Information Sciences 276 (2014) 63–79

Contents lists available at ScienceDirect

Information Sciences

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

METSK-HD^e: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems ☆

M.J. Gacto [a,*], M. Galende [b], R. Alcalá [c], F. Herrera [c,d]

[a] Dept. of Computer Science, University of Jaén, 23071 Jaén, Spain
[b] CARTIF Centro Tecnológico, 47151 Boecillo, Valladolid, Spain
[c] Dept. of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain
[d] Faculty of Computing and Information Technology – North Jeddah, King Abdulaziz University, 21589 Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 11 August 2012
Received in revised form 20 January 2014
Accepted 9 February 2014
Available online 19 February 2014

Keywords:
Accurate fuzzy modeling
Takagi–Sugeno–Kang rules
Multi-objective genetic algorithms
Embedded genetic data base learning
Regression
High-dimensional and large-scale problems

ABSTRACT

In this contribution, we propose a two-stage method for Accurate Fuzzy Modeling in High-Dimensional Regression Problems using Approximate Takagi–Sugeno–Kang Fuzzy Rule-Based Systems. In the first stage, an evolutionary data base learning is performed (involving variables, granularities and slight fuzzy partition displacements) together with an inductive rule base learning within the same process. The second stage is a post-processing process to perform a rule selection and a scatter-based tuning of the membership functions for further refinement of the learned solutions. Moreover, the second stage incorporates an efficient Kalman filter to learn the coefficients of the consequent polynomial function in the Takagi–Sugeno–Kang rules. Both stages include mechanisms that significantly improve the accuracy of the model and ensure a fast convergence in high-dimensional and large-scale regression datasets.

We tested our approach on 28 real-world datasets with different numbers of variables and instances. Five well-known methods have been executed as references. We compared the different approaches by applying non-parametric statistical tests for pair-wise and multiple comparisons. The results confirm the effectiveness of the proposed method, showing better results in accuracy within a reasonable computing time.

1. Introduction

In Takagi–Sugeno–Kang (TSK) Fuzzy Rule-Based Systems (FRBSs) [42,43], the rule structure is formed by linguistic variables in the antecedent and a polynomial function of the input variables in the consequent. This rule structure involves the loss of interpretability [22] to some degree with respect to linguistic (Mamdani) FRBSs [36,35], although it allows the model to be much more accurate. For this reason, TSK FRBSs have been successfully applied to regression and control problems [30,28] with the main aim of obtaining highly accurate approximators [41,7]. The learning of premises and consequents

☆ Supported by the Andalusian Government under Grant no. P10-TIC-6858 and the Spanish Ministry of Science and Innovation under Grants nos. TIN2011-28488, DPI2012-39381-C02-02 and TIN2012-33856.

* Corresponding author. Tel.: +34 953212261.
E-mail addresses: mgacto@ujaen.es (M.J. Gacto), margal@cartif.es (M. Galende), alcala@decsai.ugr.es (R. Alcalá), herrera@decsai.ugr.es (F. Herrera).

http://dx.doi.org/10.1016/j.ins.2014.02.047
0020-0255/

of TSK FRBSs is usually carried out in different stages, or alternately, due to the high complexity of the search space involved. However, both parts should be obtained together within the same process, since they are dependent on each other. Evolutionary Algorithms (EAs) are able to learn the antecedents and consequents of the TSK rules together. This widely recognized hybridization of fuzzy systems and EAs is known as Evolutionary or Genetic Fuzzy Systems (GFSs) [26]. However, they usually have scalability problems in terms of computational time and convergence in datasets with a high number of variables and/or with a large amount of data.

In recent years, having to deal with large or high-dimensional datasets has become more common, and remains an open topic in the GFSs framework [26,17], which has led to some emergent approaches such as data pre-processing [14], data information granulation [38], etc. Large datasets include many instances, while high-dimensional datasets refer to datasets with a high number of variables [31]. These kinds of datasets represent a challenge for GFSs: the size of large datasets affects the fitness function computation, thus increasing the computational time, whereas high-dimensional datasets increase the search space. Moreover, in most cases, the wider the search space the greater the number of rules generated. Resulting models can be very complex, including hundreds of rules [44], which increases their evaluation computational time still further and usually leads to overfitting (especially when dealing with potentially highly accurate systems such as TSK-type FRBSs).

Therefore, even if accuracy is the main single objective, tackling large or high-dimensional datasets not only involves dealing with many variables and/or instances but also controlling the complexity of the obtained models. Since it is difficult to find this optimal trade-off (the simplest structure presenting near optimal accuracy), Multi-Objective Evolutionary Algorithms (MOEAs) [12,10] is an interesting tool with which to find the best compromise by focussing on the most accurate solutions while avoiding those that are too complex [20]. This combination of fuzzy systems and MOEAs is an important branch of GFSs known as Multi-Objective Evolutionary Fuzzy Systems (MOEFSs) [17]. Even though some MOEFSs have been applied to the learning of TSK FRBSs [24,25], most of these cases were related to improving their interpretability in regard to synthetic or very simple problems with a small number of variables and instances (see [17] for a complete review of these kinds of systems).

Only a few MOEFSs have been specifically proposed to tackle high and/or large datasets. The first two approaches [3,6] are devoted to a fast derivation of Mamdani linguistic rules including some mechanisms to control dimensionality and the appropriate use of a reduced set of examples or instance selection (training set selection) respectively. However, the search space for learning TSK rules is more complicated than in the case of Mamdani rules due to the complexity involved in obtaining the consequent linear parameters. In fact, one of the few approaches devoted to tackling scalability in the derivation of TSK FRBSs [9] is devoted to a speed-up of the Kalman filter [33], which obtains the consequent coefficients, in order to integrate it in an MOEA. This is based on decoupling the rules in order to only re-estimate the coefficients of the added or modified rules when they apply mutation operators. While this is not applicable to global learning (a global modification of the rules carried out by a tuning of the antecedents, etc.) this is an interesting approach that was applied to several synthetic problems with up to 10 variables and 70,000 examples, using user predefinition of the fuzzy partitions (uniform partitions with a given number of fuzzy sets per variable fixed by hand for each particular dataset). Finally, another recent trend is the use of parallel computation [32,18] in order to share the computing load among different CPUs. However, we will focus on the design of improved sequential algorithms as these kinds of mechanisms could easily be integrated to any evolutionary approach to parallel fitness computation, improving to an even greater extent those sequential algorithms that were specifically designed for big datasets.

In this contribution, we present a scalable two-stage MOEFS for accurate fuzzy modeling through learning the global structure of TSK FRBSs, namely METSK-HD^e (Multiobjective Evolutionary learning of TSK systems for High Dimensional problems with *estimated* error). The first stage is based on the adaptation of some components from [3] (devoted to learning Mamdani-type rules) in order to perform a fast identification of the most accurate TSK zero-order candidate structure, by automatic learning of the appropriate fuzzy partitions and candidate rules. This includes a new objective to control overfitting and a new rule generation method to obtain zero-order TSK consequents. However, the second stage includes the main novelties considering a completely different coding scheme (which involves a much larger search space) and a new way of efficiently integrating the Kalman filter while performing a global scatter-based tuning of the whole TSK FRBSs (evolving antecedents and consequents together). In both stages, we will also consider the use of MOEAs as a tool to control the complexity of the models and system overfitting, but with the main global objective of obtaining accurate models. In this way, even though our first and main objective is *minimizing the system error*, two additional objectives have been considered in both stages: *minimizing the number of rules* and *maximizing the medium coverage degree of the examples*.

In order to do this, in the first stage we present an effective Multi-Objective Evolutionary Algorithm (MOEA) [12,10], based on an embedded genetic Data Base (DB) learning [26] (involving variables, granularities and a slight lateral displacement [2] of fuzzy partitions). The Rule Base (RB) is obtained within the same process using a new efficient *ad hoc* algorithm that also estimates the coefficients of the TSK zero-order consequents. The proposed MOEA includes some specific mechanisms to ensure a fast learning of TSK FRBSs in order to obtain and fix a candidate model structure but preventing a premature convergence in problems with a high number of variables and examples. The second stage represents a new post-processing process based on a second MOEA that performs a rule selection and a fine scatter-based tuning of the Membership Functions (MFs). Furthermore, it incorporates a new efficient hybridization of a Kalman filter [33] and the proposed MOEA to estimate the coefficients of the consequent polynomial functions together with the antecedent parameters of the TSK rules, which helps to significantly improve the accuracy of the model.

We tested our approach on 28 real-world regression datasets with a number of variables ranging from 2 to 40 and a number of samples ranging from 337 to 40,768. Where possible, depending on the dimensionality, we executed four well-known accuracy-driven single-objective methods in order to obtain some good performance references. Further, as some components of the proposed approach are based on certain ideas from [3], we have also performed an internal comparison in order to show the differences between the new scalable method for precise scatter-based modeling and the previous scalable MOEFS for linguistic fuzzy modeling [3]. To assess the results obtained by the different algorithms, we have applied non-parametric statistical tests for pair-wise and multiple comparisons, considering for the MOEAs the average of the most accurate solution from each Pareto front. The results obtained demonstrate the effectiveness of the proposed method, particularly in terms of accuracy when dealing with high-dimensional and large-scale datasets. The method proposed obtained the most accurate results with significant statistical differences within a reasonable computing time.

This article is organized as follows. The next section describes the general TSK fuzzy model structure considered in this work. Section 3 presents the proposed method describing its main characteristics and the genetic operators considered. Section 4 shows the experimental study and the results obtained. Finally, in Section 5 we draw some conclusions.

