Local Identification of Prototypes for Genetic Learning of Accurate TSK Fuzzy Rule-Based Systems

R. Alcalá,[†] J. Alcalá-Fdez,[‡] J. Casillas,[§] O. Cordón,[¶] F. Herrera[*]
University of Granada, Department of Computer Science and Artificial Intelligence, 18071 Granada, Spain

This work presents the use of local fuzzy prototypes as a new idea to obtain accurate local semantics-based Takagi–Sugeno–Kang (TSK) rules. This allow us to start from prototypes considering the interaction between input and output variables and taking into account the fuzzy nature of the TSK rules. To do so, a two-stage evolutionary algorithm based on MOGUL (a methodology to obtain Genetic Fuzzy Rule-Based Systems under the Iterative Rule Learning approach) has been developed to consider the interaction between input and output variables. The first stage performs a local identification of prototypes to obtain a set of initial local semantics-based TSK rules, following the Iterative Rule Learning approach and based on an evolutionary generation process within MOGUL (taking as a base some initial linguistic fuzzy partitions). Because this generation method induces competition among the fuzzy rules, a post-processing stage to improve the global system performance is needed. Two different processes are considered at this stage, a *genetic niching-based selection process* to remove redundant rules and a *genetic tuning process* to refine the fuzzy model parameters. The proposal has been tested with two real-world problems, achieving good results.

1. INTRODUCTION

One of the most important areas for the application of Fuzzy Set Theory is Fuzzy Rule-Based Systems (FRBSs). There are at least two different kinds of FRBSs in the literature, the Mamdani[1] and Takagi-Sugeno-Kang (TSK)[2,3] ones, which differ on the composition of the rule consequent. The use of one or the other depends on the fact that the main requirement is the interpretability or the accuracy of the model, respectively.

*Author to whom all correspondence should be addressed: e-mail: herrera@decsai.ugr.es.
[†]e-mail: alcala@decsai.ugr.es.
[‡]e-mail: jalcala@decsai.ugr.es.
[§]e-mail: casillas@decsai.ugr.es.
[¶]e-mail: ocordon@decsai.ugr.es.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 22, 909–941 (2007)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20232

Some automatic techniques have been proposed to learn a proper set of TSK rules from numerical data. However, global learning of the fuzzy subspaces and the linear relation that must be established in each of them is a hard task.[4] As said in Refs. 5–7, the learning of the premises and consequents is usually performed separately, obtaining the optimum consequents for a previously learned premise set without considering the interaction between input and output variables.[8,9]

In this context, fuzzy clustering is one of the most useful techniques, detecting the possible groupings that exist and establishing some hypothesis about the structure present in the data.[8,10–12] To include output behaviors in the fuzzy partition (i.e., to consider the output data to obtain the premises), different works have considered the product space of input and output variables instead of only the input space.[2,5,10] However, some authors state that fuzzy clustering and fuzzy c-regression algorithms suffer from several drawbacks. On the one hand, they are very sensitive to the presence of outliers.[6,7,13] Furthermore, the membership of a point to a cluster depends on the membership of such a point to all the other clusters. So, the cluster centers or estimates for the parameters are poor.[13] This behavior is far from the TSK fuzzy nature.

Recently, these kinds of learning techniques have been taken into account as prototype-identification algorithms, summarizing a data set by a number of representative prototypes (objects lying in the same space as the sample points). This point of view follows the original ideas of Ruspini,[14] later expanded in many significant directions by relaxing the concept of prototype in a variety of ways, for example, line segments, ellipsoids, and so forth.[15] Some works particularize this concept by considering these prototypes as being fuzzy rules.[5,8,10–12] However, the prototype identification process still comes with the same drawbacks.

Having these concepts in mind, fuzzy rule generation methods also can be seen as identification algorithms with fuzzy rule prototypes, that is, fuzzy model builders whose main purpose is to extract the most suitable set of fuzzy rules from an object (input–output data) according to an optimization measure, which evaluates the quality of the approximation. Additionally, they organize results and summarize them by interest criteria to provide a more compact and useful representation of the resulting structures.

Two main approaches can be considered to obtain FRBSs:

- *Global semantics-based approach:* A global collection of fuzzy sets is considered by all the fuzzy rules. In global semantics-based FRBSs, each fuzzy set can have a real-world meaning associated. Conversely, this approach is less flexible than the local one, presenting some limitations in terms of accuracy.
- *Local semantics-based approach:* Each fuzzy rule has associated its own local fuzzy sets. In local semantics-based FRBSs, the rules have more freedom to improve their performance. However, these kinds of systems are less interpretable than the global ones.

In this work, we propose the use of local semantics-based Mamdani fuzzy rules as local fuzzy prototypes from which to obtain accurate local semantics-based TSK rules. This new idea allows us to start from prototypes considering the interaction between input and output variables and taking into account the fuzzy

nature of the TSK rules. To do so, we present a two-stage Genetic FRBS (GFRBS)[16] following the MOGUL paradigm,[17] a methodology to obtain GFRBSs under the Iterative Rule Learning (IRL) approach. First, we obtain an acceptable model as a first approximation by means of a local evolutionary learning performing the identification of local prototypes. Then, a postprocessing stage considers the global cooperation of such promising rules. Therefore, our main objective in this article is to obtain highly accurate fuzzy models, although it involves the loss of interpretability to some degree. Anyway, we must point out that there exist other research line to improve the interpretability in TSK fuzzy systems. These improvements are usually developed by reducing the fuzzy rule set (usually with orthogonal transformations or feature selection),[18,19] reducing the number of fuzzy sets (usually with similarity measures) with the subsequent merging of rules,[7,20] or exploiting the local description of the rules (basically smoothing the consequent polynomial function of the Takagi–Sugeno rule or isolating the fuzzy rule actions).[21,22] See Refs. 23 and 24 for detailed coverage of the state of the art in the trade-off between interpretability and precision.

The article is organized as follows. The next section describes the general TSK fuzzy model structure considered in this work. Section 3 discusses the main differences between the global and the local fuzzy prototype identification. Section 4 presents the structure of the proposed GFRBS. Sections 5 and 6 explain in detail each stage of the proposed GFRBS. Experimental results are shown in Section 7. Finally, some concluding remarks are made in Section 8.

2. TSK FUZZY RULE-BASED SYSTEMS

In Refs. 2 and 3, Takagi, Sugeno, and Kang presented a mathematical tool to procure a fuzzy model of a system. They suggested a multidimensional fuzzy reasoning in which the number of implications can be surprisingly decreased; that is, we would need fewer rules in the knowledge base. This structure will be introduced in the following.

2.1. Architecture

The fuzzy model is based on rules in which the consequent is not a linguistic variable (as in Mamdani fuzzy systems) but a function of the input variables. These kinds of rules usually present the following structure:

$$R_i\text{: If } X_1 \text{ is } A_{i1} \text{ and } \ldots \text{ and } X_n \text{ is } A_{in}\text{, then } Y = p_{i0} + p_{i1}\cdot X_1 + \cdots + p_{in}\cdot X_n$$

where X_i are the system input variables and Y is the system output variable that determines a local linear input–output relation by means of the real-valued coefficients p_{ij}.

The output of a FRBS considering a knowledge base composed of m TSK rules is computed as the weighted average of the individual rule outputs 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 $x_1, \ldots, x_n$, and with T being a t-norm.

The presented fuzzy reasoning is based on first dividing the (uni- or multidimensional) input space into some (uni- or multidimensional) fuzzy subspaces and then building a linear input–output relation in each individual subspace. As seen, these partial relations are combined by aggregation, taking into account their dominance in their respective area of application and the conflict existing in the overlapped areas.[3] In this way, TSK fuzzy systems present the following interesting features:

- *Locality:* This feature allows us to represent even quite complicated nonlinear static relationships through a collection of relatively simple local linear or nonlinear models.[25]
- *Smooth switching:* Because the outcomes are calculated as a weighted average of the individual rule outputs, a sort of gradual activation of the individual models can be guaranteed, procuring a smooth switching between them.[25]
- *Existence of mathematical tools for system design:* Due to the linear nature of TSK consequents, they can be obtained by means of specific mathematical techniques, which get the least system error for predetermined antecedents (e.g., by least squares estimation[3]).

The learning of TSK FRBSs is devoted to obtaining the premise and consequent structures. In the Introduction we have seen some approaches to obtaining the premises for these kinds of systems. We will be back to this later. In the following, we introduce some consequent structure generation approaches.

2.2. Consequent Structure Generation

The most common approaches for the generation of the consequent structure in TSK FRBSs are the following[26]:

- *Global least squares*: This gives the best solution in terms of the squared model output.[18,27,28] It is computationally expensive but the most used. This technique is very sensitive to the presence of outliers and to overfitting when training data is not quite representative.
- *Local least squares*: This neglects the overlap between the validity functions and estimates the parameters for each local linear model separately by a least squares technique with the data weighted according to their validity.[5,12] It is much more efficient than global estimation and reduces overfitting.
- *Product space clustering*: Under this approach for product space partitioning, the Gustafson–Kessel algorithm[29] automatically obtains the rule consequent parameters as a by-product.[9,10] It is similar to a global least squares if it converges (presenting the same drawbacks).

- *Evolutionary algorithms*: In the last few years, many different approaches have been presented taking evolutionary algorithms[30] as their base. Examples of these kinds of methods are to be found in Refs. 31 and 32. In this article, we will choose this approach, but any of the others could be used.

Other approaches for the estimation of the consequent model parameters in TSK FRBSs are the orthogonal transformation-based methods,[8,22] the use of neural networks,[27,33] and so on.

