es

***Abstract*—In genetic fuzzy systems (GFS) the size of the problem has a huge influence in the performance of the obtained models, since i) the fuzzy rule bases learned suffer from exponential rule explosion when the number of variables increases, and ii) the convergence time increments with the number of examples. In this paper we present S-FRULER, a scalable distributed version of FRULER which is a GFS that learns simple and linguistic TSK-1 knowledge bases for regression problems. S-FRULER obtains models with high accuracy and low complexity, whilst reducing the algorithm runtime. S-FRULER focuses on splitting the problem into smaller partitions and incorporates a feature selection process for reducing the number of variables used in each partition. Each partition is then solved independently using the FRULER algorithm. The Reduce function obtains linguistic TSK fuzzy rule bases from the information generated in each partition. S-FRULER has been validated in terms of scalability, precision and complexity using 10 large-scale datasets and has been compared with three state of the art GFSs. Experimental results show that S-FRULER scales well while achieving simple models with a linguistic approach and a precision comparable with approximative models.**

I. Introduction

The vast majority of techniques in statistical learning and data mining were traditionally designed to work with a limited amount of data [1]. The adaptation of these approaches to large scale datasets has been a huge challenge addressed in the last years [2]. Usually, all the problems that emerge with the use of big amounts of data are labelled under the term Big Data, which covers distributed processing and storing of data. Statistical learning algorithms in a Big Data context suffer from the problem of scalability, that is, the capability of the algorithm to maintain competitive performance as the size of the problem increases. Performance of the algorithms refers in this context to the computational cost, the precision of the obtained models, and their complexity. Also, the size of the problems can grow in two different dimensions: the number of examples available in training and the number of variables considered.

Particularly, in a genetic fuzzy system (GFS) approach, the size of the problem has a huge influence in the performance of the obtained models [3], [4]. The fuzzy rule bases learned suffer from exponential rule explosion when the number of variables increases. Thus, with huge search spaces, the convergence time towards precise and simple models rises. Moreover, evolutionary algorithms are computationally expensive by themselves due to the large number of evaluations needed to reach convergence. Furthermore, in many cases, the evaluation process to obtain the fitness may take a long time. There are different techniques to improve the scalability of GFS that can be classified into three categories [5]: i) algorithm oriented that adapts the structure of the evolutionary algorithm [6], [7], [8], [9], [10], ii) data oriented that modifies the training data to reduce the computational cost of the learning process [8], [11], and iii) distributed approaches that take advantage of the availability of several machines to reduce the runtime.

Solving large scale regression problems with GFSs can be found in some of the most recent works in the field [6], [9], [10]. However, the number of variables and/or number of examples in the datasets used in these works are still not high enough to be properly labelled as Big Data. Among the different approaches, FRULER [10] obtains Takagi-Sugeno-Kang 1-order (TSK-1) fuzzy rule bases with high accuracy and the lowest number of rules. Although the runtime of this approach is acceptable (between 1-23 minutes) for the most simple datasets, it does not scale properly when solving large scale problems (from 1 to 30 hours). Moreover, with larger problems it may not converge to a good solution in reasonable time.

In this paper we propose a scalable version of FRULER, called S-FRULER, which allows to obtain models with similar characteristics than those obtained by FRULER —accurate and simple—, but reducing the runtime of the algorithm to converge. The main contributions of this work are: i) a novel distributed GFS approach that uses the Spark software, ii) a random feature selection process to reduce the number of variables used in each dataset partition, and iii) an aggregation function to obtain linguistic TSK fuzzy rule bases from the rule bases obtained in each dataset partition.

II. S-FRULER

This section presents S-FRULER (Scalable Fuzzy Rule Learning through Evolution for Regression), a distributed approach for applying FRULER [10] with scalability properties that allow its application to large scale problems. FRULER is a GFS that learns simple and linguistic TSK-1 knowledge bases for regression problems. It is composed of a two-stage preprocessing — formed by an instance selection and a multi-granularity fuzzy discretization—, and a genetic algorithm, which contains an ad-hoc TSK-1 rule generation module. Although the runtime of this approach is acceptable for

This research was supported by the Spanish Ministry of Economy and Competitiveness (grant TIN2014-56633-C3-1-R) and the Galician Ministry of Education (grants EM2014/012, CN2012/151 and GRC2014/030). All grants were co-funded by the European Regional Development Fund (FEDER program).