2. Takagi–Sugeno–Kang fuzzy rule-based systems

In [42,43], Takagi and Sugeno proposed a fuzzy model based on rules in which the antecedents are comprised of linguistic variables as in the case of Mamdani FRBSs [36,35]. The principal distinguishing feature of this kind of model is that it is based on rules in which the consequent is not a linguistic variable but a function of the input variables. These kinds of rules present the following structure:

$$\begin{aligned} &\textit{If } X_1 \textit{ is } A_1 \textit{ and } \ldots \textit{ and } X_n \textit{ is } A_n \textit{ then} \\ &\quad Y = p_1 \cdot X_1 + \cdots + p_n \cdot X_n + p_0, \end{aligned}$$

where X_i are the system input variables, Y is the system output variable, p_i are real-values coefficients and A_i are fuzzy sets. Such rules are called *TSK fuzzy rules*, in reference to their creators [43].

The output of a TSK FRBS considering a Knowledge Base (KB) composed of m TSK rules is computed as the weighted average of the individual rule output $Y_i, i = 1 \ldots m$:

$$\frac{\sum_{i=1}^{m} h_i \cdot Y_i}{\sum_{i=1}^{m} h_i},$$

with $h_i = T(A_1(x_1), \ldots, A_n(x_n))$ being the matching degree between the antecedent part of the ith rule and the current system inputs $x = (x_1, \ldots, x_n)$, and with T being a t-norm.

TSK FRBSs have been applied successfully to a large quantity of problems. The main advantage of these kinds of systems is the fact that they present a compact system equation for estimating the parameters p_i using classical methods, and obtaining an accurate system, which can be very useful for accurate fuzzy modeling.

On the other hand, instead of considering linguistic partitions, scatter partitions could be considered. The scatter approach is based on rules presenting the following structure:

$$R_i : \textit{If } X_1 \textit{ is } A_{i1} \textit{ and } \ldots \textit{ and } X_n \textit{ is } A_{in} \textit{ then } Y \textit{ is } B_i$$

where A_i and B_i are fuzzy sets specific to each fuzzy rule. Approaches based on scatter partitions present interesting advantages that make them very suitable for precise modeling purposes:

- The expressive power of the rules that present their own specificity in terms of the fuzzy sets involved in them, thus introducing additional degrees of freedom in the system.
- The number of rules is adapted to the complexity of the problem, requiring fewer rules in simple problems, and being able to use more rules if necessary. This is likely to be of benefit in tackling the curse of dimensionality when scaling to multidimensional systems.

In this article we focus on developing accurate TSK fuzzy models based on scatter partitions, which can provide more accurate solutions to different problems, especially real-world high-dimensional and large-scale regression problems which have accuracy as the main requirement of their solution.

3. A method for evolutionary learning of scatter-based TSK FRBSs in high dimensional and large-scale problems

This section presents the proposed two stage method for regression problems with a high number of variables and/or examples. In the first stage, an effective MOEA is applied in order to learn an initial DB, based on a fuzzy grid in order to obtain zero-order TSK candidate rules, while the second stage applies an advanced post-processing MOEA for fine scatter-based evolutionary tuning of MFs combined with a rule selection (see Fig. 1). These algorithms incorporate some of the ideas of the fast and scalable multi-objective genetic fuzzy system, FS$_{\text{MOGFS}}^{e}$ [3], for linguistic fuzzy modeling in complex regression problems.

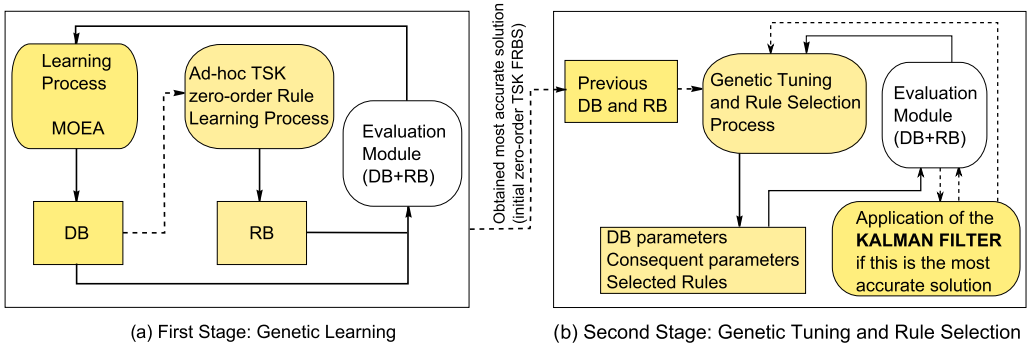

Fig. 1. Learning scheme of the proposed two stage algorithm.

In the following, we include a preliminary section describing a mechanism for error estimation in large-scale problems [3] and an adaptation of the Wang and Mendel [45] method (WM) for obtaining zero-order TSK rules. Then, Sections 3.2 and 3.3 present both stages of the proposed method.

3.1. Preliminaries: mechanisms integrated in the MOEAs

In this section, we present two mechanisms used in the proposed algorithm. The first one is an error estimation mechanism used in both stages of the algorithm. This mechanism avoids using a high percentage of the examples for error computation, estimating it from a reduced subset of the examples. The second one is used only in the first stage to derive a set of TSK zero-order rules as the RB generation process.

3.1.1. Mechanism for error estimation: partial error computation on large-scale datasets

In order to handle the scalability problem in datasets with a large amount of data, we propose the use of a new mechanism presented in [3] for fast error computation in large-scale datasets. This procedure is based on taking a small percentage of the training examples to estimate the error of the newly generated solutions. Once these errors are estimated, only those solutions in the elite set (non-dominated solutions) are evaluated, considering the whole set of examples E.

The subset of examples E^e for error estimation is obtained by a random selection of $\lfloor r^e * m \rfloor$ examples in each generation. Where r^e is the percentage of samples used to estimate the error and m is the dataset size. If $\lfloor r^e * m \rfloor \geqslant 1000$ then $r^e = 1000/m$, i.e., no more than 1000 examples will be considered as this was shown to be good enough when compared to the use of all the examples (see [3] on the use or not of the error estimation mechanism). E^e is fixed for a generation. After each generation the examples are replaced by a random selection from those examples that were not used in the previous generation. In this way, we promote a rotation of the selected examples.

For each new solution to be evaluated we compute its error in E^e (error estimation). If by taking into account the estimated error and the current non-dominated solutions the new individual represents a new non-dominated solution, we perform a complete evaluation by considering the estimated error and the examples in $E - E^e$. In this way, the Pareto set will always contain solutions evaluated considering 100% of the examples. See Fig. 2 for a scheme of this mechanism.

3.1.2. Method for an effective generation of TSK candidate rules

We apply an adaptation of the WM method [45] in order to obtain a whole KB from a given DB (a given set of linguistic terms and their associated MFs definitions). In contrast to WM, the consequents of TSK rules are obtained, with all the

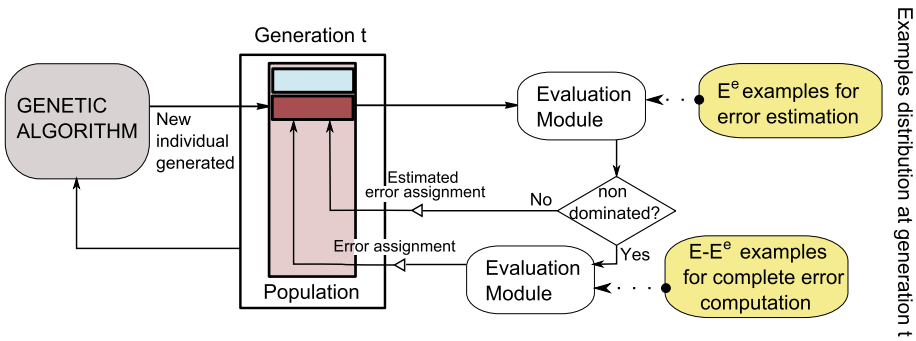

Fig. 2. Flowchart scheme of the error estimation mechanism for a new individual generated by the genetic algorithm.

coefficients with value 0 and the independent terms computed as the average of the examples covered by the rule weighted by their matching.

However, in problems with a high number of variables and/or examples this method can take a long time to derive thousands of rules. To avoid this situation, once it is integrated within the MOEA of our first stage a cropping criterion has been added to the method. In this way, the method stops the process if the RB reaches a limit of 100 rules and marks the RB as incomplete. We propose a maximum number of 100 rules for the rule cropping mechanism based on some empirical trials, which showed no significant differences in models obtained with more rules. Higher values or even those that do not use cropping do not obtain significantly more accurate solutions. In fact, the final number of rules obtained by the proposed algorithm in 28 real-world problems is always under 70. To penalize incomplete solutions (which should disappear during the evolution of the first stage MOEA), we estimate the number of rules as the product of the number of labels of the input variables and in the case of the Mean Squared Error it is penalized with a fixed large error.

3.2. First stage: an effective MOEA for the initial KB learning

The proposed MOEA is based on embedded genetic DB learning [26] (using variables, granularities and lateral displacements of fuzzy partitions [2]) which enables the structure of the initial TSK FRBSs to be learned fast, reducing its dimensionality and making use of some effective mechanisms in order to ensure a fast convergence in high-dimensional and large-scale regression datasets. Embedded genetic DB learning is based on an evolutionary process that codes and evolves different DBs which are evaluated by applying a fast inductive RB generation method and computing the error of the thus obtained FRBS (see Fig. 1a for a scheme of this kind of approach).

The following subsections describe the main features of the proposed algorithm: coding scheme, objectives, initial population, crossover and mutation operators, incest prevention mechanism and stopping condition.

3.2.1. DB codification

In order to improve the performance and to decrease the complexity of the classic tuning approaches in complex search spaces, an effective tuning model has recently been proposed for FRBSs in [2] considering the linguistic 2-tuples representation scheme [37]. The linguistic 2-tuples representation allows the lateral displacement of the MFs by considering only one parameter (slight displacements to the left/right of the original MFs). Since the three parameters usually considered per label are reduced to only one symbolic translation parameter, this proposal decreases the learning problem complexity, helping to decrease the model error and facilitating a significant decrease in the model complexity. It was extended in [3] in order to use only one parameter per variable so that the same displacement is applied for all the corresponding MFs. See Fig. 3 for an example of this kind of representation. We will use this latter approach as a way of allowing a slight efficient tuning while learning the DB.