3. LOCAL VERSUS GLOBAL FUZZY RULE PROTOTYPE IDENTIFICATION

In this work, we propose the use of Mamdani fuzzy rules as fuzzy prototypes to identify a set of fuzzy subspaces grouping data with similar behavior. As we have seen, two different approaches can be considered to obtain these kinds of rules, those based on global and local semantics. From this point of view, we could obtain two different kinds of fuzzy prototypes, the global and the local ones. Each of them presents different advantages and drawbacks in terms of accuracy and interpretability. Figure 1 shows a graphical comparison of both kinds of fuzzy prototypes.

As a consequence of the inflexibility of the concept of linguistic variable, those approaches based on global semantics present the following drawbacks[34]:

- There is a lack of flexibility in the FRBS because of the rigid partitioning of the input and output spaces.
- When the system input variables are dependent among them, it is very hard to fuzzy partition the input spaces.
- The homogeneous partitioning of the input and output spaces when the input–output mapping varies in complexity within the space is inefficient and does not scale to high-dimensional spaces.
- The size of the fuzzy rule base directly depends on the number of variables and linguistic terms in the system. Obtaining an accurate FRBS requires a significant granularity amount; that is, it needs the creation of new linguistic terms. This granularity increase causes the number of rules to rise significantly, which may cause the system to lose the capability of being interpretable for human beings.

On the other hand, instead of considering a global semantics-based approach, we could consider a local approach. In this case, the local fuzzy prototypes are based on rules presenting the following structure:

$$R_i\text{: If } X_1 \text{ is } A_{i1} \text{ and } \ldots \text{ and } X_n \text{ is } A_{in}\text{, then } Y \text{ is } B$$

where A_i and B are fuzzy sets specific to each fuzzy rule.

Approaches based on local semantics present interesting advantages that make them very suitable for fuzzy 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, needing fewer rules in simple problems, and being able to use more rules if it is necessary. This is likely to be of benefit in tackling the curse of dimensionality when scaling to multidimensional systems.

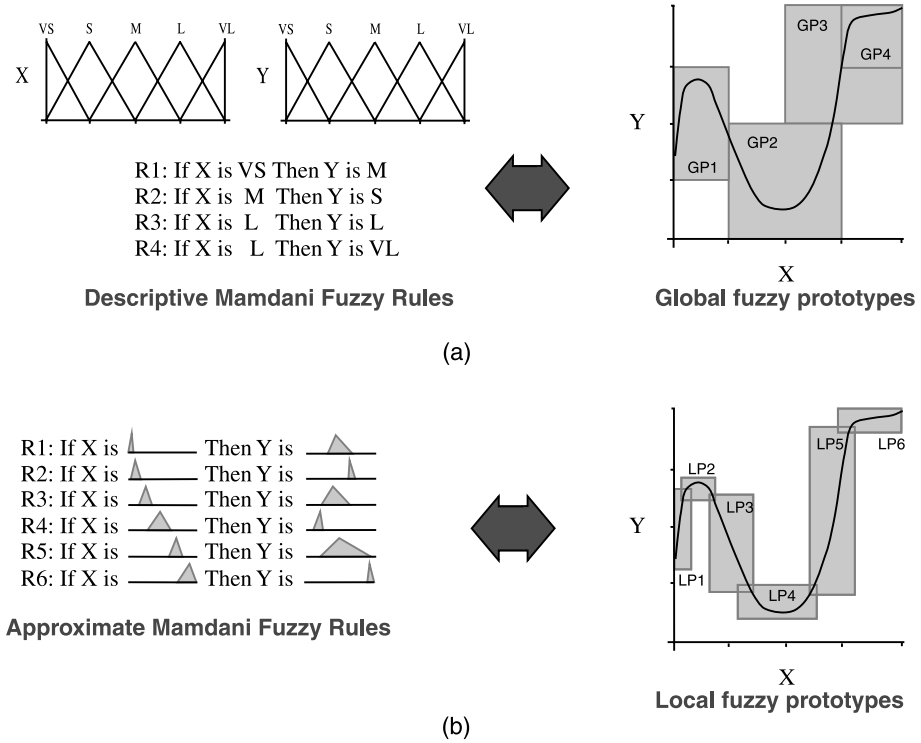


Figure 1. Graphical comparison between global and local fuzzy prototypes. (a) Global semantic-based Mamdani rules as global fuzzy prototypes. (b) Local semantic-based Mamdani rules as local fuzzy prototypes.

However, an important drawback of this local approach is that these kinds of systems are less interpretable than the global ones. Nevertheless, these kinds of FRBSs can be interpreted from a local point of view, and make use of expert knowledge and deductive processes.

The choice between how interpretable and how accurate the model must be usually depends on the user's needs for a specific problem, and it will condition the kind of FRBS selected to model it. As well as that, in this article we focus on developing more accurate fuzzy models, which can provide approximate solutions to different problems, especially real-world problems with accuracy requirements. Therefore, we propose the use of local semantics-based Mamdani rules as local fuzzy prototypes for local identification of TSK fuzzy rules. Figure 2 represents this concept.

In the literature, many works have considered both approaches to obtain TSK FRBSs based on global[2,3,5,9,35] and local[7,8,11,25,32] semantics. Figure 3 depicts both kinds of systems. In this work, we will focus on local semantics-based TSK FRBSs, those in which A_{ij} represent fuzzy sets specific to each fuzzy rule.

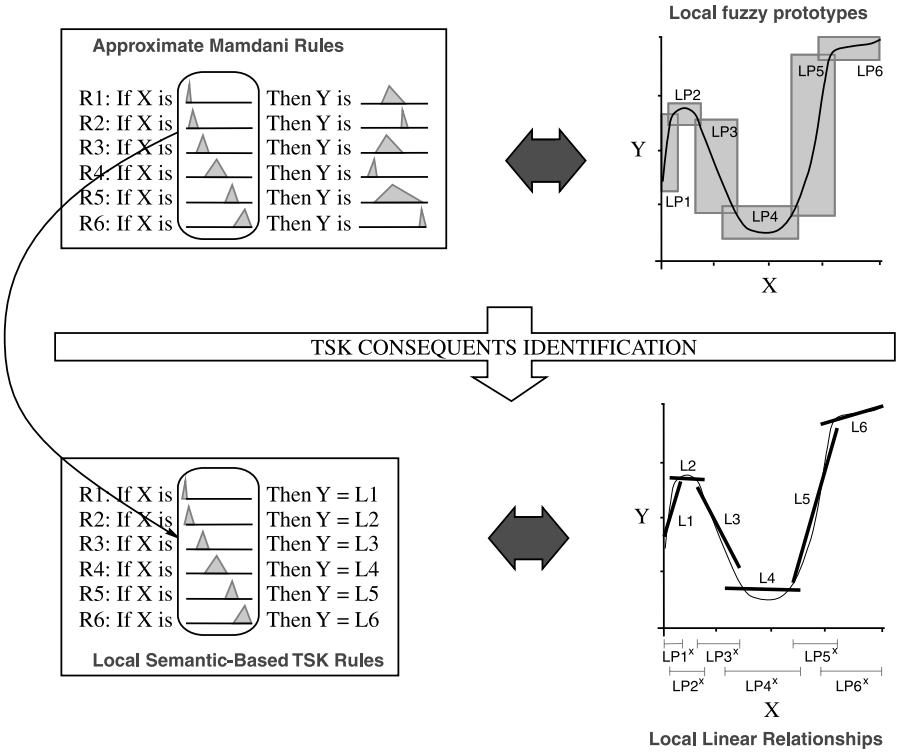
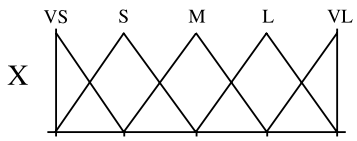


Figure 2. TSK consequents identification from local fuzzy prototypes.



R1: If X is VS Then Y = 0.4 X + 0.2
R2: If X is S Then Y = -0.9 X + 1.3
R3: If X is M Then Y = 1.0 X - 1.1
R4: If X is L Then Y = 0.3 X + 0.7
R5: If X is VL Then Y = 0.8 X + 0.1

(a)

R1: If X is Then Y = 0.2 X + 0.1
R2: If X is Then Y = -1.8 X + 0.3
R3: If X is Then Y = 0.6 X + 0.5
R4: If X is Then Y = 1.0 X + 0.7

(b)

Figure 3. Graphical comparison between (a) a global and (b) a local semantics-based TSK FRBS.

4. MULTISTAGE STRUCTURE OF THE PROPOSED GFRBS

In this section, we present a two-stage GFRBS to generate local semantics-based TSK FRBSs. It is based on the existence of a set of input–output training data $E_N = \{e_1, \ldots, e_l, \ldots, e_N\}$ with $e_l = (ex_1^l, \ldots, ex_n^l, ey^l)$, N being the data set size and n being the number of input variables.

The local identification of prototypes induces competition among rules by only considering the quality of the approximation performed by each rule. To do so, the proposed method has been integrated in MOGUL[17] by using an IRL-based approach. However, the global cooperation among rules should be considered in order to increase the generalization power of the system modeled. Following the MOGUL approach, a postprocessing stage is considered for this purpose. This way, the learning method substantially reduces the search space size by dividing the genetic learning process into two stages.

In this way, our method follows the ideas of the method presented by Yen et al.[22] in which global learning and local learning are combined in a single process. In our case, this process is divided in several stages; that is, first local learning is performed and then global learning is performed in a postprocessing phase.