978-1-5090-0626-7/16/$31.00

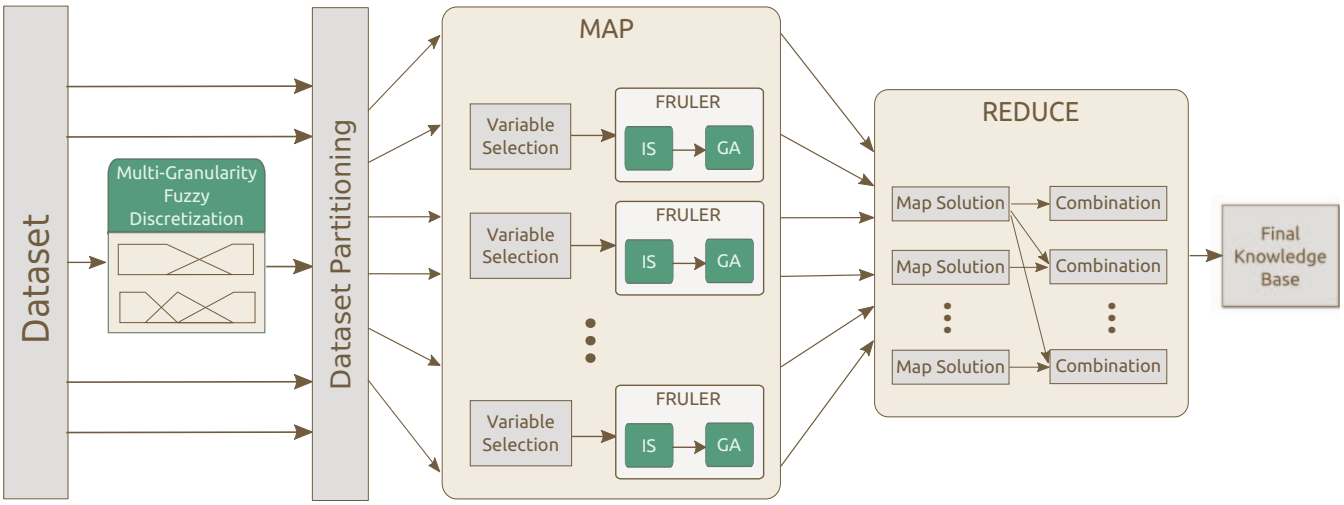

Figure 1: S-FRULER architecture showing the preprocessing, Map and Reduce phases.

medium size datasets, it does not scale properly when solving large scale problems as it may not converge. To solve that, S-FRULER divides the problem into a set of smaller problems that are more tractable using a distributed approach. Each of the divisions is then solved independently in the Map phase using FRULER. Then, the solutions obtained in each Map are combined in the Reduce phase in order to obtain a final solution for the original data.

The algorithm structure is shown in Fig. 1. First, the multi-granularity fuzzy discretization process is performed using the whole training dataset. Then, the training dataset is splitted into n_{map} partitions, which correspond to the tasks to be distributed into the Working Nodes. Also, for each dataset partition, only a subset of randomly selected variables is taken into account. Each task is solved using FRULER —without the discretization phase—, as if it was an independent problem. Thus, only the instance selection (IS) and the genetic algorithm (GA) are executed. After obtaining the solutions for each task, these are combined completing the variables not used in one dataset partition with information of the others. The following subsections describe each of these phases in more detail.

A. Preprocessing before mapping

This stage comprises the multi-granularity fuzzy discretization of the input variables and the partition of the training dataset.

1) Multi-granularity Fuzzy Discretization: The first step of S-FRULER consists in the application of the Multi-granularity Fuzzy Discretization method used in FRULER [10] to discretize the input variables into fuzzy labels. This method obtains non-uniform fuzzy partitions with different degrees of granularity. A granularity g^i_{var} divides the variable var into i fuzzy labels, i.e., $g^i_{var} = \{A^{i,1}_{var}, \ldots, A^{i,i}_{var}\}$.

The algorithm works as follows:

- First, each variable is discretized to obtain a set of split points C^g for each granularity g.
  - The split points are searched iteratively, i.e., only a new split point is added at each granularity, starting from the most general granularity. Therefore, the approach proposed in this work aims to preserve interpretability between contiguous granularities: adding a new label to the previous granularity and modifying the adjacent labels.
  - The split points that minimize the error when a linear model is applied to each of the resulting intervals are selected.
  - The method adds split points until a bayesian information criterion (BIC) worsens. To calculate the BIC, the error is measured as the summation of the mean squared error of a least squares fitted model for each interval of the discretization, while the complexity is determined by the number of inner splits and the parameters fitted by each regression applied in each interval.
- Then, the method proposed in [12] is applied to each C^g —set of split points for the granularity g— in order to get the multi-granularity fuzzy partitions. This method uses a parameter that assesses the fuzziness of the linguistic labels. Fuzziness 0 indicates crisp intervals, while fuzziness 1 indicates the selection of a fuzzy set with the smallest kernel —set of points with membership equal to 1.