In this way, a double coding scheme ($C = C_G + C_L$) to represent both parts, *granularity* and *translation parameters*, is considered:

- Number of labels (C_G): This part is a vector of integer numbers of size N (with N representing the number of input variables) in which the granularities of the different variables are coded,

$$C_G = (L^1, \ldots, L^N).$$

Each gene L^i represents the number of labels used by the ith variable and takes values in the set $\{2, \ldots, 7\}$. Additionally, it may take a value equal to 1 to determine that the corresponding variable is not used.

- Lateral displacements (C_L): This part is a vector of real numbers of size N in which the displacements of the different variables are coded [2]. In this way, the C_L part has the following structure (in which each gene is the displacement value of the fuzzy partition of the corresponding linguistic variable and takes values from $[-0.1, 0.1]$),

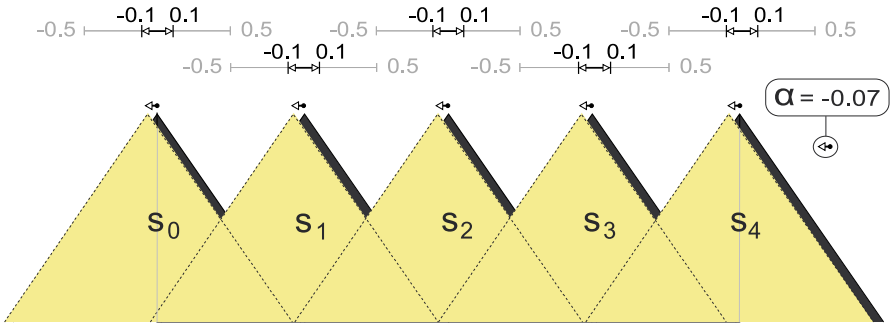

Fig. 3. Slight lateral displacement in $[-0.1, 0.1]$.

$$C_L = (\alpha^1, \ldots, \alpha^N).$$

3.2.2. Objectives

In order to evaluate a given individual, the adaptation of the WM method (see Section 3.1.2) is applied to the associated DB in order to obtain the corresponding RB. Once a complete KB is obtained the following objectives are calculated:

1. Minimize the Mean Squared Error (*MSE*), which is our main objective:

$$MSE = \frac{1}{2 \cdot |E|} \sum_{l=1}^{|E|} (F(x^l) - y^l)^2,$$

with $|E|$ being the dataset size, $F(x^l)$ being the output obtained from the FRBS decoded from a given chromosome when the lth example is considered and y^l being the known desired output.
2. Minimize the Number of Rules (*NR*), to control the complexity and overfitting.
3. Maximize the medium coverage degree of the examples, to help control overfitting.

3.2.3. Initial gene pool

The initial population will be comprised of two different subsets of individuals:

- In the first subset, each chromosome has the same number of labels for all the system input variables. In order to provide diversity in the C_G part, these solutions have been generated by considering all the possible combinations of the input variables, i.e., from 2 labels to 7 labels. Additionally, for each of these combinations two copies are included with different values in the C_L part. The first one with random values in $[-0.1, 0.0]$ and the second one with random values in $[0.0, 0.1]$. If there is no space for these solutions, they are included from the smallest granularities (the most interesting combinations in principle) to the highest possible ones.
- In the second subset, we generate random solutions in order to completely fill the population (values in $\{2, \ldots, 7\}$ for C_G and values in $[-0.1, 0.1]$ for C_L).

Finally, except in the cases of problems with less than three input variables, an input variable v is removed at random, $L^v = 1$. This action is repeated until no more than 5 variables remain in all the individuals. This process is applied to all the individuals in the population in order to avoid the generation of solutions that make no sense (because of the exorbitant number of rules).

3.2.4. Crossover and mutation operators

The crossover operator depends on the part of the chromosome to which it is applied. A crossover point is randomly generated and the classical crossover operator is applied to this point for the C_G part. The Parent Centric BLX (PCBLX) operator [34], which is based on BLX-α, is applied to the C_L part.

In this way, four new individuals are obtained by combining the two offspring generated from C_G with the two offspring generated from C_L. For each of them, the mutation operator is applied with probability P_m. The mutation operator decreases by 1 the granularity in a gene g selected at random ($L^g = L^g - 1$) or randomly determines a higher granularity in $\{L^g + 1, \ldots, 7\}$ with the same probability. No decreasing is performed when it provokes DBs with only one input variable. The same gene is also changed at random in C_L. Finally, after considering mutation, only the two most accurate individuals are taken as descendants.

3.2.5. Incest prevention and stopping condition

An incest prevention mechanism has been included in the C_L parts by following the concepts of CHC [15], to maintain the population diversity and avoid premature convergence. Only parents whose hamming distance divided by 4 is greater than a threshold is crossed. Because it uses a real encoding scheme in C_L, each gene is transformed into gray code with a fixed number of bits per gene (*BGenes*). This threshold value is initialized as follows: $L = (\#GenesC_L * BGene)/4$, where $\#GenesC_L$ is the number of genes in the C_L part. The algorithm ends when a maximum number of evaluations are reached or when L is below zero.

3.3. Second stage: an advanced post-processing MOEA to perform rule selection, fine tuning of MFs and efficient least-squares-based consequent coefficients adjustment

Once a complete zero-order TSK KB is obtained in the first stage, a post-processing MOEA is applied to perform a tuning of MFs and a rule selection, which will help to significantly improve the accuracy. To this end, we present a new MOEA for accurate TSK FRBSs tuning and rule selection based on a previous MOEA, namely SPEA2$_{E/E}$ [21]. The new proposed MOEA includes the error estimation procedure, described in Section 3.1.1. Further, a least-squares-based iterative mechanism

has been integrated to allow consequent parameters adaptation according to the system evolution (see Fig. 1b for a scheme of this kind of approach).

The following subsections describe the main components of the post-processing MOEA.

3.3.1. Coding scheme, objectives and initial default rule generation

A triple coding scheme for *classical tuning* (C_T), *rule selection* (C_S) and *coefficients of the consequents* (C_C) is used: $C = C_T + C_S + C_C$

- Tuning of MFs (C_T): Since this stage performs a scatter-based fine tuning, in this part a real coding is used where we consider the parameters of all the MFs per rule individually,

$$C_i = (\ldots, a_1^i, b_1^i, c_1^i, \ldots, a_{N'}^i, b_{N'}^i, c_{N'}^i, \ldots), \quad i = 1, \ldots, m,$$

  with a_j^i, b_j^i and c_j^i being the definition points of the jth MF of the ith rule, with N' being the number of input variables determined in the first stage and with m being the number of initial rules.
- Rule selection (C_S): consists of binary-coded strings with size m. Depending on whether a rule is selected or not, values '1' or '0' are respectively assigned to the corresponding gene.
- Coefficients of the consequents (C_C): This is a vector of real numbers of size $(N' + 1) * m$ in which the coefficients of the consequent polynomial function for each TSK rule are encoded,

$$C_C = (\ldots, p_1^i, \ldots, p_{N'}^i, p_0, \ldots), i = 1, \ldots, m.$$

This stage of the algorithm considers the same three objectives presented in Section 3.2.2. However, since this time we are performing a rule selection, in order to ensure a complete covering of the training examples, we apply a penalization of the *MSE* value if any training example is not covered by any rule. In this case, once we compute the *MSE* associated with this undesired solution we add to it the *MSE* of the initial solution as a penalization. This ensures that the most accurate solution through evolution always covers all the training examples.

In any case, based on the initial zero-order TSK KB obtained in the first stage, a default general rule (to be used in case of uncovered data) is initially generated by applying the standard Kalman filter (10 iterations) and by taking into account examples whose coverage by the initial KB is under 0.2 (which are close to uncovered regions). In any event, at least two examples, those with the least degree of covering, are taken into account. Since this is a default rule, its activation degree (matching) is fixed to 1.0 for all the examples selected in order to apply the filter. The rule obtained will be applied each time a given input is not covered by any of the rules when we are evaluating a new individual (*MSE* computation). Since we ensure the covering of all the training examples, the real aim of this rule is to provide a reasonable output for new uncovered data from real systems or test data applications.

3.3.2. Initial gene pool

The initial population is obtained with all individuals having all genes with value '1' in C_S. In the C_T part, the initial DB is included as an initial solution and the remaining individuals are randomly generated, maintaining their values within their respective variation intervals. The variation intervals for each of the three definition points of each MF are fixed in the following way from the initial corresponding MF (see Fig. 4):

$$\begin{aligned}
[I_{a_j}^l, I_{a_j}^r] &= [a_j - (b_j - a_j)/2, a_j + (b_j - a_j)/2], \\
[I_{b_j}^l, I_{b_j}^r] &= [b_j - (b_j - a_j)/2, b_j + (c_j - b_j)/2], \\
[I_{c_j}^l, I_{c_j}^r] &= [c_j - (c_j - b_j)/2, c_j + (c_j - b_j)/2].
\end{aligned} \tag{1}$$

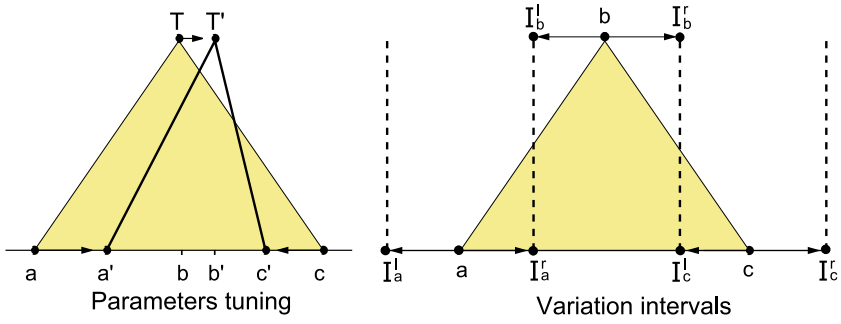

Fig. 4. Tuning by changing the basic MF parameters and corresponding variation intervals.