Table I presents a summary of the main algorithm steps. The algorithm basically consists of the following stages:

1. *Local Process for Identifying Prototypes*. This stage performs a local identification of local semantics-based TSK rules following the IRL approach and based on an evolutionary generation process within MOGUL. The method takes as a base some initial linguistic fuzzy partitions. Then, for each iteration of the method:
   - A set of candidate global fuzzy prototypes is obtained by generating the global semantics-based Mamdani-type fuzzy rule best covering every example.
   - Then, the most promising Mamdani-type rule is selected and locally tuned to identify the local fuzzy prototype best grouping the data located in the corresponding subspace (the best local semantics-based Mamdani-type fuzzy rule). This process for local refinement of global prototypes is based on a (1 + 1)-Evolution Strategy ((1 + 1)-ES).[30]
   - Finally, the obtained prototype is added to the final set of fuzzy prototypes. The data covered by this set to a certain degree are removed and not considered for future iterations. The iterative process ends when no more uncovered training data remain.

   Once the set of local fuzzy prototypes (grouping similar data) is obtained and considering the same antecedents, the existing partial linear input–output relation is computed using the data located in each input subspace by means of a (μ, λ)-ES.[30]

Table I. Structure of the proposed GFRBS.

Local Process for Identifying Prototypes
Step 0. Initial linguistic partitioning
Step 1. Obtain the best global prototype, GP_r
Step 2. Identify the best local prototype, $LP_r \leftarrow ES_{(1+1)}(GP_r)$
Step 3. Introduce LP_r in the final set of fuzzy prototypes
Step 4. If uncovered data remain, go to Step 1
Step 5. Linear consequents derivation, $L_r \leftarrow ES_{(\mu,\lambda)}(LP_r^x)$
Postprocessing Stage
(a) Genetic niching-based selection process
(b) Genetic tuning process

2. *Postprocessing Stage*. Because the genetic generation method proposed in the previous stage induces competition among the fuzzy rules and the cooperation is slightly considered by means of the rule penalization criterion, a postprocessing stage to improve system performance is needed. Two different processes are considered at this stage, the *genetic niching-based selection process* and the *genetic tuning process*:
   (a) *Genetic Niching-Based Selection Process*. This summarization component has the aim of selecting the subset of rules best cooperating among the rules generated in the previous stage. This is capable of generating not only a single simplified TSK FRBS as output from the process but different ones presenting the best possible cooperation among the fuzzy rules composing them, and thereby the best possible behavior.
   (b) *Genetic Tuning Process.* Then, the evolutionary tuning process will be applied over these definitions and the most accurate will be the one given as the output of the two-stage GFRBS. Therefore, a model not presenting the best behavior after the second stage may be the best one after the third stage due to the fact that the new membership function shapes and consequent parameters make its rules cooperate in a better way.

In the following sections both stages are presented in depth. After that, some experiments are shown.

5. GFRBS I: LOCAL PROCESS FOR IDENTIFYING PROTOTYPES

This section presents the iterative process proposed for the local identification of fuzzy prototypes and the derivation of the corresponding TSK linear consequents. It is based on the *Soft Constrained Learning*-based evolutionary generation process presented in Ref. 36 to generate the set of local semantics-based Mamdani rules best covering the input–output data (local fuzzy prototypes). Then, the corresponding TSK consequents are obtained by means of the (μ, λ)-ES presented in Ref. 31.

The discussion starts introducing some criteria to be considered in this method. Taking these criteria into account, the algorithm to obtain local semantics-based TSK FRBSs is proposed. Finally, the two ESs considered in the algorithm are explained.

5.1. Some Considerations on the Stopping Criterion and the Fitness Function

Because this method is based on local covering measures to induce competition among rules, considering the *completeness* and *consistency* properties[37] is recommended to improve the behavior of the generated fuzzy rule bases. A fuzzy rule base is complete if it is possible to infer a proper output for every input and consistent if it does not contain contradictions.

In our case, completeness is verified by demanding that each example is covered to a degree $\epsilon \in \Re$ (determined by the system expert). It is achieved through an iterative process that will finish when each example is covered with a covering degree greater than ϵ.

On the other hand, to verify the consistency, the *positive* and *negative example* concepts[38] are considered in the fitness function by means of several criteria. An example is positive for a fuzzy rule when it matches with its antecedent and consequent, and it will be considered a negative example when it matches with its antecedent and not with its consequent. Hence, the fewer negative examples each rule has, the more consistent the fuzzy rule base is. Anyway, some negative examples are accepted when it presents a large number of positive examples.

Thereby, the accuracy of a simple fuzzy rule,[a] R_i, on the set of examples, E_N, is measured by using a multicriteria fitness function, designed to take into account three different criteria. This allows us to procure the consistency of the final set of generated rules. The criteria considered by the fitness function are:

- **High frequency value**[38]: The frequency of a fuzzy rule, R_i, through the set of examples, E_N, is defined as

$$\Psi_{E_N}(R_i) = \frac{\sum_{l=1}^{N} R_i(e_l)}{N}$$

  with $R_i(e_l)$ being the covering degree of the fuzzy rule R_i over the example e_l (antecedent and consequent part).
- **High average covering degree over positive examples**[38]: With $E_\omega^+(R_i) = \{e_l \in E_N \text{ s.t. } R_i(e_l) \geq \omega\}$ being the set of positive examples with a covering degree greater than or equal to $\omega \in [0,1]$, and $n_\omega^+(R_i) = |E_\omega^+(R_i)|$, the *average covering degree* on $E_\omega^+(R_i)$ can be defined as

$$G_\omega(R_i) = \sum_{e_l \in E_\omega^+(R_i)} \frac{R_i(e_l)}{n_\omega^+(R_i)}$$

- **Small negative example set**[39]: With $E^-(R_i) = \{e_l \in E_N \text{ s.t. } R_i(e_l) = 0 \text{ and } A_i(x^l) > 0\}$ being the set of negative examples, $n_{R_i}^- = |E^-(R_i)|$ and $A_i(\cdot)$ being the antecedent covering degree of the rule R_i over an example, the *penalty function on the set of negative examples* is

$$g_n(R_i^-) = \begin{cases} 1, & \text{if } n_{R_i}^- \leq k \cdot n_\omega^+(R_i) \\ \dfrac{1}{n_{R_i}^- - k \cdot n_\omega^+(R_i) + exp(1)}, & \text{otherwise} \end{cases}$$

  where we allow up to $k \cdot n_\omega^+(R_i)$ negative examples per rule without any penalty, with $k \in [0,1]$, and $exp(1)$ is the Napierian number.

These criteria are combined into a fitness in the following way (rules obtaining higher values in this function will be more accurate):

$$F(R_i) = \Psi_{E_N}(R_i) \cdot G_\omega(R_i) \cdot g_n(R_i^-)$$

[a]With global or local nature.

5.2. Identification Algorithm

Once the values of ϵ, ω, and k (defined in the previous subsection) are given by the GFRBS designer, the method for identification of local semantics-based TSK FRBSs may be summarized in the following algorithm:

Initializations:
(a) Initialize the set of examples E_p to E_N.
(b) Set the example covering degree $CV[l] \leftarrow 0$, $l = 1, \ldots, N$.
(c) Initialize the final set of prototypes B^i to empty.

Step 0 Initial linguistic partitioning: A strong fuzzy partition (those in which the sum of membership degrees within the variable domain is kept to 1) is considered for each variable. At this point, we should note that we are using uniform triangular-shaped membership functions (see Figure 1a).

Step 1 Obtain the best global prototype, GP_r:
- Initialize the candidate fuzzy rule set B^c to empty.
- For every $e_l \in E_p$, generate the fuzzy rule R_c best covering it by taking the linguistic label of the fuzzy partition best matching with the e_l component value for each variable. If $R_c \notin B^c$, add it to B^c.
- Evaluate all the fuzzy rules contained in B^c and select the one obtaining the highest value in the fitness function: R_r s.t. $F(R_r) = \max_{R_i \in B^c}(F(R_i))$
- Let GP_r be R_r.

Step 2 Identify the best local prototype, $LP_r \leftarrow ES_{(1+1)}(GP_r)$.

Step 3 Introduce LP_r in the final set of fuzzy prototypes: $B^i \leftarrow LP_r$.

Step 4 If uncovered data remain go to Step 1:
(a) For every $e_l \in E_p$ do
(i) $CV[l] \leftarrow CV[l] + LP_r(e_l)$, with $LP_r(e_l)$ being the covering degree of the prototype LP_r over the example e_l.
(ii) If $CV[l] \geq \epsilon$ then remove it from E_p.
(b) If $E_p \neq \emptyset$ return to Step 1.

Step 5 Linear consequents derivation, $L_r \leftarrow ES_{(\mu,\lambda)}(LP_r^x)$: For every $LP_r \in B^i$, take the same antecedents, LP_r^x, and obtain the corresponding linear input–output relation, L_r, considering the data located in the corresponding input subspace. Replace the fuzzy consequent of $LP_r \in B^i$ by L_r.

In the following subsections, the (1 + 1)-ES and the (μ, λ)-ES, considered in the Steps 2 and 5 of the proposed algorithm, will be explained.

5.3. (1 + 1)-ES for Local Refinement of Global Prototypes

Let GP_r be the global prototype to be adapted. This process, based on a (1 + 1)-ES,[30] performs a local tuning on GP_r (Step 2) to identify the local prototype best grouping the sample data located in its covered region. In the following, its main components are pointed out.

Coding scheme. A solution is directly encoded in a chromosome C, by joining the definition points of each one of the $n + 1$ triangular-shaped membership functions composing the corresponding fuzzy prototype: $C = (a_1, b_1, c_1) \ldots (a_n, b_n, c_n)(a_{n+1}, b_{n+1}, c_{n+1})$, with n being the number of input variables.