2) Dataset Partitioning: After the discretization of the input variables, the dataset is partitioned and distributed along the Workers Nodes of the computation cluster. The training set is divided randomly into n_{map} partitions with equal size $\frac{n}{n_{map}}$, where n is the total number of examples in the training data. In scalability terms, the higher the number of partitions n_{map} that can be used, the better. However, when the number of partitions surpasses a certain problem-dependent threshold, the obtained results worsen. This means that the problem has been divided too much, and each Map has not enough information

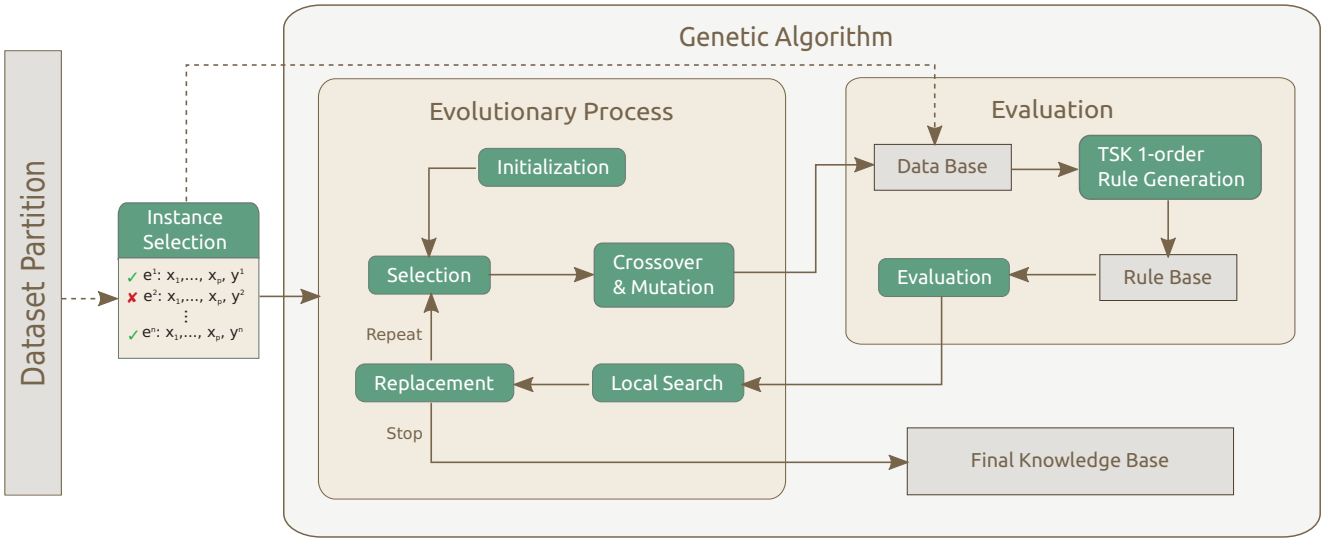

Figure 2: FRULER [10] architecture showing each of the two stages. Dashed lines indicate flow of data sets and solid lines represent process flow.

to get a valid result. Thus, the automatic definition of the most adequate number of partitions for each problem can be useful when no information about the underlying characteristics of the problem is known. In S-FRULER, the number of partitions n_{map} is defined heuristically as:

$$n_{map} = log_2(p^2 * l * n) \tag{1}$$

where p is the number of input variables, n is the number of examples and l is the maximum granularity over all the input variables.

B. Map function

The Map function is applied independently and distributively to each training dataset partition. For each training dataset partition, only a subset of selected variables is used. Thus, all the process done in each task uses the corresponding subset of examples and subset of variables, which is going to be referred as the dataset partition. For each task, FRULER is applied without the discretization step, since it was already performed before the partition of the data. It is composed by an instance selection of the most representative examples and a genetic algorithm, which contains an ad-hoc TSK-1 rule generation module (Fig. 2). The evolutionary learning process obtains a definition of the data base. Then an ad-hoc TSK-1 rule generation module obtains the antecedents and consequents of each possible rule using only the representative examples.

1) Variable Selection: To simplify the learning process in each task, only a subset of randomly selected variables is used for each dataset partition. The probability of selecting a particular input variable X_j in a dataset partition is:

$$P(X_j \in X_s^i) = \frac{p_m}{p}, \tag{2}$$

where X_s^i is the selected subset of input variables in the dataset partition i, p_m is the subset size of selected input variables and p is the total number of input variables. Thus, the probability that a particular input variable is not selected for all the dataset partitions is:

$$P(X_j \notin X_s^i, \forall i = 1, \ldots, n) = \left(\frac{p - p_m}{p}\right)^{n_{map}}. \tag{3}$$

where n_{map} is the total number of dataset partitions. Therefore, to define a subset size of selected input variables p_m that assures that a particular feature X_j is selected in at least one dataset partition with probability α_{p_m}, the following equation can be used:

$$p_m \geq -p \cdot ((1 - \alpha_{p_m})^{1/n_{map}} - 1) \tag{4}$$

Therefore, each task uses a dataset formed by $\frac{n}{n_{map}}$ examples from the original training dataset and p_m randomly selected input variables.