Finally, the C_C part of the first individual includes the consequents obtained in the first stage. Then, we apply the standard Kalman filter to the initial individual (1 iteration) in order to obtain estimated initial coefficients. The remaining individuals are initialized with these same coefficients. We do not use the Kalman filter to obtain the coefficients for all individuals, because it would significantly increase the computational time of the algorithm. In the next subsections, we will present an efficient way of integrating the Kalman filter in order to also apply it through evolution.

3.3.3. Crossover and mutation operators

An incest prevention mechanism has been included by following the concepts of CHC [15] and by only taking into account the C_T parts. Following the original CHC scheme (for binary coding), two parents are crossed if their hamming distance divided by 4 is over a predetermined threshold, L (see formulation in Section 3.2.5).

The BLX-0.5 [16] crossover is applied to obtain the C_T part of the offspring. The binary part C_S is obtained based on the C_T parts (MF parameters) of the corresponding parents and offspring [20,21]. For each gene in the C_S part which represents a concrete rule:

1. The displacement parameters of the MFs involved in such rules are extracted from the corresponding C_T parts for each individual involved in the crossover (offspring and parents 1 and 2). These displacements represent the specific differences between these three individuals for such rules.
2. Euclidean normalized distances are computed between the offspring rule and each parent rule by considering the center points (vertex) of the MFs comprising such rules. The differences between each pair of centers are normalized by the amplitudes of their respective variation intervals.
3. The parent with the closest distance to the offspring is the one that determines whether this rule is selected or not for the offspring by directly copying its value in C_S for the corresponding gene.

The C_C part is obtained by directly copying its values from the parent with the closest distance in C_S to the offspring. In this way, the coefficients are only inherited from the closest parent as they will in fact be mainly learned through the integrated efficient Kalman filter proposed in the following section.

The mutation operator is only applied in the C_S part and this favors rule extraction since mutation is only engaged to remove rules.

3.3.4. Efficient application of the Kalman filter

The Kalman filter [33] is a classic technique to estimate the coefficients of the consequent polynomial function in the TSK rules. This technique obtains good results in training, but usually presents overfitting in test. To avoid this undesired situation, only a small percentage of samples (the same examples used to estimate errors, see Section 3.1.1) is used to obtain the coefficients of the TSK rules.

Once a new solution is generated by crossover and mutation, and evaluated using the small percentage of examples, if the estimated error is the best known error, which would make it non-dominated and therefore a candidate to be evaluated in the whole set of examples, the Kalman filter is applied to the same subset of examples, E^e, to obtain the corresponding consequent parameters before the evaluation as a whole is undertaken. This way of working provides a validation mechanism for the obtained coefficients since they should also work when using the examples that were not used by the Kalman filter.

We do not apply the Kalman filter to obtain the coefficients for all individuals, since this would significantly increase the computational time of the algorithm. Further, in order to save more time and in order to make the coefficients converge with the MFs and the selected rules, only one iteration of the Kalman filter is run each time. Thus, the Kalman filter is only initialized at the beginning of the algorithm and each time restarting is applied, so that the coefficients are progressively improved for those combinations of MFs and rules that continuously promote new, more accurate solutions. This is possibly due to the kind of process used (post-processing), which does not change the system structure (the same Kalman parameters can be maintained from one solution to another), and the fact that not selected rules are not activated by examples (matching 0) to apply the filter. As the subset of examples E^e changes randomly at each new generation, we would like to point out that all the examples are finally seen by the Kalman filter, but the strong overfitting that sometimes appears when they are considered together is avoided.

See Fig. 5 for a flowchart scheme of the Kalman filter application integrated with the error estimation mechanism once a new individual has been generated by the evolutionary algorithm. Dashed lines represent the additional steps for this efficient application of the Kalman filter.

3.3.5. Restarting

This mechanism is applied when the threshold value L is below zero. Once restarting is applied, L is set to its initial value (see Section 3.3.3). The restarting operator is applied by only copying the best individual for each of the three objectives as the first three individuals of a new initial population. The external population is then set to empty. The remaining individuals of this initial population copy the values of the most accurate individual for the C_S part and take values generated at random in the C_T part. In order to assign good candidate values to the C_C part of these new individuals and to avoid an unnecessary particular computation for each of them, the Standard Kalman filter is only applied to the most accurate individual and the obtained parameters are copied in the C_C part for all these new individuals. Additionally, it regenerates the default rule as

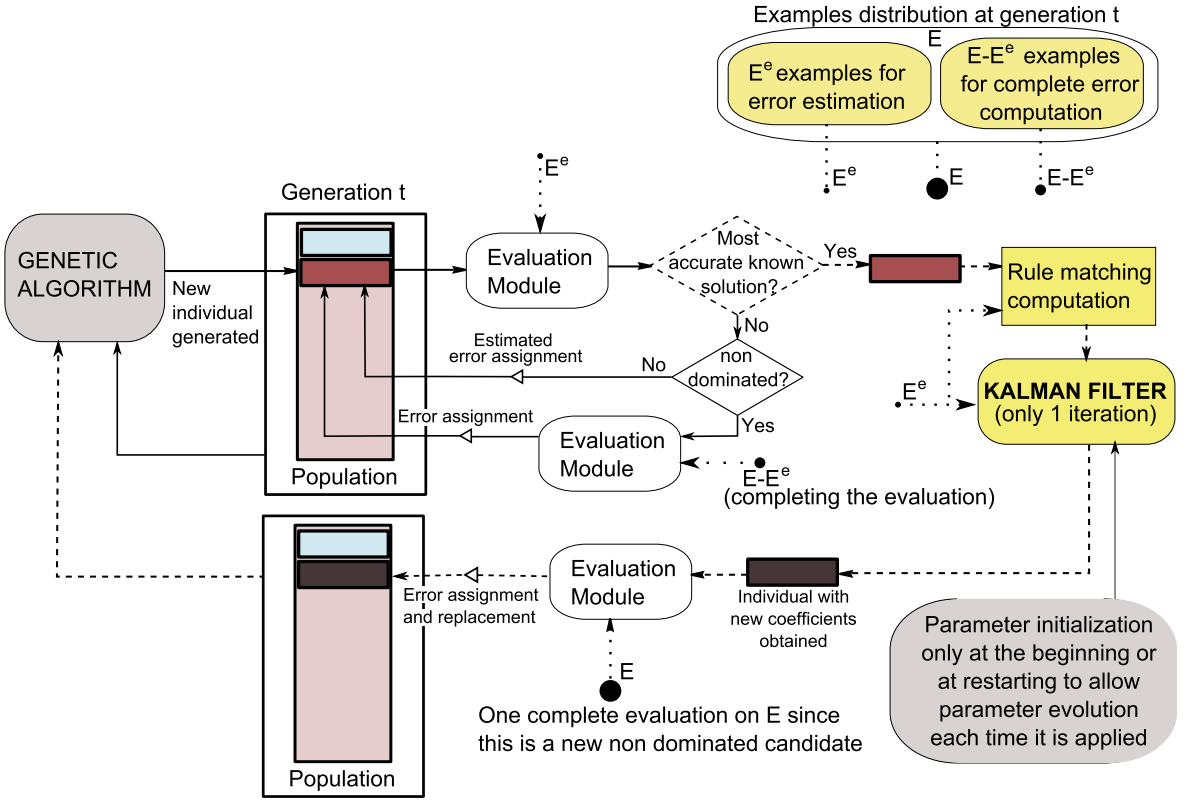

Fig. 5. Flowchart scheme of the Kalman filter integration together with the error estimation once a new individual has been generated by the evolutionary algorithm. Dashed lines represent the part related to the filter application while dotted lines represent the use of the example subsets.

explained in Section 3.3.1, but taking into account examples whose coverage by the current most accurate individual is under 0.2 (which are close to current uncovered regions).

In each stage of the algorithm (between restarting points), the number of solutions in the external population considered to form the mating pool is progressively reduced, by focusing only on those with the best accuracy. To do that, the solutions are sorted from the best to the worst (considering accuracy as criterion) and the number of solutions considered for selection is reduced progressively from 100% at the beginning to 50% at the end of each stage. This helps to focus the search on the desired Pareto zone (see [20]), highly accurate solutions with the least possible number of rules, giving more selective pressure to those solutions that have a high accuracy (crossing dissimilar solutions in principle and similar ones at the end). It is done by taking into account the value of L. In the last evaluations when restart is disabled, this mechanism for focusing on the most accurate solutions (the most difficult objective), is also disabled in order to obtain a wide, well-formed Pareto front, from the most accurate solutions to the most interpretable ones.

4. Experiments and analysis of results

In order to evaluate the usefulness of the proposed approach, namely METSK-HD^e (Multiobjective Evolutionary learning of TSK systems for High Dimensional problems with *estimated* error), in high-dimensional and large-scale regression datasets, we have used 28 real-world problems with different numbers of variables and cases. Table 1 sums up the main characteristics of the different problems considered in this study and shows the link to the KEEL project webpage [5,4] from which they can be downloaded. These problems have been selected from minor to major complexity, covering a range from 2 to 40 input variables and from 337 to 40,768 examples (even though each is a complicated problem in itself in terms of the modeling task). The most complex problems are MV, HOU, ELV, CA, POLE, PUM and AIL because of the large number of variables and data. These problems represent an important challenge for this algorithm. This is due to the long time needed to evaluate an individual and to the minimum number of evaluations needed to reach convergence.