Mutation scheme. Two changes have to be performed in the usual ES mutation scheme when this technique is applied to this problem:

- On the one hand, two restrictions imposed in the membership function locations and shapes must be considered in the ES: it must lie in a local interval and it must be meaningful. With $C_i = (x_0, x_1, x_2)$ being a membership function to be adapted, the associated interval of performance is $[C_i^l, C_i^r] = [x_0 - (x_1 - x_0)/2, x_2 + (x_2 - x_1)/2]$. Moreover, $x_0 \leq x_1 \leq x_2$ must be verified. Hence, an incremental optimization of the individual parameters is required because the intervals of performance for each one of them depend on any of the others. Thus, the mutated fuzzy set $C_i' = (x_0', x_1', x_2')$ is obtained by first defining the mutated value x_1' in the interval $[x_0, x_2]$, and then defining the values x_0' and x_2' in the intervals $[C_i^l, x_1']$ and $[x_1', C_i^r]$, respectively.
- On the other hand, the parameter σ usually determines the mutation strength for all the individual parameters in ESs. However, in this case the membership functions encoded are defined over different universes. For this reason, the *definition of multiple step sizes* for each component, $\sigma_i = \sigma \cdot s_i = \sigma \cdot s(x_i)$, has to be performed. For a concrete membership function being mutated, $s(x_i)$ is computed before each x_i is adapted. These values are respectively calculated as

$$s(x_1) = \frac{Min(x_1 - x_0, x_2 - x_1)}{2}$$

$$s(x_0) = \frac{Min(x_0 - C_i^l, x_1' - x_0)}{2}$$

and

$$s(x_2) = \frac{Min(x_2 - x_1', C_i^r - x_2)}{2}$$

If the mutated value x_i' lies outside of its expected interval, the value of the interval extreme closer to it is taken.

Fitness function. It is based on the function, $F(\cdot)$, presented in Section 5.1. This function looks for the most general rule, widening its applicability and covering more examples with the least possible number of inconsistencies. However, to get a better interaction rate among rules, a penalty function based on a *low niche interaction rate* criterion[39] is used to avoid excessive proximity among them. With $N_i = (N_i x, N_i y)$, $i = 1, \ldots, |B^i|$, the centers of the local fuzzy prototypes already identified (those in B^i), and T being a t-norm (the minimum in this article), the following penalty function will be considered:

$$LNIR(C) = 1 - NIR(C)$$

$$NIR(C) = \max_i \{T(C_1(N_i x_1), \ldots, C_n(N_i x_n), C_{n+1}(N_i y))\}, \quad i = 1, \ldots, |B^i|$$

$$C \sim \text{IF } X_1 \text{ is } C_1 \text{ and } \ldots \text{ and } X_n \text{ is } C_n \text{ THEN } Y \text{ is } C_{n+1}$$

Finally, the used fitness function is defined as

$$F'(C) = F(C) \cdot LNIR(C) = \Psi_{E_N}(C) \cdot G_\omega(C) \cdot g_n(C^-) \cdot LNIR(C)$$

5.4. (μ, λ)-ES for Identification of TSK Consequents

Let LP_r^x be the fuzzy input subspace of the local fuzzy prototype LP_r. This process, based on the (μ, λ)-ES presented in Ref. 31, computes the existing partial linear input–output relation L_r (TSK rule consequent) of the data located in LP_r^x. This evolutionary process is applied on every $LP_r \in B^i$ in the Step 5 of the identification algorithm (see Section 5.2). The main aspects of this process are described in the following (for more information about the (μ, λ)-ES, refer to Ref. 30).

Coding scheme. The $\vec{x}$ part of the μ individuals in the ES population is composed of the $n + 1$ values defining the TSK consequent, $\vec{x} = p_1, p_2, \ldots, p_n, p_0$ (see Section 2.1). However, due to the infinite nature of such values, their intervals of performance cannot be defined. This problem is solved by the *angular coding*, presented in Ref. 31. It is based on coding the values of the angles instead of the tangent ones for each TSK rule consequent parameter (including p_0) that provide us fixed variable intervals, $(-(\pi/2), \pi/2)$, and a whole representation of the solution search space. Therefore, to code the parameter values, the arc tangent function is used (Figure 4 shows some examples):

$$AC : \mathbb{R} \rightarrow \left(-\frac{\pi}{2}, \frac{\pi}{2}\right); \qquad AC(x) = \arctan(x)$$

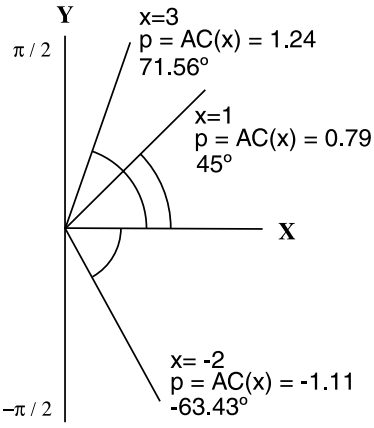


Figure 4. Examples of angular coding.

Initial pool. To generate the initial population, the following indices and the set E_θ must be computed[b]:

$$y_{med} = \frac{\sum_{e_l \in E_N} y^l}{|E_N|}; \qquad y_{min} = \min_{e_l \in E_N} \{y^l\}; \qquad y_{max} = \max_{e_l \in E_N} \{y^l\}$$

$$h_{max} = \max_{e_l \in E_N} \{h_l\}; \qquad E_\theta = \{e_l \in E_N \text{ s.t. } h_l \geq \theta \cdot h_{max}\}$$

Then, the initial population is generated in three different steps:

1. One individual is generated setting the parameters p_i, $i = 1, \ldots, n$, to zero and p_0 to the angular coding of y_{med}.
2. γ individuals are generated, with $\gamma \in \{0, \ldots, \mu - 1\}$ being provided by the GFRBS designer, setting the parameters p_i to zero and p_0 to the angular coding of a random value in $[y_{min}, y_{max}]$.
3. The remaining $\mu - \gamma - 1$ individuals are generated by setting the parameters p_i to small angular values computed in the interval $(-(\pi/2), \pi/2)$ and p_0 to the angular coding of a value computed from a randomly selected element e in E_θ in such a way that e belongs to the hyperplane defined by the TSK rule consequent. Thus, parameters p_i are set by using the function $f(x, z) = z \cdot (\pi/2) \cdot x^q$, with x and z generated at random in $[0, 1]$ and $\{-1, 1\}$, respectively, q is provided by the GFRBS designer, and p_0 is set to the angular coding of $y - \sum_{k=1}^{n} AC^{-1}(p_k) \cdot x_k$, with $AC^{-1}(\beta) = \tan(\beta)$.

Fitness function. The fitness function is composed of a local error measure, computed on the set of examples $E' = \{e_l \in E_N \text{ s.t. } LP_r^x(ex_l) > 0\}$—the examples in the input fuzzy subspace defined by the prototype antecedent:

$$\sum_{e_k \in E'} h_k \cdot (ey^k - S(ex^k))^2$$

with $h_k = T(LP_r^{x_1}(ex_1^k), \ldots, LP_r^{x_n}(ex_n^k))$, and $S(ex^k)$ being the output of the TSK rule with the ex^k input.

6. GFRBS II: POSTPROCESSING STAGE

It is possible that the iterative nature of the identification stage may cause the appearance of an overfitting phenomenon. This happens when some examples are covered to a higher degree than the desired one and it makes the obtained fuzzy rule base perform worse because of the existence of redundant rules. Rule selection methods[38–41] directly select a subset of rules from a given fuzzy rule set to minimize the number of rules while at the same time maintaining (or even improving) system performance. Redundant and inefficient rules that degrade the performance are eliminated, thus obtaining a more cooperative fuzzy rule set and therefore

[b]With $\theta \in [0.5, 1]$ being defined by the GFRBS designer.

involving a potential improvement of system accuracy. Moreover, in many cases accuracy is not the only requirement of the model, and interpretability becomes an important aspect. Reducing model complexity is a way to improve system readability; that is, a compact system with a low number of rules requires minor effort to be interpreted.

On the other hand, due to the same reason, the latter stage induces competition among rules and global cooperation is slightly considered. Improving the way in which the fuzzy rule base performs interpolative reasoning by inducing better cooperation among the rules would make the FRBS more accurate.

To solve these problems and to improve system accuracy, a postprocessing stage has been developed. This consists of two evolutionary processes. The first one performs a rule selection on the previously identified TSK fuzzy rule base, removing inefficient and/or redundant rules in order to improve the cooperation among them. The second one tunes the simplified fuzzy rule base parameters (membership functions and consequent parameters). It will enable rational behavior and improved performance of the FRBS and is a necessary condition to solve complex problems when an accurate solution is required. In the following subsections both processes are presented.

6.1. Genetic Niching-Based Selection Process

This process, presented in Ref. 39, performs an iterative generation of compact fuzzy rule bases, $B^{s_j}, j = 1, \ldots, S$, with S being the number of required different solutions. It is based on the *Sequential Niche Technique*[42] and a *basic genetic selection process*.[38] In each iteration, this basic process is applied, penalizing the fitness function of the individuals close to the solutions from previous iterations (those in the same niche). It is based on a standard binary-coded Genetic Algorithm (GA), whose characteristics are presented in the following. The general scheme for this basic GA is showed in Figure 5.

Coding scheme and initial pool. Let B^i be the TSK rule set derived from the previous step, $m = |B^i|$, and let B^s be a subset of B^i. Each chromosome is a binary string $C = (c_1, \ldots, c_m)$ representing a subset B^s of candidate rules to constitute the set of rules finally obtained, B^{s_j}. In this way, $c_i = 1$ denotes that $R_i \in B^s$ and $c_i = 0$ denotes that $R_i \notin B^s$.