2) Instance Selection for Regression: The instance selection method for regression used in FRULER [10] is an improvement of the CCISR (Class Conditional Instance Selection for Regression) algorithm [11]. The method is based on a relation called class conditional nearest neighbor which is used to obtain the nearest example with the same class and the nearest example of a different class for each instance in the dataset partition. With this information, two graphs can be built: one where each example points to the nearest example with the same class and another where each example points to the nearest example with a different class. Using these graphs, the method obtains the in-degree for same-class and different-class for each example, and then an information measure based on the K-divergence is calculated. This measure is then used to sort the examples by its ability to represent its own class and to differentiate contiguous classes.

Since the class conditional relation uses classes, firstly it is necessary to discretize the output variable. For that, the Kernel Density Estimation (KDE) with a Gaussian Kernel is used to estimate the probability density function of the output. Then, the local minima of this function are used as the split points between classes, and, therefore, to determine which class corresponds to each example.

Then the instance selection method performs three stages:

- First, the k_0 first examples —sorted by the K-divergence score— are picked as the initial set of selected examples E_S, where k_0 is defined as:

$$k_0 = max\left(c, \left\lceil \frac{\epsilon^{E_{tra}} \cdot |E_{tra}|}{max(y) - min(y)} \right\rceil\right) \quad (5)$$

  where c is the number of classes obtained from KDE and $\epsilon^{E_{tra}}$ is the 1-nearest neighbor error for regression using the complete training dataset partition E_{tra}.
- Then, the second stage adds examples in order, until the error worsens for more than $\sqrt{|E_{tra}|/|E_S|}$ iterations.
- Finally, the last stage removes points that are not close to the decision boundary of the 1-nearest neighbor rule, that is, examples with zero in-degree in the different-class graph —there is no other instance that points to the example.

3) Genetic Algorithm: The evolutionary algorithm is visually described in the Evolutionary Process box in Fig. 2. Its objective is to get the best data base configuration using the obtained fuzzy partitions (Fig. 1). To evaluate each individual, the algorithm generates the entire linguistic TSK-1 fuzzy rule base using the selected fuzzy partitions and the selected examples E_S (sec. II-B2), and then calculates the Mean Squared Error (MSE) using the whole training dataset partition E_{tra}.

The chromosome has two parts: the first part (C_1) represents the granularity used in each input variable, while the second part (C_2) codifies the lateral displacements of the split points of the input variables fuzzy partitions. The initial pool of individuals is generated by a combination of two initialization procedures. A half of the individuals are generated with the same random granularity for each variable, while the other half is created with a different random granularity for each variable. The lateral displacements are initialized to 0 in all cases.

The iterative part of the evolutionary algorithm starts with a binary tournament selection process. Then, two crossover operations can be applied: one-point crossover for exchanging the C_1 parts (it also exchanges the corresponding C_2 genes) and, when the C_1 parts are equal, the parent-centric BLX (PCBLX) [13] is used to crossover the C_2 part. In order to prevent the crossover of too similar individuals, an incest prevention was implemented. After the crossover, the mutation is applied to each offspring with probability p_{mut}. It implements two possible operations with equal probability to a randomly selected gene of the C_1 part: i) decreasing the granularity by 1 or ii) increasing the granularity to a more specific granularity —all the granularities have the same chance. In order to calculate the new lateral displacements in the corresponding C_2 part, the algorithm uses the displacements of the two nearest split points of the previous granularity (before mutation) weighted by the distance between the split points.

The offsprings are evaluated using the mechanism described in Sec. II-B4. Then, a local search is performed for each offspring, generating n_{ls} new C_1 parts with equal or less granularity —with equal probability— for each variable and the C_2 is generated randomly. The new chromosomes are decoded and evaluated and, if there is a solution that obtains better fitness, then it replaces the original individual. After that, the previous and current populations are merged, and the N best individuals are selected as the new population. The algorithm incorporates a restart mechanism that uses an incest prevention threshold L as a trigger.

4) Evaluation: The evaluation process is described in the Evaluation box in Fig. 2. First, the real data base is obtained applying the displacements in C_2 to the fuzzy partitions selected in C_1. Then, the Wang & Mendel algorithm [14] is used to create the antecedent part of the rule base for each individual. The consequent part of the rules is learned using the Elastic Net method [15] in order to obtain the coefficients of the degree 1 polynomial for each rule. In order to solve the minimization problem of Elastic Net, a Stochastic Gradient Descent optimization technique was used [16], [17]. Only those examples in E_s are used to obtain the rule base from the codified chromosome. In this manner, those examples that are not representative are discarded for the rule generation.

Then, the resulting TSK-1 fuzzy rule base is evaluated using the following equation:

$$fitness = MSE(E_{tra}) = \frac{1}{2 \cdot |E_{tra}|} \sum_{i=1}^{|E_{tra}|} (F(x^i) - y^i)^2, \quad (6)$$

where E_{tra} is the full training dataset and $F(x^i)$ is the output obtained by the knowledge base for the input x^i. Using all the examples for evaluation can be seen, in some way, as a validation process, as the rule base was constructed with a subset of them (E_S).