This section is organized as follows:

- First, we describe the experimental set-up, datasets and methods considered in this article (Section 4.1).
- Second, we perform an internal comparison in order to show the differences between the new scalable method for precise scatter-based modeling and the previous scalable method for linguistic modeling [3] (Section 4.2).

Table 1
Datasets considered for the experimental study.

Problem	Abbr.	Variables	Cases
Electrical Length	ELE1	2	495
Plastic Strength	PLA	2	1650
Quake	QUA	3	2178
Electrical Maintenance	ELE2	4	1056
Friedman	FRIE	5	1200
Auto MPG6	MPG6	5	398
Delta Ailerons	DELAIL	5	7129
Daily Electricity Energy	DEE	6	365
Delta Elevators	DELELV	6	9517
Analcat	ANA	7	4052
Auto MPG8	MPG8	7	398
Abalone	ABA	8	4177
California Housing	CAL	8	20,640
Concrete Compressive Strength	CON	8	1030
Stock prices	STP	9	950
Weather Ankara	WAN	9	1609
Weather Izmir	WIZ	9	1461
MV Artificial Domain	MV	10	40,768
Forest Fires	FOR	12	517
Mortgage	MOR	15	1049
Treasury	TRE	15	1049
Baseball	BAS	16	337
House-16H	HOU	16	22,784
Elevators	ELV	18	16,559
Computer Activity	CA	21	8192
Pole Telecommunications	POLE	26	14,998
Pumadyn	PUM	32	8192
Ailerons	AIL	40	13,750

Available at: http://www.keel.es/.

- Third, we compare the solutions of our proposed method with respect to four well-known accuracy-driven methods in Section 4.3.
- Finally, we show the computational costs of the different algorithms and we discuss the scalability of the proposed approach in Section 4.4.

4.1. Experimental set-up

To evaluate the effectiveness of the proposed method designed for high dimensional and large-scale regression problems, several well-known accuracy-driven methods have been included for comparison. Four different methods for precise modeling have been considered: The classical method (ANFIS), one statistical regression method and two related evolutionary methods.

The first one, ANFIS [30] is a neural FRBS to obtain global semantics-based TSK FRBSs. This classical method obtains very accurate FRBSs, thanks to gradient descent and least squares estimation mechanisms. However, it was only possible to run its classical version (based on grid partitioning) on the first eight datasets, obtaining in all cases worse results than those obtained by the proposed algorithm. A better alternative is provided by using ANFIS with the technique of subtractive clustering [8] found in the Fuzzy Logic Toolbox of Matlab instead of grid-partitioning, which is known to be preferred to grid partitioning in high dimensional datasets. We will call this approach ANFIS-SUB. ANFIS-SUB applies the subtractive clustering method to obtain an initial TSK-based FRBS that is tuned by ANFIS. We have considered this version for comparison. The second one, LINEAR-LMS [39] is a classical statistical regression method based on gradient techniques, which obtains a regression model as a result of a linear combination of its features. The weights of such combinations are fitted as a linear discriminant using Least Mean Squares. In a broad sense, it can be considered as an effective LMS-based approach for precise modeling with local linear models. The third one, TSK-IRL [11] is an evolutionary method based on MOGUL (a methodology to obtain Genetic FRBSs under the Iterative Rule Learning approach) which combines an inductive algorithm and a (μ, λ) evolution-strategy. This enables the automatic generation of a preliminary TSK-type KB for a concrete problem which is tuned in a second evolutionary stage. The fourth one, LEL-TSK [1] obtains accurate local semantics-based TSK rules. This two-stage evolutionary algorithm, also based on MOGUL, has been developed to consider the interaction between input and output variables.

Additionally, the $\text{FS}_{\text{MOGFS}}^{e}$ + TUN^{e} [3] method is used for internal comparison. This method is a Fast and Scalable Multi-Objective Genetic Fuzzy System for Linguistic Fuzzy Modeling in High-Dimensional Regression Problems and is only considered in order to show the higher accuracy of the proposed approach and the differences between both algorithms (since they have some common operators) and between both types of modeling.

Table 2
Methods considered for the experimental study.

Ref.	Method	Type of learning
[30,8]	ANFIS-SUB	Adaptive Neuro-Fuzzy Inference System using Subtractive Clustering
[39]	LINEAR-LMS	Least Mean Squares Linear Regression
[11]	TSK-IRL	Genetic learning of TSK Rules under Iterative Rule Learning
[1]	LEL-TSK	Local Evolutionary Learning of TSK Rules
[3]	FSMOGFS^e + TUN^e	*FMOGFS* for internal comparison. This learns: (Gr. and lateral partition params. and RB by WM) + (tuning of MF parameters and rule selection) by SPEA2$_{E/E}$ including error estimation
–	METSK-HD^e	Proposed here to learn: (Gr. and lateral partition params and zero-order TSK RB) + (tuning of MF parameters, rule selection and Kalman-based consequents) by SPEA2$_{E/E}$ including error estimation

Gr.: Granularities.

A brief description of the studied methods is presented in Table 2, which summarizes their main characteristics. The values of the parameters considered by LINEAR-LMS [39], TSK-IRL [11] and LEL-TSK [1] are those proposed by the authors of the methods. These methods are available at: http://www.keel.es/ and are accuracy oriented single-objective-based algorithms whose main objective is to obtain FRBSs as accurately as possible. In the case of the ANFIS-SUB method [8,30] the parameters considered are: Range of Influence, 0.5, Squash Factor, 1.25, Accept Ratio, 0.5 and Reject Ratio, 0.15 (the standard values commonly used in the literature). However, since the method crashed in a few datasets with these values, we changed them specifically in order to allow an appropriate application in these cases. In the case of the MOEA-based methods (FSMOGFS^e + TUN^e and METSK-HD^e) based on the well-known *SPEA2* [47], we have considered an external population size of 61 and a proportion of 1/3 rounded to 200 as the standard population size. The remaining parameters for them are: a maximum of 100,000 evaluations, 0.2 as mutation probability (crossover is always applied in SPEA2), 30 bits per gene for the Gray codification, $r^e = 0.2$ for the fast error computation technique (with a upper bound of 1000 instances), and the set $\{2, \ldots, 7\}$ as possible numbers of labels in all the system variables for the learning approaches. Table 3 resumes the parameters used by the proposed method. They are general fixed parameters for all the 28 datasets, so that it is not necessary to find particular parameter values for a given dataset.

In all the experiments, we adopted a *5-fold cross-validation model*, i.e., we randomly split the dataset into 5 folds, each containing 20% of the patterns of the dataset, and used four folds for training and one for testing.[1] For each of the five partitions, we executed six trials of the algorithms (6 different seeds). For each dataset, we therefore consider the average results of 30 runs. In the case of the MOEA-based algorithms (FSMOGFS^e + TUN^e and METSK-HD^e), the average values are calculated considering the most accurate solution from each obtained Pareto front. Our main aim following this approach is to have the possibility of statistically comparing the single objective approaches (only accuracy) with the most accurate solution found by the proposed MOEA.

In order to assess whether significant differences exist among the results, we adopt statistical analysis [13,23] and in particular non-parametric tests, according to the recommendations made in [13,23], where a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers has been analyzed. We will employ different approaches for multiple comparison, including Friedman's test [19], Iman and Davenport's test [29] and Holm's method [27]. For a detailed description of these tests and for detailed explanations of the use of non-parametric tests for data mining and Computational Intelligence see the Website at: http://sci2s.ugr.es/sicidm/. To perform the tests, we use a level of confidence $\alpha = 0.1$.

4.2. Internal comparison: linguistic vs. precise modeling

In this section, we present a brief comparison of the proposed method for precise modeling with respect to a recent evolutionary method for linguistic modeling called FSMOGFS^e + TUN^e [3], which was developed for application on high dimensional and large-scale datasets. Even though linguistic and precise models are not comparable (they are developed with different purposes), as some of the ideas and philosophy of the proposed method are based on this previous one, we want to show that it clearly outperforms the previous method in terms of accuracy, thus giving an interesting alternative when this is the principal requirement. This previous method is also based on a multi-objective embedded genetic DB learning, which allows a slight uniform displacement of linguistic fuzzy partitions and includes the fast error estimation.

The results obtained by the studied methods are shown in Table 4. This table is grouped in columns by algorithms and it shows the average of the results obtained by each algorithm in all the studied datasets. For each one, the first column shows the average number of rules and variables used (R/V). The second and third columns show the average *MSE* in training and test data (Tra./Tst.). Moreover, Table 4 also includes the results obtained in the first stage of the proposed method (intermediate results of the METSK-HD^e method).

In this case (with only two algorithms to compare), we adopt statistical analysis for pair-wise comparison, in particular we use Wilcoxon's Signed-Ranks test [40,46]. Wilcoxon's test is based on computing the differences between two sample means (typically, mean test errors obtained by a pair of different algorithms on different datasets). In the classification

[1] The corresponding data partitions (5-fold) for these datasets are available at the KEEL project webpage [5]: http://sci2s.ugr.es/keel/datasets.php.

Table 3
METSK-HD^e fixed parameters.

Population size	200
External population size	61
Evaluations	100,000
Mutation probability	0.2
Number of labels	$\{2,\ldots,7\}$
Bits per gene for the gray codification	30
Rate of examples used to estimate the error	0.2
Maximum of examples used to estimate the error	1000
Maximum number of rules for the rule cropping (first stage)	100

Table 4
Average results of the proposed accurate method (METSK-HD^e) vs. linguistic method (FS$_{MOGFS}^e$ + TUN^e). Results in this table (Tra. and Tst.) should be multiplied by $10^{+5}, 10^{-8}, 10^{-6}, 10^{+9}, 10^{+8}, 10^{-6}, 10^{-4}$ or 10^{-8} in the case of ELE1, DELAIL, DELELV, CAL, HOU, ELV, PUM or AIL respectively. The test values of the algorithm with the best result on each dataset are shown in boldface.