The whole initial gene pool is generated at random, but one individual with all $c_i = 1$ is included in the initial population representing the complete previously obtained rule set B^i.

Fitness function. The fitness function is based on the mean square error (MSE) over the example set E_N and on the completeness property (notice that $R_j(e_l)$ will be the covering degree of the rule R_j over the antecedents of the example e_l). Let τ be the minimum training set completeness degree accepted (given by the GFRBS designer), M be the number of rules, $C_{R(C_j)}(e_l) = \bigcup_{j=1,\ldots,M} R_j(e_l)$, and $R(C_j)$ be the set of rules coded in C_j. The completeness degree of $R(C_j)$ over E_N is $CD(R(C_j), E_N) = \bigcap_{e_l \in E_N} C_{R(C_j)}(e_l)$. The final fitness function penalizing the lack of completeness is

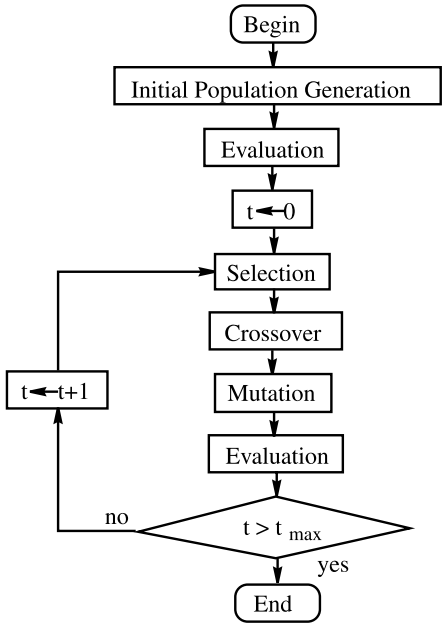


Figure 5. Flowchart of a standard GA.

$$F(C_j) = \begin{cases} \dfrac{1}{2|E_N|} \displaystyle\sum_{e_l \in E_N} (y^l - S(x^l))^2, & \text{if } CD(R(C_j), E_N) \geq \tau \\ \infty, & \text{otherwise} \end{cases}$$

with $S(x^l)$ being the output value of the FRBS using the set of rules coded in $R(C_j)$ when the input variable values are $x^l = (x_1^l, \ldots, x_n^l)$ and y^l is the known desired value obtained from e_l.

However, to consider the niche concept, a modified fitness function F' is needed. This function is obtained by using a function $G(C_j, S)$ that penalizes the closeness of C_j to the set S of the previously obtained solutions. Thus, the modified fitness function F' is defined as $F'(C_j) = F(C_j) \cdot G(C_j, S)$ and the penalization function G as

$$G(C_j, S) = \begin{cases} \infty, & \text{if } d = 0 \\ 2 - \left(\dfrac{d}{r}\right)^{\beta}, & \text{if } d < r \text{ and } d \neq 0 \\ 1, & \text{otherwise} \end{cases}$$

with r being the niche radius, β the *power factor* determining how concave or convex the penalization curve is (both defined by the GFRBS designer), and with

$d = \min_{s_i \in S}\{H(C_j, s_i)\}$ being the Hamming distance to the closest solution in S with $H(C_j, s_i)$ defined as $H(C_j, s_i) = \sum_{i=1}^{m} |C_{ji} - s_{ji}|$.

Selection scheme and genetic operators. The selection procedure is Baker's stochastic universal sampling[43] together with the elitist selection. The genetic operators are the classical two-point crossover and the uniform mutation.

6.2. Genetic Tuning Process

This method is an adaptation of that proposed in Ref. 31 to tune TSK FRBSs based on global semantics. To consider a local semantics approach, changes in the coding scheme and in the computation of the variation intervals have to be done, but the remaining characteristics will remain unchanged. This genetic tuning process adjusts the antecedent membership function definitions of the fuzzy rules obtained from the previous stage as well as their consequent parameters, trying to improve global FRBS performance by minimizing a global error measure over the example set.

It is based on a hybrid GA-ES algorithm in which each individual represents a complete knowledge base. A $(1 + 1)$-ES is considered as a genetic operator to locally tune a percentage δ of the best individuals in each generation. The GA components are presented in the following.

Coding scheme. The individuals are composed of two well-differentiated parts, C^1 and C^2, which code the definition of the antecedent membership functions and the consequent parameters, respectively. The C^1 part is an array built by joining the partial representations of the antecedents of each one of the m fuzzy rules composing the FRBS, C_i^1, with n being the number of input variables, such that

$$C_i^1 = (a_{i1}, b_{i1}, c_{i1}, \ldots, a_{in}, b_{in}, c_{in}), \quad i = 1, \ldots, m$$

$$C^1 = C_1^1 C_2^1 \ldots C_m^1$$

Before the tuning process starts, the variation intervals for each fuzzy set coded in the first part of any chromosome are calculated from the preliminary FRBS as

$$[D_{ij}^{min}, D_{ij}^{max}] = \left[a_{ij} - \frac{b_{ij} - a_{ij}}{2}, c_{ij} + \frac{c_{ij} - b_{ij}}{2}\right], \quad i = 1, \ldots, m, \text{ and } j = 1, \ldots, n$$

In the same way, C^2 is obtained by joining the consequent parameters of each one of the m fuzzy rules in the TSK fuzzy system, C_i^2. These parameters are coded considering the *angular coding* (see Section 5.4), by which each gene in C^2 has the same variation interval $(-(\pi/2), \pi/2)$. Finally, C^2 is defined as

$$C_i^2 = (p_{i0}, p_{i1}, \ldots, p_{in}), \quad i = 1, \ldots, m$$

$$C^2 = C_1^2 C_2^2 \ldots C_m^2$$

Pool initialization. Let M be the population size; the available knowledge contained in the preliminary FRBS is used to initialize these M individuals in three different ways:

1. The preliminary knowledge base is directly coded in the first chromosome.
2. The genes of the first part, C^1, of the following $(M/2)-1$ chromosomes are initiated at random in their respective variation intervals, taking the genes of the second part, C^2, from the preliminary knowledge base.
3. The remaining $M/2$ are generated completely at random, with the values in C^2 generated from a normal distribution $N(0,d)$.

In this initialization process, the variation intervals for each gene a_{ij}, b_{ij}, or c_{ij} of a fuzzy set C_{ij}^1, defined by (a_{ij},b_{ij},c_{ij}) in C^1, are computed as

$$[a_{ij}^l, a_{ij}^r] = \left[a_{ij} - \frac{b_{ij}-a_{ij}}{2}, a_{ij} + \frac{b_{ij}-a_{ij}}{2} \right]$$

$$[b_{ij}^l, b_{ij}^r] = \left[b_{ij} - \frac{b_{ij}-a_{ij}}{2}, b_{ij} + \frac{c_{ij}-b_{ij}}{2} \right]$$

and

$$[c_{ij}^l, c_{ij}^r] = \left[c_{ij} - \frac{c_{ij}-b_{ij}}{2}, c_{ij} + \frac{c_{ij}-b_{ij}}{2} \right]$$

These intervals are only used in this initialization process.

Operators and fitness function. The stochastic universal sampling[43] is considered together with the elitist selection as in the previous stage. The fitness function is the MSE on the example set E_N, represented by the following expression:

$$F(C_j) = \frac{1}{2|E_N|} \sum_{e_l \in E_N} (y^l - S(x^l))^2$$

The offspring population is generated by using the max-min-arithmetical crossover[44] and Michalewicz's nonuniform mutation.[45] If $C_v = (c_1,\ldots,c_k,\ldots,c_H)$ and $C_w = (c'_1,\ldots,c'_k,\ldots,c'_H)$ are going to be crossed, the resulting descendents of the max-min-arithmetical crossover are the two best of the next four offspring:

$$O_1 = aC_w + (1-a)C_v$$

$$O_2 = aC_v + (1-a)C_w$$

$$O_3 \text{ with } c_{3k} = \min\{c_k, c'_k\}$$

$$O_4 \text{ with } c_{4k} = \max\{c_k, c'_k\}$$

with a being a constant parameter chosen by the GFRBS designer.

In the case of the Michalewicz's nonuniform mutation, a gene c_k, with a variation interval $[c_{kl}, c_{kr}]$, is mutated as $c'_k = c_k + \Delta(t, c_{kr} - c_k)$ with probability 0.5 or as $c'_k = c_k - \Delta(t, c_k - c_{kl})$ otherwise. With t being the current generation, the function $\delta(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\delta(t, y)$ being close to 0 increases with the number of generations. This function is formulated as $\delta(t, y) = y(1 - r^{(1-(t/T))^b})$, with r being a random number in $[0,1]$, T the total number of generations, and b being selected by the user to determine the dependency with t.

With D_{ij}^{min} and D_{ij}^{max} being the extreme values of the variation interval of a fuzzy set C_{ij}^1, defined by (a_{ij}, b_{ij}, c_{ij}) in C^1, the intervals of performance for each gene a_{ij}, b_{ij}, or c_{ij} of C_{ij}^1 in the mutation process will be computed as

$$a_{ij} \in [a_{ij}^l, a_{ij}^r] = [D_{ij}^{min}, b_{ij}]$$

$$b_{ij} \in [b_{ij}^l, b_{ij}^r] = [a_{ij}, c_{ij}]$$

$$c_{ij} \in [c_{ij}^l, c_{ij}^r] = [b_{ij}, D_{ij}^{max}]$$

Evolution strategy. As an additional genetic operator, a $(1 + 1)$-ES is considered with the same coding scheme and the same fitness function as those in the GA. For the first part of the chromosome C^1, the $(1 + 1)$-ES works in a way similar to that presented in Section 5.3. However, there is a little difference when C^2 is considered, because an incremental adaptation of the parameters is not necessary in this case and the ES works in the usual way.[30] Moreover, because all the C^2 components are defined on the same interval of performance, $(-(\pi/2), \pi/2)$, the same step size is used for all of them (with $s_i = 0.00001$).