C. Reduce function

After the execution of FRULER for each dataset partition the reduction phase is applied in order to get the final solution. Each of the FRULER executions obtains a TSK-1 Knowledge Base, composed by a linguistic partition of the input variables —data base— and the TSK fuzzy rule set —rule base. These Knowledge Bases may be combined using an ensemble technique, that takes into account the degree of fulfillment of the input data to each knowledge base, to calculate a weighted average of the outputs. However, this approach increases remarkably the complexity of the model, as the number of rules of the final solution increases with the number of partitions of the data.

Thus, to combine the solutions generated in the Map phase without increasing significantly the complexity, the properties of each of the obtained Knowledge Bases must be taken into

1: **function** REDUCE($S = \{s_1, s_2, \ldots, s_{n_{map}}\}, E_S = E_{S_1} \cup E_{S_2} \cup \cdots \cup E_{S_m}$)
2: **for** $i = 1, \ldots, n_{map}$ **do**
3: **for** $k = 1, \ldots, n_{map}$ **do**
4: $$r_{i,k} = \bigcup_{j=1,\ldots,p} \left\{ a_{i,k,j} \ : \ a_{i,k,j} = \left\{ \begin{array}{ll} s_{i,j} & \text{if } s_{i,j} \neq \text{ NULL} \\ s_{k,j} & \text{if } s_{i,j} = \text{ NULL and } s_{k,j} \neq \text{ NULL} \\ 1 & \text{otherwise} \end{array} \right. \right\}$$
5: **end for**
6: **end for**
7: $R = \cup\ r_{i,k},\ i = 1, \ldots, n_{map},\ k = 1, \ldots, n_{map}$
8: $error_{min} = \infty$
9: **for each** $r \in R$ **do**
10: $rb =$ Generate rule base from r using E_S
11: **if** *MSE*$(rb, E_{tra}) < error_{min}$ **then**
12: $error_{min} =$ *MSE*(rb, E_{tra})
13: $Best = rb$
14: **end for**
15: **return** $Best$

Figure 3: Pseudocode of the Reduce function.

account. On one hand, the rule base strongly depends on the label partitions of the data base. A change of granularity in one input variable partition can lead to a substantial change on the rule consequents. On the other hand, the granularity for each input variable can be easily combined, in a similar way to a crossover operation for integer valued individuals. This combination of granularities can take into account the variables not used in each dataset partition. After combining the granularities, the rule base can be generated using the ad-hoc TSK rule base Generation process of FRULER.

Fig. 3 shows the pseudocode of the Reduce function. The Reduce function uses two parameters: S contains the solutions generated by FRULER in each dataset partition as a list of granularities, and E_S contains the union of all the selected instances in each dataset partition. Thus, E_S can be seen as the selected instances of the entire training dataset if a stratification approach is used to perform the instance selection method taking each dataset partition as a stratum. The solution obtained in each dataset partition contains the displacements in addition to the granularity for each input variable[1].

The process is as follows: for each partition solution (lines 2) the Reduce function completes the input variables not previously selected in the dataset partition with information of the other solutions (line 3). Thus the Reduce function generates a maximum of n_{map}^2 final solutions. This combination is done, for each input variable j, as follows (line 4):

- The granularity $s_{i,j}$ (and the displacements associated with it) is selected if it exists.
- If the value is not defined in s_i ($s_{i,j} =$ NULL) but it is in s_k, then $s_{k,j}$ is used.
- Finally, if the value is not defined neither s_i nor s_k, this variable is not considered (granularity 1).

[1]This information was omitted in the pseudocode for the sake of clarity.

Note that the inner loop (line 4) takes also into account s_i, thus the original solution is also kept. Then, R (line 7) is the set of the final solutions that combine the information of the different results obtained in each dataset partition.

Fig. 4 illustrates how the solutions obtained by three partitions are combined with this Reduce function. In the (a) case, the third input variable was removed in the feature selection process and this information is completed with two choices: i) adding a granularity 1 to the variable (the variable is not used) and ii) the granularity indicated in the other solutions. In the (b) scenario, we show the particular case where the other solutions indicate the same granularity for the incomplete information of the selected solution. However, two solutions are created because the lateral displacement information may not be the same.

Once all the possible combinations have been obtained, the Reduce function generates the associated rule bases (line 10) using the ad-hoc TSK rule base generation process of FRULER. This process uses the combination of selected instances for all dataset partitions E_S. Then, the error is measured using the whole training dataset E_{tra} (line 11), so a validation is performed due to the use of examples not seen in the TSK rule base generation process. Finally, the solution with the lowest error (lines 8-14) is selected as the best solution and returned by the Reduce function (line 15).

III. RESULTS

In order to analyze the performance of S-FRULER, we have done a comparison with other GFSs using 10 regression problems from the KEEL project repository [18]. Table I shows the characteristics of the regression datasets, with the number of instances ranging from 7,129 to 40,768 examples, and the number of input variables from 5 to 40. The datasets are sorted in incremental order of the number of variables.