DATASET	FS$_{MOGFS}^e$ + TUN^e			METSK-HD^e					
				(FIRST STAGE)			(FINAL RESULT)		
	R/*V*	Tra.	Tst.	*R*/*V*	Tra.	Tst.	*R*/*V*	Tra.	Tst.
ELE1 (2/495)	8.1/2	1.516	**1.954**	15/2	1.691	1.925	11.4/2	1.350	2.022
PLA (2/1650)	18.6/2	1.106	1.194	23/2	1.192	1.218	19.2/2	1.057	**1.136**
QUA (3/2178)	3.2/1.3	0.0175	**0.0178**	35.9/2.9	0.0181	0.0185	18.3/2.9	0.0171	0.0181
ELE2 (4/1056)	8/2	9665	10,548	59/4	19,452	20,095	36.9/4	2270	**3192**
FRIE (5/1200)	22/3.1	2.71	3.138	95.1/4.1	2.868	3.084	66/4.1	1.075	**1.888**
MPG6 (5/398)	20/3	2.86	4.562	99.6/4.9	2.904	4.469	53.6/4.9	1.082	**4.478**
DELAIL (5/7129)	6.2/2.6	1.498	1.528	98.3/5	1.547	1.621	36.8/5	1.190	**1.402**
DEE (6/365)	18.3/3.8	0.059	**0.093**	96.4/4.2	0.064	0.095	50.6/4.2	0.030	0.103
DELELV (6/9517)	7.9/2.6	1.072	1.086	91/4.2	1.102	1.119	39.1/4.2	0.9725	**1.031**
ANA (7/4052)	10/3	0.003	**0.003**	48.9/4.3	0.005	0.006	33.3/4.3	0.002	0.004
MPG8 (7/398)	23/3	2.757	**4.747**	98.7/4.7	2.692	5.610	64.2/4.7	1.154	5.391
ABA (8/4177)	8/3	2.445	2.509	42.4/4.2	2.523	2.581	23.1/4.2	2.205	**2.392**
CAL (8/20,640)	8.4/2.9	2.94	2.95	99.8/4.9	2.617	2.638	55.8/4.9	1.64	**1.71**
CON (8/1030)	15.4/3.5	29.901	32.977	96.5/4.2	34.089	38.394	53.7/4.2	15.054	**23.885**
STP (9/950)	23/3	0.764	0.912	100/5.3	0.696	0.780	66.4/5.3	0.167	**0.387**
WAN (9/1609)	8/2	1.441	1.635	91.1/4.7	1.428	1.773	48/4.7	0.701	**1.189**
WIZ (9/1461)	10/2	0.929	1.011	55.4/4	1.185	1.296	29.1/4	0.729	**0.944**
FOR (12/517)	10/3	1418	**2628**	93.7/5.2	1643	4633	40.6/5.2	551	5587
MOR (15/1049)	7/2	0.016	0.019	40.9/4.3	0.026	0.028	27.2/4.3	0.005	**0.013**
TRE (15/1049)	9/3	0.034	0.044	42.8/4.6	0.045	0.052	28.1/4.6	0.017	**0.038**
BAS (16/337)	17/6	141,320	**261,322**	95.7/7	112,347	320,133	59.8/7	47,900	368,820
MV (10/40,768)	14/3	0.158	0.158	76.4/4	0.244	0.244	56.5/4	0.06	**0.061**
HOU (16/22,784)	11.7/4.4	9.35	9.4	68.9/5	10.224	10.368	30.5/5	8.29	**8.64**
ELV (18/16,559)	8/3	9	9	76.4/5.5	8.79	8.90	34.9/5.5	6.75	**7.02**
CA (21/8192)	14/5	5.021	5.216	71.3/6.1	5.760	5.880	32.9/6.1	4.376	**4.949**
POLE (26/14,998)	13.1/4.5	100.845	102.816	100/6.3.	149.641	150.673	46.3/6.3	57.964	**61.018**
PUM (32/8192)	17.6/2	0.29	0.292	87.5/4	0.587	0.594	63.3/4	0.2669	**0.2871**
AIL (40/13,750)	15/4	1.95	2	99.1/6	1.788	1.822	48.4/6	1.39	**1.51**

framework these differences are well defined since these errors are in the same domain. In our case, to have well defined differences in *MSE*, we propose the adoption of a normalized difference *DIFF*, defined as:

$$DIFF = \frac{Mean(Other) - Mean(ReferenceAlgorithm)}{Mean(Other)}, \tag{2}$$

where $Mean(x)$ represents the *MSE* means obtained by the x algorithm. This difference expresses the improvement percentage of the reference algorithm.

Table 5 shows the results of the Wilcoxon test on the test error for the proposed method and linguistic-based one. The results show that METSK-HD^e outperforms FS$_{MOGFS}^e$ + TUN^e on the test error. The null hypothesis associated with Wilcoxon's test is rejected ($p < \alpha$), in favor of METSK-HD^e due to the differences between R^+ and R^-. This is due to the use of TSK FRBS and the new ideas included in the algorithm such as the efficient Kalman filter and the use of a new objective which prevents overfitting.

4.3. Comparison to other related well-known methods for precise modeling

This section analyzes the results of the proposed method, METSK-HD^e, with respect to the previous accuracy oriented contributions as explained in Section 4.1. The results obtained by the studied methods are shown in Table 6. This table is

Table 5
Wilcoxon's test: METSK-HD^e (R^+) vs FS$_{MOGFS}^e$ + TUN^e (R^-) on *MSE* in tst.

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.1$)	p-Value
METSK-HD^e vs. FS$_{MOGFS}^e$ + TUN^e	321	85	Rejected	0.007

grouped in columns by algorithms and it shows the average of the results obtained by each algorithm in all the studied datasets. For each one, the first column shows the average number of rules (R), except in the case of the Linear-LMS method since this method is not a rule-based approach. The second and third columns show the average *MSE* in training and test data (Tra./Tst.). We also show the number of labels (NL) used for some algorithms. Further, since the proposed algorithm is able to reduce the number of variables used, we also show the average number of variables together with the rules (R/V).

No values are shown for TSK-IRL and LEL-TSK in several datasets since the large number of variables and cases provoked memory overflow errors after several hours running without finishing the evaluation of the initial population (some memory issues were improved in these methods to solve this problem, which helped to show results in at least some of the datasets with more than 8 variables, but it was impossible to run them with more complex problems). For this reason, as TSK-IRL is only applicable to a small number of datasets, we will compare it first by only considering the results obtained in the corresponding datasets. In this case, we again have two algorithms to compare, so that we adopt the same statistical analysis for pair-wise comparison as was presented in the previous subsection.

Additionally, since this time we will also compare more than two algorithms together, we also use non-parametric tests for multiple comparison. In order to perform a multiple comparison, it is necessary to check whether any of the results obtained by the algorithms present any inequality. In the case of finding some we can find out, by using a post hoc test, which algorithms' partners' average results are dissimilar. Of course, since accuracy is our main objective we will use the results obtained in Tst., defining the control algorithm as the best performing one (which obtains the lowest value of ranking, computed through a Friedman test [19]). In order to test whether significant differences exist among all the mean values we use Iman and Davenport's test [29]. Finally, we use Holm's [27] post hoc test to compare the control algorithm with the remainder.

In the following subsections, we present the corresponding statistical analysis depending on the group of datasets to which each method is applicable. As noted above, the first subsection compares METSK-HD^e to the method that can only

Table 6
Average results of the different algorithms. Results in this table (Tra. and Tst.) should be multiplied by 10^{+5}, 10^{-8}, 10^{-6}, 10^{+9}, 10^{+8}, 10^{-6}, 10^{-4} or 10^{-8} in the case of ELE1, DELAIL, DELELV, CAL, HOU, ELV, PUM or AIL respectively. The test values of the algorithm with the best result on each dataset are shown in boldface.