7. EXPERIMENTAL STUDY

In this section, we analyze the behavior of the local semantics-based TSK two-stage GFRBS (in the following called Local Evolutionary Learning of TSK, LEL$_{\rightarrow}$TSK), when solving two different real-world problems. The first is an electrical engineering distribution problem[46] and the second is the *Sunspots* time series prediction.[47] Some well-known methods from the literature are considered for comparison purposes. We have chosen different methods aiming to obtain highly accurate FRBSs, instead of very interpretable ones. Table II shows a summary of their main characteristics.

On the other hand, in order to see some interesting characteristics of the proposed method, an internal comparison (with some methods within the MOGUL paradigm) has been performed considering the first problem. The methods used will be introduced in Section 7.1.1.

Unless we specify something else, the initial linguistic partitions considered by some of the proposed methods are comprised by *five linguistic terms* with triangular-shaped fuzzy sets giving meaning to them. The same values for all the related parameters have been considered (i.e., 0.6 as crossover probability in all the genetic approaches). Finally, the values of the parameters considered for LEL$_{\rightarrow}$TSK are the following:

Table II. Summary on the methods considered for comparison with other techniques.

Ref.	Method	FRBS type	Semantics	FP	Algorithm type	MOGUL
3	TS	TSK	G		AH + LS	
35	LT	TSK	G	✓	GA	
31	M-TSK	TSK	G		AH + ES + GA	✓
27	ANFIS	TSK	G		BNN + LS	
10	FMID	TSK	L	✓	FC	
48,49	GRNN	GRNN	L		AH	

FP: methods including output behavior in the fuzzy partition, MOGUL: methods considering the MOGUL paradigm, G: global, L: local, AH: ad hoc method, GA: genetic algorithm, ES: evolutionary strategy, GRNN: generalized regression neural network, LS: least squares estimation, BNN: backpropagation neural network, FC: fuzzy clustering-based method.

- *Local process for identifying prototypes*: $\epsilon = 1.5$, $\omega = 0.05$, and $k = 0.1$ in the fitness function; $c = 0.9$ and 100 iterations for the $(1+1)$-ES; $\mu = 15$, $\lambda = 100$, $\gamma = 0.2 \cdot \mu = 3$, $\theta = 0.7$, $q = 5$, $\vec{r} = (r_{\vec{x}}, r_{\vec{\sigma}}, r_{\vec{\alpha}}) = (2,0,0)$, $\vec{\zeta} = (\zeta_{\vec{x}}, \zeta_{\vec{\sigma}}, \zeta_{\vec{\alpha}}) = (\mu, \mu, 1)$, $(n_\sigma, n_\alpha) = (0,0)$, and 500 iterations for the (μ, λ)-ES.
- *Genetic niching-based selection process*: Three solutions, $\tau = 1.5$, $r = 2.5\%$ of the number of rules in the preliminary set of rules $(0.25 \cdot m)$, $\beta = 0.5$, $N = 61$, $P_c = 0.6$, $P_m = 0.1$, and 500 generations.
- *Genetic tuning process*: $M = 61$, $P_c = 0.6$, $P_m = 0.1$, $a = 0.35$, $b = 5$, $d = 0.001$, 1000 generations, 25 $(1+1)$-ES iterations, $\alpha = 0$ and $c = 0.9$ (the updating amount of Rechenberg's 1/5-success rule in the $(1+1)$-ES[30]).

To see that the proposed method takes a reasonable computing time (to solve off-line data-driven learning problems), we have included the computing time of the methods considered for comparison in the tables of results of both problems. Notice that we have used a Pentium 4 with a CPU to 2.40 GHz to run the methods and that the computing times have been rounded to minutes.

7.1. Estimating the Length of Low Voltage Lines

For an electricity distribution company, it may be of interest to measure the maintenance costs of its own electricity lines. These estimations could be useful to allow them to justify their expenses. However, in some cases these costs cannot be directly calculated. The problem comes when trying to compute them for low voltage lines for the following reasons. Although maintenance costs depend on the total length of the electrical line, the length of low voltage lines would be very difficult and expensive to measure because they are contained in little villages and rural nuclei. The installation of these kinds of lines is often very intricate and, in some cases, one company can serve more than 10,000 rural nuclei.

For this reason, the length of low voltage lines cannot be directly computed. Therefore, it must be estimated by means of indirect models. The problem involves relating *the length in meters of the low voltage line of a certain village* with the following two variables: *the number of users in the village* and *the mean of the distances in meters from the center of the town to the three furthest clients in it*

Table III. Electrical problem characteristics.

Input variable X_1:	Number of users
Input variable X_2:	Village radius
Output variable Y:	Low voltage line length
Number of examples:	495
Domain of X_1:	[1, 320]
Domain of X_2:	[60, 1673.329956]
Range of Y:	[80, 7675]

(village radius).[46] We were provided with the measured line length, the number of inhabitants, and the mean distance from the center of the town to the three furthest clients in a sample of 495 rural nuclei. Table III presents a summary of the main characteristics of this problem.

To evaluate the models obtained from the different methods considered in this article, this sample has been randomly divided into two subsets, a training set with 396 elements and a test set with 99 elements, 80% and 20%, respectively.[c] The existing dependency of the two input variables with the output variable in the training and test data sets is shown in Figure 6 (notice that they present strong nonlinearities).

In this problem, it would be preferable that the solutions obtained verify the following requirement: they have not only to be numerically accurate in the problem solving, but these solutions should be almost locally interpretable by human beings. Therefore, the use of gray-box techniques such as fuzzy modeling is recommended.

7.1.1. Internal Comparison

To see how the identification process affects to the learning, two ad hoc algorithms have been developed for comparison as TSK prototype identification algorithms together with their Mamdani counterparts. These methods are based on two simple Mamdani fuzzy rule generation methods for global and local fuzzy prototyping, respectively (they are presented in Section A of the Appendix). Moreover, to see some interesting characteristics of the proposed method, the counterpart method to obtain local semantics-based Mamdani rules within the MOGUL paradigm (M-SCL[36]) is also analyzed. The M-SCL algorithm has been tested on this problem in previous works, presenting good results in comparison to other related techniques. Table IV shows a summary of the main characteristics of the methods considered.

The results obtained by the considered methods following the MOGUL paradigm[17] are shown in Table V, where $\#R$ stands for the number of rules and MSE_{tra} and MSE_{tst}, respectively, for the mean square error obtained over the training and test data. The best results are highlighted in boldface.

[c]Both data sets considered are available at http://decsai.ugr.es/~casillas/fmlib/.

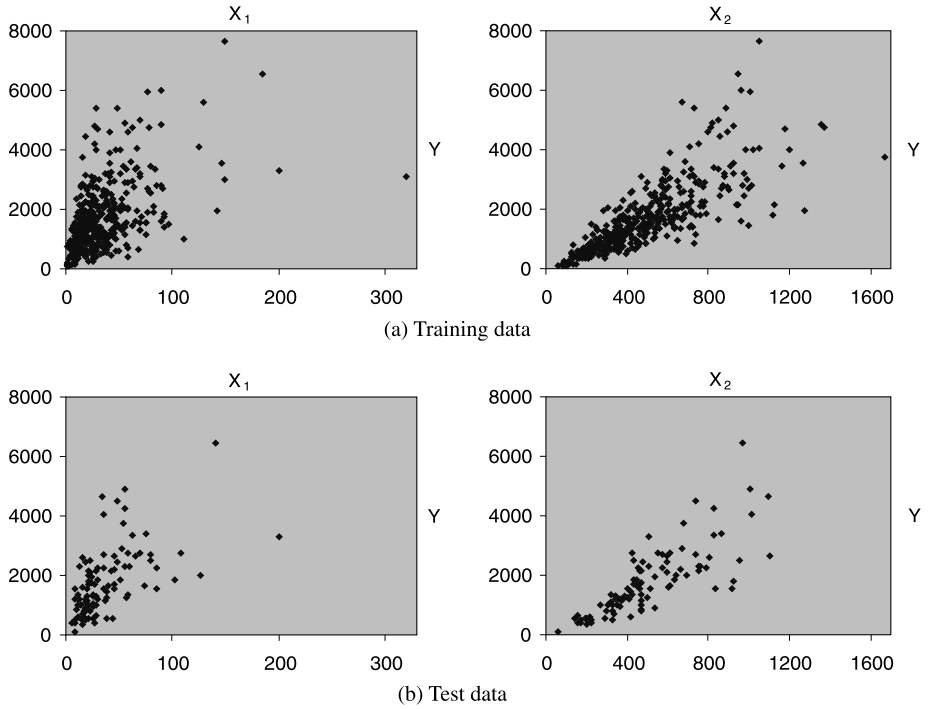


Figure 6. (a) (X_1, Y) and (X_2, Y) dependency in the training data; (b) (X_1, Y) and (X_2, Y) dependency in the test data.

Analyzing the model obtained by our LEL$_{\rightarrow}$TSK method, we can conclude that it presents the best performance in approximation (MSE_{tra}) and generalization (MSE_{tst}), with improvements of about 17.3% in test and 12.7% in training over M-SCL, the method achieving the best generalization capability among the remaining ones considered. Notice that the three models obtained by our LEL$_{\rightarrow}$TSK method are the best ones in the table.

On the other hand, two important aspects should be pointed out. First, there are significant differences among the methods considering Mamdani rules and their

Table IV. Summary on the methods considered for internal comparison.