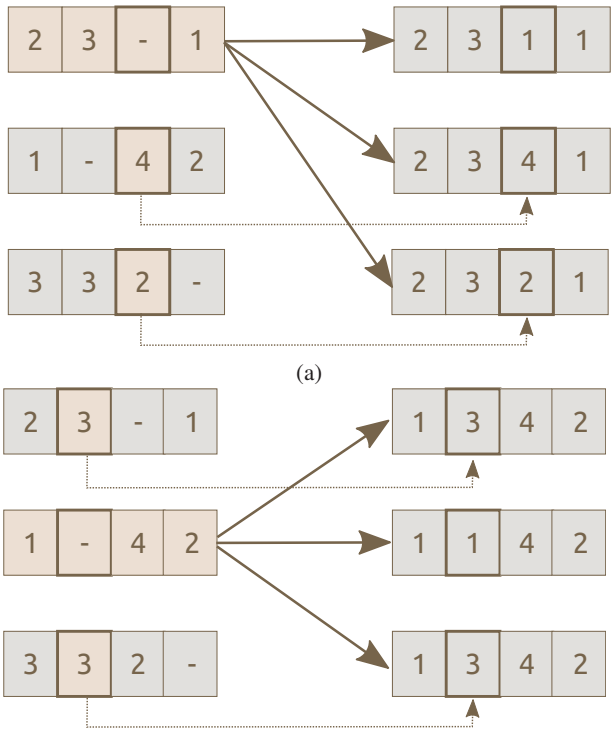

Figure 4: Visual illustration of the Reduce function for three different solutions obtained in the Map phase.

Problem	Abbr.	# Variables	# Cases
Delta Ailerons	DELAIL	5	7,129
Delta Elevators	DELELV	6	9,517
California Housing	CAL	8	20,640
MV Artificial Domain	MV	10	40,768
House-16H	HOU	16	22,784
Elevators	ELV	18	16,559
Computer Activity	CA	21	8,192
Pole Telecommunications	POLE	26	14,998
Pumadyn	PUM	32	8,192
Ailerons	AIL	40	13,750

Table I: The 10 datasets of the experimental study.

A. Experimental Setup

In terms of parameters, S-FRULER only adds to FRULER the parameter of the probability that a particular feature is selected in at least one dataset partition (α_{p_m}). For the experiments performed in this section $\alpha_{p_m} = 0.9$. Moreover, FRULER [10], which is embedded in S-FRULER, was designed to keep the number of parameters as low as possible. In the multi-granularity fuzzy discretization, the fuzziness parameter used for the generation of the fuzzy intervals was 1, i.e., the highest fuzziness value. For the instance selection technique, no parameters are needed. For the evolutionary algorithm, the values of the parameters were: population size $= 61$, maximum number of evaluations $= 100,000$, $p_{cross} = 1.0$, $p_{mut} = 0.2$, and the number of neighbours generated in the local search was $n_{ls} = 5$. For the generation of the TSK fuzzy rule bases, the weight of the tradeoff between ℓ_1 and ℓ_2 regularizations on the Elastic Net was $\alpha = 0.95$, and the regularization parameter λ was obtained from a grid search in the interval $[1, 1E-10]$. η^0 was obtained halving the initial value (0.1) until the result worsens.

For each dataset, we performed 10 trials (with different seeds for the random number generation) of S-FRULER. The results shown in the next section are the mean values over all the runs. The runtimes have been obtained in two different ways:

- Using the Spark Standalone mode, where the workers are simulated as threads, executed in an HP Proliant composed by four processors AMD Opteron 6262 HE with a total of 64 cores and 128 GB of memory.
- Using the Spark Cluster mode executed in an Amazon Elastic MapReduce (EMR) 4.0.0 that deploys an Apache Hadoop NextGen MapReduce (YARN) system and uses m3.xlarge machines (Intel Xeon E5-2670 v2 with 4 cores and 15 GB of memory). For each dataset, a different configuration on the number of workers was used depending on the number of dataset partitions, assuring that there is at least one core for each dataset partition.

B. Statistical Analysis

In this section we first compare S-FRULER with FRULER [10] in terms of runtime of the full learning process in order to highlight the obtained speedup. Table II shows the runtimes of FRULER and S-FRULER and also shows the speedup obtained in each execution mode of S-FRULER (Standalone using threads or in a Cluster) when compared with FRULER and the number of dataset partitions n_{map} obtained for each dataset. It can be seen that the number of dataset partitions depends on the problem, but, in general, as the number of variable increases so does the number of dataset partitions used. S-FRULER, in both Standalone (threads) and Cluster, obtains lower runtimes than FRULER, as expected. The speedup depends on the problem, as datasets with a similar number of dataset partitions (e.g., DELAIL, DELELV, CAL and MV with 21-23 partitions) have a very different speedup (8, 15, 27 and 8 respectively in Standalone or 12, 21, 35 and 13 in Cluster mode). It is worth emphasizing that for the problems with the worst runtime in FRULER (HOU, ELV, PUM and AIL), S-FRULER achieves speedups far above the parallelization level given by the number of dataset partitions. This is because the scalability obtained by S-FRULER is not only given by the distributed approach, but also by the simplification of the convergence of FRULER in each dataset partition.