DATASET	ANFIS-SUB			TSK-IRL				LINEAR-LMS		LEL-TSK			METSK-HD^e		
	R	Tra.	Tst.	NL	R	Tra.	Tst.	Tra.	Tst.	R	Tra.	Tst.	R/V	Tra.	Tst.
ELE1 (2/495)	27.8	1.513	2.150	5	19.2	1.414	2.074	1.993	2.093	27	1.190	2.402	11.4/2	1.350	**2.022**
PLA (2/1650)	114	1.011	1.504	5	21	1.090	1.146	1.166	1.172	66	1.032	1.188	19.2/2	1.057	**1.136**
QUA (3/2178)	40.4	0.015	0.155	5	102	0.0164	0.0230	0.0178	**0.0179**	127	0.0151	0.0308	18.3/2.9	0.0171	0.0181
ELE2 (4/1056)	2	8208	8525	5	262	17,024	19,786	13,361	13,541	44.8	2928	3752	36.9/4	2270	**3192**
FRIE (5/1200)	53.8	0,085	3.158	3	3055	0.433	1.419	3.612	3.653	435	0.322	**1.070**	66/4.1	1.075	1.888
MPG6 (5/398)	299.6	0.002	8.079	3	785.8	1.338	5.029	5.780	6.084	79	1.473	6.357	53.6/4.9	1.082	**4.478**
DELAIL (5/7129)	57.2	0.973	1.484	3	233.2	1.321	1.419	1.478	1.480	105.2	1.193	1.760	36.8/5	1.190	**1.402**
DEE (6/365)	290.6	3087	2083	3	3054.2	0.545	882.016	0.081	**0.085**	57.8	0.662	0.682	50.6/4.2	0.030	0.103
DELELV (6/9517)	2	1.010	**1.020**	3	727.4	1.005	1.345	1.048	1.049	219.9	0.9642	2.788	39.1/4.2	0.972	1.031
ANA (7/4052)	10.4	0.027	0.029	3	111	0.028	460,488	0.085	0.085	301	0.009	0.014	33.3/4.3	0.002	**0.004**
MPG8 (7/398)	13.8	1.191	26.136	3	819	1.784	4.338	5.397	5.678	129	1.574	7.111	64.2/4.7	1.154	**5.391**
ABA (8/4177)	9	2.008	2.733	3	2077	2.581	2.642	2.413	2.472	107	2.040	2.412	23.1/4.2	2.205	**2.392**
CAL (8/20,640)	5.2	1.742	1.783					2.42	2.43	542	2.37	2.54	55.8/4.9	1.64	**1.71**
CON (8/1030)	20.6	12.286	188.290	3	2754	13.167	**19.151**	53.475	54.735	325	10.692	31.430	53.7/4.2	15.054	23.885
STP (9/950)	13.2	0.134	**0.307**					2.686	2.761	78.9	0.606	0.849	66.4/5.3	0.167	0.387
WAN (9/1609)	6	0.639	**0.845**					1.213	1.241	123	0.709	1.632	48/4.7	0.701	1.189
WIZ (9/1461)	6.2	0.544	**0.701**					0.782	0.800	116	0.699	2.227	29.1/4	0.729	0.944
FOR (12/517)	42.2	0.176	204,755					1968.57	**2013.89**	418	160.35	14074.50	40.6/5.2	551.38	5587.44
MOR (15/1049)	9.4	0.001	**0.003**					0.009	0.010	64.3	0.259	0.472	27.2/4.3	0.005	0.013
TRE (15/1049)	10	0.009	**0.019**					0.030	0.032	63.7	0.267	0.504	28.1/4.6	0.017	0.038
BAS (16/337)	6.4	119,561	1,089,824					224,684	**269,122**	374	9607	461,402	59.8/7	47,900	368,820
MV (10/40,768)	2	0.051	**0.052**					10.076	10.085				56.5/4	0.060	0.061
HOU (16/22,784)	3	7.254	**7.598**					10.34	10.40				30.5/5	8.29	8.64
ELV (18/16,559)	3	61.417	61.350					4.254	**4.288**				34.9/5.5	6.75	7.02
CA (21/8192)	3	7.14E+11	6.09E+11					45.809	46.820				32.9/6.1	4.376	**4.949**
POLE (26/14,998)	3	127.40	131.69					463.44	465.01				46.3/6.3	57.96	**61.02**
PUM (32/8192)	4	4.482	4.852					3.55	3.59				63.3/4	0.2669	**0.2871**
AIL (40/13,750)	2.2	1.33	**1.37**					1.55	1.62				48.4/6	1.39	1.51

be run on a small number of datasets (i.e., small datasets). The second subsection compares METSK-HD^e to those methods that are also able to be applied to medium and even highly complex datasets. The final subsection draws some conclusions regarding the results obtained.

4.3.1. Analysis of the method that can only be executed in small datasets

First, we compare the proposed algorithm with the method that can only be executed with the simplest datasets. The test error of the proposed method (METSK-HD^e) is compared to that of TSK-IRL [11] in the first 13 datasets.

We use Wilcoxon's Signed-Ranks test [40,46] to compare the proposed method and TSK-IRL. Table 7 shows the results of the Wilcoxon test on the test error. The results show that METSK-HD^e outperforms TSK-IRL on the test error for the datasets considered. The null hypothesis associated with Wilcoxon's test is rejected ($p < \alpha$), in favor of METSK-HD^e due to the differences between R^+ and R^-.

4.3.2. Analysis of methods that can be executed in all or almost all datasets

In this subsection, we compare the proposed algorithm with the methods that can be executed with almost every dataset. The test error of the proposed method (METSK-HD^e) is compared to that of LEL-TSK [1], ANFIS-SUB [8,30] and Linear-LMS [39] for the first 21 datasets.

Table 8 shows the rankings of the different methods considered in this study. Iman and Davenport's test [29] tells us that significant differences exist among the observed results in all datasets. The best rankings are obtained by the proposed method (METSK-HD^e). We now apply Holm's method [27] to compare the best ranking method with the remaining methods. Table 9 presents these results. Holm's test rejects the hypothesis of equality with the remaining methods in Tst ($p < \alpha/i$). From this analysis we can state that METSK-HD^e outperforms the other methods.

Furthermore, we compare the proposed algorithm with the methods that can be executed with all the datasets. The test error of the proposed method (METSK-HD^e) is compared to that of ANFIS-SUB [8,30] and Linear-LMS [39].

Table 10 shows the rankings of the different methods considered in this study. Iman and Davenport's test [29] tells us that significant differences exist among the observed results in all datasets. The best ranking is obtained by the proposed method (METSK-HD^e). We now apply Holm's method [27] to compare the best ranking method to the remaining methods. Table 11 presents these results. In this table, the algorithms are ordered with respect to the z-value obtained. Holm's test rejects the hypothesis of equality with the rest of the methods in Tst ($p < \alpha/i$). From this analysis we can state that METSK-HD^e outperforms the other two methods in accuracy.

4.3.3. Global analysis of the results

Analyzing the results shown in Table 6 and taking into account the results of the statistical tests, we can draw the following conclusions:

- The TSK-IRL [11] and LEL-TSK [1] methods obtain very accurate results on training, which usually causes them to present overfitting and very bad test errors.

Table 7
Wilcoxon's test: METSK-HD^e (R^+) vs TSK-IRL (R^-) on *MSE* in tst.

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.1$)	p-Value
METSK-HD^e vs. TSK-IRL	84	21	Rejected	0.048

Table 8
Rankings obtained through Friedman's test on *MSE* in tst (first 21 datasets).

Algorithm	Ranking on *MSE* in tst
METSK-HD^e	1.714
Linear-LMS	2.476
ANFIS-SUB	2.667
LEL-TSK	3.143

Table 9
Holm's post hoc test with $\alpha = 0.1$ on *MSE* in tst (first 21 datasets).

i	Algorithm	z	p	α/i	Hypothesis
3	LEL-TSK	3.586	3.362E−4	0.03	Rejected
2	ANFIS-SUB	2.390	0.017	0.05	Rejected
1	Linear-LMS	1.912	0.056	0.1	Rejected

Table 10
Rankings obtained through Friedman's test on *MSE* in tst.

Algorithm	Ranking on *MSE* in tst
METSK-HD^e	1.607
ANFIS-SUB	2.143
LINEAR-LMS	2.250

- The ANFIS-SUB [8,30] and Linear-LMS [39] methods present very competitive results in both training and test with respect to the previous approaches. Moreover, both methods (apart from the one proposed) can be applied to the most complex datasets.
- The proposed method presents simple solutions (a lower number of variables and rules) without significant overfitting. This method obtains the best results in test error in general as shown by the statistical tests in the previous subsections.

4.4. Computational times and scalability of the proposed algorithm

With respect to scalability it is very important to analyze the running times of the different methods (these times were obtained in an Intel Core 2 Quad Q9550 2.83 GHz, 8 GB RAM by using only one of the four cores). Table 12 shows the running times of the different algorithms. Moreover, Table 12 also includes the times obtained in the first stage of the proposed method.

In this case, except for the most complex datasets, the proposed method is able to obtain solutions taking only several minutes (less than one hour). Further, the times for the most complex ones are also very good, taking into account the kinds of problems they represent.

Table 11
Holm's post hoc test with $\alpha = 0.1$ on *MSE* in tst.

i	Algorithm	z	p	α/i	Hypothesis
2	Linear-LMS	2.405	0.016	0.05	Rejected
1	ANFIS-SUB	2.004	0.045	0.1	Rejected

Table 12
Average time of a run of the different methods – hours, minutes and seconds (h:m:s).

DATASET	ANFIS-SUB	TSK-IRL	LINEAR-LMS	LEL-TSK	METSK-HD^e	
					(FIRST STAGE)	(FINAL)
ELE1 (2/495)	0:00:30	0:02:16	0:00:02	0:02:01	0:00:02	0:01:01
PLA (2/1650)	0:17:53	0:07:18	0:00:02	0:18:12	0:00:05	0:03:09
QUA (3/2178)	0:05:55	0:32:35	0:00:02	0:50:02	0:00:17	0:08:38
ELE2 (4/1056)	0:00:49	0:27:25	0:00:02	0:09:39	0:00:11	0:12:58
FRIE (5/1200)	0:25:50	11:12:23	0:00:06	1:33:33	0:00:27	0:39:57
MPG6 (5/398)	4:09:47	0:55:30	0:00:00	0:12:47	0:00:12	0:27:42
DELAIL (5/7129)	1:45:13	4:05:31	0:00:02	2:48:41	0:02:41	2:30:39
DEE (6/365)	6:55:34	3:24:38	0:00:03	0:06:20	0:00:11	0:20:15
DELELV (6/9517)	0:00:52	23:01:40	0:06:36	10:33:35	0:02:41	1:29:30
ANA (7/4052)	0:03:29	1:20:18	0:00:01	5:22:26	0:00:48	0:40:23
MPG8 (7/398)	0:00:34	1:46:18	0:00:01	0:26:19	0:00:12	0:25:41
ABA (8/4177)	0:03:28	20:54:04	0:00:01	2:41:04	0:00:58	0:28:55
CAL (8/20,640)	0:08:56		0:00:06	63:17:03	0:05:07	5:13:28
CON (8/1030)	0:03:26	13:26:42	0:00:02	1:52:33	0:00:24	0:35:02
STP (9/950)	0:01:47		0:00:02	0:34:39	0:00:26	0:43:45
WAN (9/1609)	0:00:54		0:00:02	1:41:19	0:00:30	0:47:12
WIZ (9/1461)	0:00:51		0:00:02	1:29:14	0:00:37	0:19:33
FOR (12/517)	0:17:31		0:00:02	6:20:35	0:00:27	0:27:55
MOR (15/1049)	0:02:31		0:00:02	0:33:14	0:00:31	0:07:55
TRE (15/1049)	0:02:45		0:00:02	0:43:16	0:00:30	0:10:59
BAS (16/337)	0:00:28		0:00:02	2:02:47	0:00:26	0:51:58
MV (10/40,768)	0:08:57		0:00:01		0:07:51	3:17:54
HOU (16/22,784)	0:14:18		0:00:02		0:08:14	5:07:58
ELV (18/16,559)	0:11:25		0:00:02		0:04:59	3:06:58
CA (21/8192)	0:00:45		0:00:01		0:04:06	3:37:49
POLE (26/14,998)	0:15:04		0:00:01		0:07:56	4:40:22
PUM (32/8192)	0:16:28		0:00:02		0:07:36	2:22:25
AIL (40/13,750)	0:16:45		0:00:02		0:11:21	5:26:30

TSK-IRL [11] and LEL-TSK [1] can take a significant amount of time in problems when the number of variables and/or instances becomes high. These algorithms cannot run in high-dimensional datasets, because the large number of variables and cases provokes memory overflow errors after several hours running without finishing the evaluation of the initial population.