Ref.	Method	FRBS type	Semantics	FP	Algorithm type	MOGUL
50 + 36	WM + TUN	MAM	G		AH + GA	✓
51 + 52	WCA + TUN	MAM	L		AH + GA	✓
36	M-SCL	MAM	L	✓	IL + ES + GA	✓
—	WM$_{\rightarrow}$TSK	TSK	G		AH + ES + GA	✓
—	WCA$_{\rightarrow}$TSK	TSK	L		AH + ES + GA	✓

For FP, MOGUL, G, L, AH, GA, and ES see Table II; IL: iterative learning.

Table V. Results obtained by the multistage GFRBSs in the electrical problem.

	Identification			Niching-based selection			Tuning	
Method	#R	MSE_{tra}	MSE_{tst}	#R	MSE_{tra}	MSE_{tst}	MSE_{tra}	MSE_{tst}
WM + TUN	13	298446	282058		—		175337	180102
WCA + TUN	20	356434	311195		—		175887	180211
M-SCL	31	431904	435649	19	226403	222550	148036	191339
				20	227261	227105	142108	166578
				16	227232	225789	136826	177612
$WM_{\rightarrow}$TSK	13	167693	155424		—		137942	170707
$WCA_{\rightarrow}$TSK	20	167513	156546		—		128152	191996
$LEL_{\rightarrow}$TSK	31	169354	156165	26	165498	155045	**124022**	**137752**
				25	165152	154593	131473	153237
				26	165162	154640	127958	168778

counterparts considering TSK ones, presenting more or less the same number of rules. Second, the results obtained in training by $WM_{\rightarrow}$TSK and $WCA_{\rightarrow}$TSK are significantly better than the ones with WM + TUN and WCA + TUN. However, they present overfitting, as their respective prototype identification algorithms obtain models with too few rules. The same happens for M-SCL and $LEL_{\rightarrow}$TSK in training, although in this case the improvements are also significant in test.

Therefore, it seems that the selection of the prototype identification algorithm is an important issue and that to choose any Mamdani fuzzy rule generation method is not sufficient.

7.1.2. Comparison with Other Techniques

In this section, we have used the techniques summarized at the beginning of Section 7. In the case of Ts, FMID, and GRNN, the only parameters considered are, respectively, the number of rules, the number of clusters (rules), and the spread of the radial basis functions. All these parameters directly determine the final training performance of the obtained model and the higher their value, the lower the MSE. However, it was the contrary for test data, and the higher their value, the higher the MSE on the test set. In this work, all the integer values from 0 to 100 have been tried for these parameters, taking as a result the one presenting the best value in test (4 as the maximum number of rules for Ts, 5 clusters for FMID, and 48 as the spread value for GRNN). The results obtained by the methods mentioned are presented in Table VI.

Some previous approaches including output behaviors in the fuzzy partitioning (see Section 1) have been considered. In view of these results, we can say that the proposed method outperforms the best results obtained by the remainder, with improvements of about 13.34% in test and 27.32% in training over Ts, the method achieving the best generalization capability except the proposed approach. Moreover, the computing time of the proposed method is reasonable for this sort of problem.

Table VI. Comparison of the models obtained by different methods in the electrical problem.

Method	#R/Complexity	MSE_{tra}	MSE_{tst}	Time (min)
TS	4	170644	158949	2
LT	4	169761	160110	5
M-TSK	20	132917	167826	6
ANFIS	20	**108279**	923650	1
FMID	5	181040	164670	1
GRNN	(396×2) *neur.*	170460	198140	1
LEL$_{\rightarrow}$TSK	26	124022	**137752**	10

Notice that practically all these present methods result in overfitting except when a low number of rules are considered (see the case of ANFIS, in which the number of rules can not be specified by the user). It causes these methods to obtain fuzzy models with poor accuracy. In our case, the obtained models present a good balance between approximation and generalization, even considering a higher number of rules. This fact denotes the good fuzzy partitioning that this method achieves.

7.2. The "Sunspots" Time Series

Sunspots, often larger in diameter than the Earth, are dark blotches on the sun. They were first observed around 1610, shortly after the invention of the telescope.[47] Yearly averages have been recorded since 1700. The sunspot numbers are defined as $k(10g + f)$, where g is the number of sunspot groups, f is the number of individual sunspots, and k is used to reduce different telescopes to the same scale.[53] The observations are shown as black squares in Figure 7. The average time between maxima is 11 years. Notice, however, that the time between maxima ranges from 7 to 15 years.

The underlying mechanism for sunspot appearances is not exactly known. No first principles theory exists, although it is known that sunspots are related to other solar activities. For example, the magnetic field of the sun changes with an average period of 22 years. Sunspots usually appear in pairs, corresponding to magnetic dipoles. Sunspot pairs reverse their polarity from one cycle to the next, reflecting the underlying magnetic cycle. The sunspot series has served as a benchmark in the statistics literature.

The goal of time series prediction can be stated succinctly as follows: given a sequence $y(1), y(2), \ldots y(N)$ up to time N, find the continuation $y(N + 1)$, $y(N + 2)\ldots$. The series may arise from the sampling of a continuous time system and be either stochastic or deterministic in origin.

In time-series prediction, there are two forms to measure the output errors:

- *Single-Step Prediction*: where external inputs to the method are true observed data
- *Iterative Prediction*: where external inputs to the method are predicted outputs from previous iterations.

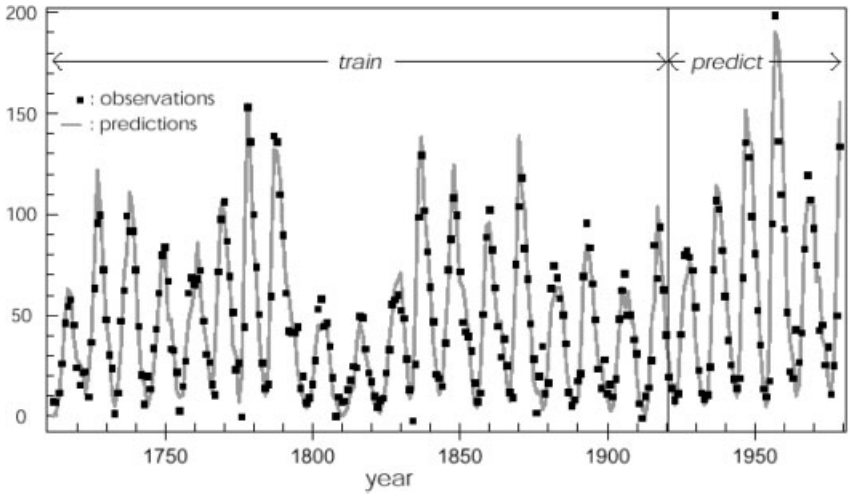


Figure 7. The sunspot data.

Single-step prediction has usually been used in the literature. In this way, we will use single-step prediction to analyze the methods. To estimate the performance of the methods we check the *normalized mean square error* (*nMSE*):

$$nMSE(N) = \frac{\sum_{l=1}^{N} (target_l - prediction_i)^2}{\sum_{l=1}^{N} (target_l - mean(target))^2} = \frac{1}{\sigma^2} \cdot \frac{1}{N} \cdot \sum_{l=1}^{N} (f(L+l) - f'(L+l))^2$$

where $l = 1, \ldots, N$ enumerates the points in the withheld data set, *target* and $f(L + l)$ are the true value of the time series, *prediction* and $f'(L + l)$ are the output of the method, and σ^2 denotes the sample variance of the observed time series in the data set. This kind of fitting error describes how well the points are approximated by the surface over the input space.

Many works try to analyze sunspot data using linear and nonlinear methods. The sunspots of years 1700 through 1920 were chosen to be the training set, and the sunspots of years 1921 through 1994 were chosen for single-step prediction. Errors are calculated considering the following four sets of data:

1. Years 1921 through 1955
2. Years 1956 through 1979
3. Years 1980 through 1994
4. Years 1921 through 1994.

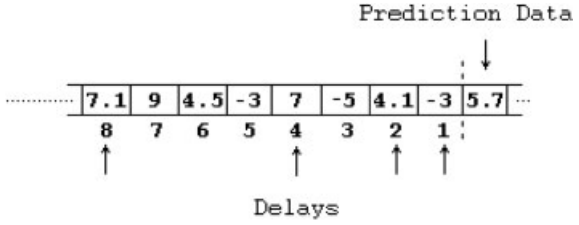


Figure 8. Temporal window.

7.2.1. Adaptation of Time-Series to System Modeling

In this work we have used a temporal window with four delays. These delays are the 1, 2, 4, and 8 delay from the predicted data. In Figure 8, we can see the window considered.

Therefore, we will only use four points but we will cover eight delays (as was proposed in Ref. 54). Because this problem will be solved with fuzzy modeling-based techniques, each delay should be considered as an input variable for these kinds of methods. Following this approach, all the methods consider four input variables.

7.2.2. Comparison with Other Techniques

In this section, we have used an additional method for comparison purposes, WNET[55] (a short description is included in Section B of the Appendix). However, the main aim is not to compete with these kinds of techniques; the main aim is to have a reference point with a classic method for time-series prediction, previously used to solve this problem. Notice that we have not denoted the computing time (min) of WNET because the WNET's results are extracted from Ref. 55 and the authors did not denote it.

As in the case of the first problem, the only considered parameter in Ts, FMID, and GRNN is the number of rules, the number of clusters (rules), and the spread of the radial basis functions, respectively. For these parameters, we have used all the integer values from 0 to 100, taking as a result the one presenting the best value in test (2 as a maximum number of rules for Ts, 5 clusters for FMID, and 15 as spread value for GRNN). In this problem, we have utilized *three and five linguistic terms* in the initial linguistic partitions considered by some of the methods. The results obtained by the methods mentioned are presented in Table VII, where #*L* stands for the number of labels in the input variables, and nMSE_{tra} and nMSE_{tst}, respectively, for the normalized mean square error obtained over the training and test data.