In order to evaluate the models learned by S-FRULER, we compared our approach with three of the most accurate genetic GFSs for regression —together with FRULER— in the literature:

- $\text{FS}_{\text{MOGFS}}^{\text{e}}$+$\text{TUN}^{\text{e}}$ [6]: a multi-objective evolutionary algorithm that learns Mamdani fuzzy rule bases. This algorithm learns the granularities from uniform multi-granularity fuzzy partitions (up to granularity 7) and the

algorithms	FRULER	S-FRULER	Standalone		Cluster	
	Time	n_{map}	Time	Speedup	Time	Speedup
DELAIL	0:09:58	21	0:01:18	8	0:00:48	12
DELELV	0:25:01	22	0:01:38	15	0:01:13	21
CAL	1:57:03	23	0:04:20	27	0:03:22	35
MV	1:17:02	23	0:09:27	8	0:05:49	13
HOU	4:15:17	25	0:04:06	62	0:03:09	81
ELV	3:01:30	26	0:03:10	57	0:03:14	56
CA	0:38:12	25	0:03:46	10	0:01:48	21
POLE	1:53:15	27	0:10:20	11	0:05:14	22
PUM	31:14:27	24	0:01:58	956	0:01:39	1,139
AIL	12:50:38	28	0:07:13	107	0:03:32	218

Table II: Runtime comparison between FRULER and S-FRULER. All times are formatted in hours:minutes:seconds.

Algorithms	S-FRULER	A-METSK-HD^e	FS$_{MOGFS}$^{e}+TUN^e	L-METSK-HD^e
DELAIL	1.44	**1.40**	1.53	1.62
DELELV	1.12	**1.03**	1.09	1.12
CAL	2.18	**1.71**	2.95	2.64
MV	**0.05**	0.06	0.16	0.25
HOU	**8.2**	8.5	9.4	10.4
ELV	**3.2**	7.0	9.0	8.9
CA	**4.6**	5.0	5.2	5.9
POLE	124	**61**	103	151
PUM	0.349	**0.287**	0.292	0.594
AIL	**1.4**	1.5	2.0	1.8
Friedman Ranking	1.8	1.5	3	3.7
Holm p-value		0.603	0.075	0.003

Table III: Average test errors for the different algorithms. The errors in this table should be multiplied by $10^{-8}, 10^{-6}, 10^{9}, 10^{8}, 10^{-6}, 10^{-4}, 10^{-8}$ in the case of DELAIL, DELELV, CAL, HOU, ELV, PUM, AIL respectively. The best results are marked in bold face. It also shows the Friedman Ranking for each approach and the Holm adjusted p-value comparing S-FRULER with the other algorithms.

lateral displacement of the labels. It includes a post-processing algorithm for tuning the parameters of the membership functions and for rule selection.

- L-METSK-HD^e [19]: a multi-objective evolutionary algorithm that learns linguistic TSK-0 fuzzy rule bases. The algorithm learns the granularities from uniform multi-granularity fuzzy partitions (up to granularity 7).
- A-METSK-HD^e [19]: a multi-objective evolutionary algorithm that learns approximative TSK-1 fuzzy rule bases. The algorithm starts with the solution obtained on the first stage and applies a tuning of the membership functions, rule selection and a Kalman-based calculation of the consequents of the rules.

Tables III and IV show the results obtained by S-FRULER and the other three GFSs for the datasets in Table I for precision — the mean test error over the executions — and complexity — the number of rules (# Rules) of the final solution. Moreover, in order to analyze statistically the difference between the different approaches, a Friedman ranking test followed by a Holm post-hoc test were performed for both measures. The Friedman ranking test calculates a rank for each approach based on its performance, where the lower the rank, the better. On the other hand, the Holm post-hoc method tests if the difference between two rankings is significant. We applied the Holm post-hoc test using the S-FRULER approach as a control method, thus calculating the p-value for each comparison of S-FRULER with each of the other approaches.

In the case of precision (Table III), both S-FRULER and A-METSK-HD^e achieve the best result in 5 of the datasets. A-METSK-HD^e obtains the lowest rank (1.5) followed by S-FRULER (1.8), however the difference between them is rather low, since the Holm p-value is high (0.603). FS$_{MOGFS}$^{e}+TUN^e and L-METSK-HD^e obtain worse errors than S-FRULER (ranks 3 and 3.7 respectively) and the differences are statistically significant (Holm p-values below 0.1).