Even though ANFIS-SUB [8,30] and Linear-LMS [39] are very fast methods, the results in Table 6 and the statistical tests in Tables 9–11 show how the additional time used by the proposed algorithm enables it to obtain the best results within a reasonable time (which is not highly affected by complex problems).

5. Conclusions

This paper presents a scalable two-stage multi-objective genetic algorithm for precise fuzzy modeling of scatter-based TSK FRBSs in high-dimensional and large-scale regression problems. In the first stage, an evolutionary DB learning is performed (involving variables, granularities and slight fuzzy partition displacements). The RB is obtained within the same process using an efficient *ad hoc* algorithm to estimate the coefficients of the TSK consequents. The MOEA includes some specific mechanisms to ensure a fast learning of TSK FRBSs, allowing us to obtain the model structure and to prevent premature convergence in problems with a high number of variables and examples. The second post-processing stage performs a rule selection and a fine scatter-based tuning of the MFs. Moreover, it incorporates an efficient Kalman filter [33] to estimate the coefficients of the consequent polynomial functions of the TSK rules, which helps to significantly improve the performance of the model. We propose the use of MOEAs as a tool, mainly focusing on obtaining accurate models.

The results obtained in 28 datasets of different complexities confirm the effectiveness of the proposed method. METSK-HD^e has shown that it is able to obtain very accurate models avoiding overfitting in test error. Moreover, the proposed method has been compared to other well recognized methods, showing the best results on *MSE* in test. The scalability of the proposed method is also a key characteristic, which is able to solve problems with 40 variables or more than 40,000 cases in a fast way (which is reasonable for an evolutionary-based approach).

References

[1] R. Alcalá, J. Alcalá-Fdez, J. Casillas, O. Cordón, F. Herrera, Local identification of prototypes for genetic learning of accurate TSK fuzzy rule-based systems, Int. J. Intell. Syst. 22 (9) (2007) 909–941.
[2] R. Alcalá, J. Alcalá-Fdez, F. Herrera, A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection, IEEE Trans. Fuzzy Syst. 15 (4) (2007) 616–635.
[3] R. Alcalá, M.J. Gacto, F. Herrera, A fast and scalable multi-objective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems, IEEE Trans. Fuzzy Syst. 19 (4) (2011) 666–681.
[4] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multiple-Valued Logic Soft Comput. 17 (2-3) (2011) 255–287.
[5] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms to data mining problems, Soft Comput. 13 (3) (2009) 307–318.
[6] M. Antonelli, P. Ducange, F. Marcelloni, Genetic training instance selection in multiobjective evolutionary fuzzy systems: a coevolutionary approach, IEEE Trans. Fuzzy Syst. 20 (2) (2012) 276–290.
[7] Mohammad Biglarbegian, William Melek, Jerry Mendel, On the robustness of type-1 and interval type-2 fuzzy logic systems in modeling, Inform. Sci. 181 (7) (2011) 1325–1347, http://dx.doi.org/10.1016/j.ins.2010.11.003.
[8] S. Chiu, Fuzzy model identification based on cluster estimation, J. Intell. Fuzzy Syst. 2 (3) (1994) 267–278.
[9] M. Cococcioni, B. Lazzerini, F. Marcelloni, On reducing computational overhead in multi-objective genetic Takagi–Sugeno fuzzy systems, Appl. Soft Comput. 11 (1) (2011) 675–688.
[10] C.A. Coello, D.A. Van Veldhuizen, G.B. Lamont (Eds.), Evolutionary Algorithms for Solving Multi-objective Problems, Kluwer Academic Publishers, 2002.
[11] O. Cordón, F. Herrera, A two-stage evolutionary process for designing TSK fuzzy rule-based systems, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 29 (6) (1999) 703–715.
[12] K. Deb, Multi-objective Optimization using Evolutionary Algorithms, John Wiley & Sons, NY, USA, 2001.
[13] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[14] J. Derrac, C. Cornelis, S. Garcfa, F. Herrera, Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection, Inform. Sci. 186 (1) (2012) 73–92.
[15] L.J. Eshelman, The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination, in: G.J.E. Rawlin (Ed.), Foundations of Genetic Algorithms, vol. 1, Morgan Kaufmann, 1991, pp. 265–283.
[16] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, Found. Genet. Algorithms 2 (1993) 187–202.
[17] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, F. Herrera, A review of the application of multiobjective evolutionary fuzzy systems: current status and further directions, IEEE Trans. Fuzzy Syst. 21 (1) (2013) 45–65.
[18] Bruno B. Ferreira, Adriano J.O. Cruz, A parallel method for tuning Fuzzy TSK systems with CUDA, in: SBC – Proceedings of SBGames, Brazilian Computer Society (SBC), 2012, pp. 5–8.
[19] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (1937) 675–701.
[20] M.J. Gacto, R. Alcalá, F. Herrera, Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems, Soft Comput. 13 (5) (2009) 419–436.
[21] M.J. Gacto, R. Alcalá, F. Herrera, A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems, Appl. Intell. 36 (2) (2012) 330–347.
[22] M.J. Gacto, R. Alcalá, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures, Inform. Sci. 181 (20) (2011) 4340–4360.
[23] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[24] A.F. Gómez-Skarmeta, F. Jiménez, G. Sánchez, Improving interpretability in approximative fuzzy models via multiobjective evolutionary algorithms, Int. J. Intell. Syst. 22 (9) (2007) 943–969.
[25] O. Guenounou, A. Belmehdi, B. Dahhou, Multi-objective optimization of TSK fuzzy models, Expert Syst. Appl. 36 (4) (2009) 7416–7423.
[26] F. Herrera, Genetic fuzzy systems: taxonomy, current research trends and prospects, Evol. Intell. 1 (2008) 27–46.

[27] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[28] Tzung-Pei Hong, Wei-Tee Lin, Chih-Ping Chu, Chen-Sen Ouyang, An iterative genetic learning approach for Takagi–Sugeno fuzzy systems, in: 2009 International Conference on Machine Learning and Cybernetics, vol. 6, IEEE, 2009, pp. 3246–3251.
[29] R.L. Iman, J.H. Davenport, Approximations of the critical region of the fbietkan statistic, Commun. Stat. Part A Theory Methods 9 (1980) 571–595.
[30] J.S.R. Jang, ANFIS: adaptive network based fuzzy inference system, IEEE Trans. Syst. Man Cybernet. 23 (3) (1993) 665–684.
[31] Yaochu Jin, Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement, IEEE Trans. Fuzzy Syst. 8 (2) (2000) 212–221.
[32] C.F. Juang, T.C. Chen, W.Y. Cheng, Speedup of implementing fuzzy neural networks with high-dimensional inputs through parallel processing on graphic processing units, IEEE Trans. Fuzzy Syst. 19 (4) (2011) 717–728.
[33] Rudolph Emil Kalman, A new approach to linear filtering and prediction problems, Trans. ASME – J. Basic Eng. 82 (Series D) (1960) 35–45.
[34] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, Real-coded memetic algorithms with crossover hill-climbing, Evol. Comput. 12 (2004) 273–302.
[35] E.H. Mamdani, Application of fuzzy logic to approximate reasoning using linguistic systems, IEEE Trans. Comput. 26 (12) (1977) 1182–1191.
[36] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. Man–Mach. Stud. 7 (1975) 1–13.
[37] L. Martínez, F. Herrera, An overview on the 2-tuple linguistic model for computing with words in decision making: extensions, applications and challenges, Inform. Sci. 207 (2012) 1–18.
[38] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems. Industrial Electronics, CRC Press/Francis Taylor, 2013.
[39] J.S. Rustagi, Optimization Techniques in Statistics, Academic Press, 1994.
[40] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC, 2003.
[41] A.H. Sonbol, M.S. Fadali, S. Jafarzadeh, TSK fuzzy function approximators: design and accuracy analysis, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 42 (3) (2012) 702–712.
[42] M. Sugeno, G.T. Kang, Structure identification of fuzzy model, Fuzzy Sets Syst. 28 (1988) 15–33.
[43] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Trans. Syst. Man Cybernet. 15 (1) (1985) 116–132.
[44] Di Wang, Xiao-Jun Zeng, John A. Keane, A simplified structure evolving method for Mamdani fuzzy system identification and its application to high-dimensional problems, Inform. Sci. 220 (2013) 110–123, http://dx.doi.org/10.1016/j.ins.2011.12.033.
[45] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybernet. 22 (6) (1992) 1414–1427.
[46] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.
[47] E. Zitzler, M. Laumanns, L. Thiele, Spea2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization, in: Proc. Evolutionary Methods for Design, Optimization and Control with App. to Industrial Problems, International Center for Numerical Methos in Engineering (CIMNE), Barcelona, Spain, 2001, pp. 95–100.