In terms of complexity (Table IV), S-FRULER obtains the best result in 9 of 10 datasets, while FS$_{MOGFS}$^{e}+TUN^e obtains the best result in the remaining dataset (POLE). The Friedman ranking test shows these results, where S-FRULER obtains the lowest rank (1.1) followed by FS$_{MOGFS}$^{e}+TUN^e (1.9), while A-METSK-HD^e and L-METSK-HD^e obtain the worst ranking (3 and 4 respectively). Comparing S-FRULER with FS$_{MOGFS}$^{e}+TUN^e, the Holm p-value is rather low (0.166) showing a noticeable difference, while the difference with A-METSK-HD^e and L-METSK-HD^e is remarkable (p-value below 0.005).

These results show that S-FRULER not only reduces the runtimes of FRULER, but also obtains the simplest models in terms of number of rules, using a linguistic approach, while maintaining a high precision comparable to an approximative approach.

Algorithms	S-FRULER	A-METSK-HD^e	FS$_{MOGFS}$^{e}+TUN^e	L-METSK-HD^e
DELAIL	**3**	37	6	98
DELELV	**2**	39.1	7.9	91
CAL	**7**	56	8	100
MV	**4**	56	14	76
HOU	**11**	30	12	69
ELV	**5**	35	8	76
CA	**11**	32	14	71
POLE	20	46	**13**	100
PUM	**4**	63	18	88
AIL	**11**	48	15	99
Friedman Ranking	1.1	3	1.9	4
Holm p-value		0.002	0.166	$<$ 1E-3

Table IV: Average number of rules for the different algorithms. The best results are marked in bold face. It also shows the Friedman Ranking for each approach and the Holm adjusted p-value comparing S-FRULER with the other algorithms.

IV. Conclusions

In this paper, we have presented an scalable version of FRULER [10] —a GFS that learns simple and linguistic TSK-1 knowledge bases for regression problems—, called S-FRULER. S-FRULER obtains simple models with high precision but reducing the runtime of FRULER. The approach is based on a distributed computing methodology and was implemented using the Spark software, thus providing both standalone and cluster modes. S-FRULER partitions the dataset into small sets, which are more tractable, and executes FRULER for each of them. Also, for each dataset partition, a random feature selection process to reduce the number of variables is used. After the Map phase, the method implements an aggregation function to obtain linguistic TSK-1 fuzzy rule bases from the rule bases generated in each dataset partition.

S-FRULER has speedups usually larger than the number of dataset partitions used, showing a scalability higher than linear in both standalone and cluster modes. It was compared with three GFSs in terms of complexity of the models (number of rules) and precision (mean squared error), showing good results with less complex models.

References

[1] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009.

[2] N. García-Pedrajas and A. de Haro-García, "Scaling up data mining algorithms: review and taxonomy," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 71–87, 2012.

[3] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena, O. Cordon, F. Herrera, and F. Hoffmann, *Genetic fuzzy systems*. World Scientific Publishing Company Singapore, 2001.

[4] F. Herrera, "Genetic fuzzy systems: taxonomy, current research trends and prospects," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 27–46, 2008.

[5] A. Fernández, V. López, M. J. del Jesus, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Systems*, vol. 80, pp. 109–121, 2015.

[6] R. Alcalá, M. J. Gacto, and F. Herrera, "A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 666–681, 2011.

[7] M. Antonelli, P. Ducange, and F. Marcelloni, "An efficient multi-objective evolutionary fuzzy system for regression problems," *International Journal of Approximate Reasoning*, vol. 54, no. 9, pp. 1434–1451, 2013.

[8] R. Alcalá, J. Alcalá-Fdez, F. Herrera, and J. Otero, "Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation," *International Journal of Approximate Reasoning*, vol. 44, no. 1, pp. 45–64, 2007.

[9] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, "Reducing the complexity in genetic learning of accurate regression TSK rule-based systems," in *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Istanbul (Turkey), 2015.

[10] ——, "FRULER: Fuzzy rule learning through evolution for regression," *Information Sciences*, vol. 354, pp. 1–18, 2016.

[11] ——, "An instance selection algorithm for regression and its application in variance reduction," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2013.

[12] H. Ishibuchi and T. Yamamoto, "Performance evaluation of fuzzy partitions with different fuzzification grades," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE).*, vol. 2. IEEE, 2002, pp. 1198–1203.

[13] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," *International Journal of Intelligent Systems*, vol. 18, no. 3, pp. 309–338, 2003.

[14] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992.

[15] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society*, vol. 67, no. 2, pp. 301–320, 2005.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics*. Springer, 2010, pp. 177–186.

[17] Y. Tsuruoka, J. Tsujii, and S. Ananiadou, "Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty," in *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. Association for Computational Linguistics, 2009, pp. 477–485.

[18] J. Alcala-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[19] M. J. Gacto, M. Galende, R. Alcalá, and F. Herrera, "Metsk-hd e: A multiobjective evolutionary algorithm to learn accurate tsk-fuzzy systems in high-dimensional and large-scale regression problems," *Information Sciences*, vol. 276, pp. 63–79, 2014.