In view of these results, we can say that the proposed method outperforms the best results obtained by the remainder, with improvements of about 28.95% in test and 56.46% in training over WNET, the method achieving the best

Table VII. Comparison of the models obtained by different methods on the sunspot time-series.

			$nMSE_{tra}$	$nMSE_{tst}$				
Method	#L	#R/Complex.	1700–1920	1921–55	1956–79	1980–94	1921–94	Times
TS		2	2.6404	2.6783	3.0586	3.8606	2.8431	3
LT	3	3	0.3489	0.2201	2.2367	1.7651	1.3130	2
ANFIS	3	72	**0.0026**	1448.9	1487.6	4182.6	1917.7	17
M-TSK	3	36	0.5009	0.2132	1.0933	1.1291	0.7312	11
LT	5	1	0.2732	0.2232	1.4733	2.3056	1.1318	2
ANFIS	5	Computing time overwhelming (the method handles 625 rules)						
M-TSK	5	140	0.1466	0.4164	1.2098	1.8161	0.9857	26
FMID		5	0.3318	0.2319	0.3181	0.4715	0.3072	1
GRNN		(213 × 2) *neur.*	1.1097	6.6488	1.3243	8.2819	4.9287	1
WNET		21 *neur.*	0.0820	**0.0860**	0.3500	0.3130	0.2190	—
LEL→TSK	3	15	0.0752	0.0983	0.2599	0.1610	0.1659	12
LEL→TSK	5	57	0.0357	0.1325	**0.2217**	**0.1324**	**0.1556**	27

generalization capability, considering five labels. Notice the good results also obtained by the proposed method considering three labels and only 15 rules.

As in the first problem, practically all these methods present overfitted results except considering a low number of rules. It causes these methods to obtain fuzzy models with a poor accuracy. Moreover, the computing time of ANFIS is overwhelming when we use five labels. In our case, the obtained models present a good balance between approximation and generalization, even considering a higher number of rules. Notice that, when we increase the number of labels, the model obtains better results and the number of rules does not increase so much. This fact denotes the good fuzzy partitioning that this method achieves.

7.3. Method Analysis

To analyze the behavior of the proposed LEL→TSK, we have focused on the problem of estimating the length of low voltage lines for an electricity distribution company. Figure 9 illustrates the evolution chart of the fitness value obtained by the LEL→TSK method considering the most accurate model. The chart depicts the values of the best individual in each generation (for the identification process it is the MSE of the model every time a new rule is added, one iteration). The obtained chart clearly states the convergence power of the LEL→TSK algorithm and allows us to analyze the efficiency of the different processes composing it.

Analyzing the chart, we can observe how the *identification process* appropriately converges, increasing the model performance until the end. The system performance is only decreased in the 10th iteration. However, the corresponding rule is kept after the selection process, considering it as a necessary rule. The *selection process* presents fast convergence and it is the process obtaining the lesser improvements, which is cogent because it is only devoted to removing inefficient or

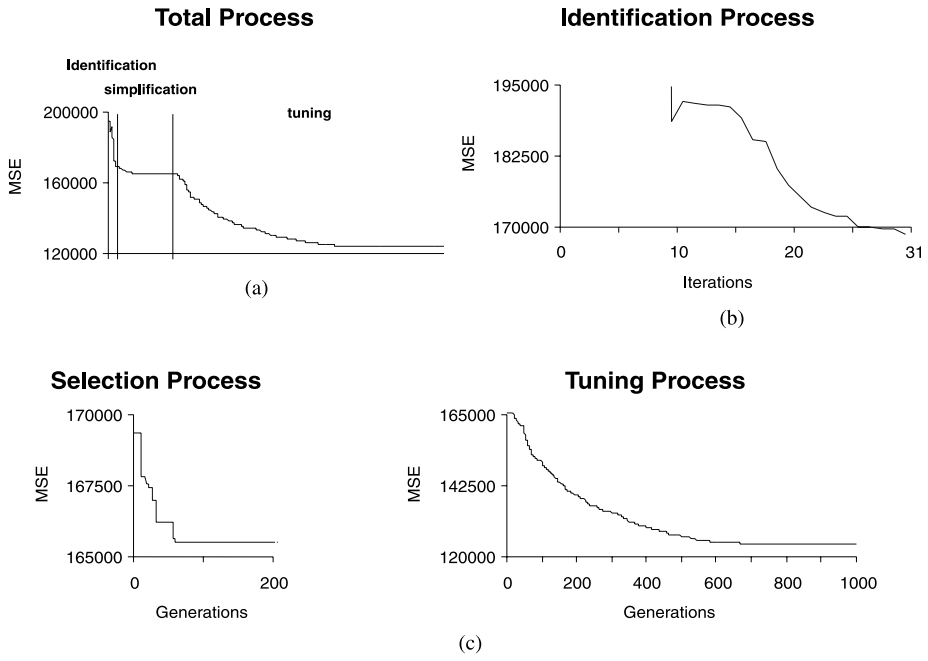


Figure 9. Process convergency to the best solution: (a) the whole learning process; (b) local process for identifying prototypes; (c) postprocessing stage.

redundant rules for the tuning process. Finally, the *tuning process* gradually improves system performance (practically until the end), presenting significant improvements.

Figure 10 graphically depicts the best TSK model obtained from the proposed method. Each rule in the figure can be interpreted as a local linear model acting on its definition subspace. Notice that some very specific rules have been obtained. These kinds of rules are devoted to model points that are quite different than their neighbors (outliers or noise points) allowing us to decrease the generalization error.

8. CONCLUDING REMARKS

In this work, we propose the use of local fuzzy prototypes as a first approximation to obtain accurate local semantics-based TSK fuzzy rules. A two-stage evolutionary algorithm considering the interaction between input and output variables has been developed following the MOGUL paradigm. First, we obtain an acceptable model as a first approximation by means of a local evolutionary learning performing the identification of local prototypes. Then, a postprocessing stage considers the global cooperation of such promising rules to improve the fuzzy model performance.

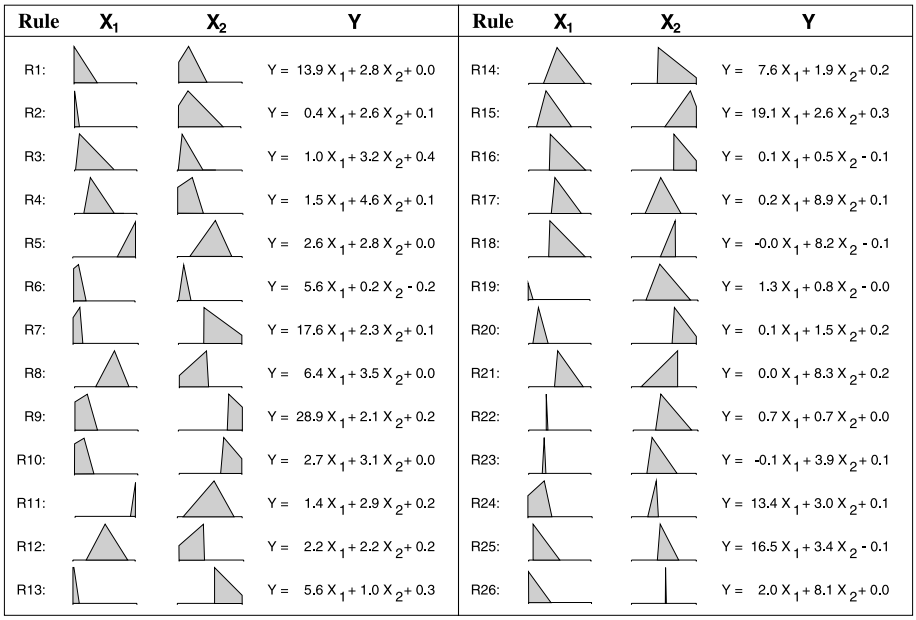


Figure 10. Representation of the best obtained TSK FRBS.

The proposed evolutionary learning method has been compared with other methods to solve two real-world problems. From the results obtained the following conclusions can been drawn:

- The election of the prototype identification algorithm is an important issue, because the quality of the initial promising rules (prototypes) highly depends on this process.
- The use of fuzzy prototypes to identify an initial set of TSK rules allow us to consider the *completeness* and *consistency* properties by means of a sophisticated fuzzy rule generation method. In this way, we can obtain high quality fuzzy rules (prototypes) from which we can generate TSK models presenting a good balance between approximation and generalization.

Acknowledgments

This research has been supported by CICYT Project TIC2002-04036-C05-01.

References

1. Mamdani EH. Application of fuzzy algorithms for control of simple dynamic plant. Proc Inst Electr Eng 1974;121:1585–1588.
2. Sugeno M, Yasukawa T. A fuzzy-logic-based approach to qualitative modeling. IEEE Trans Fuzzy Syst 1993;1:7–31.
3. Takagi T, Sugeno M. Fuzzy identification of systems and its application to modeling and control. IEEE Trans Syst Man Cybern 1985;15:116–132.
4. Wang L, Langari R. Complex systems modeling via fuzzy logic. IEEE Trans Syst Man Cybern B Cybern 1996;26:100–106.

5. Abonyi J. Babuška R. Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models. IEEE Trans Syst Man Cybern B Cybern 2002;32:612–621.
6. Chuang CC, Su SF, Chen SS. Robust TSK fuzzy modeling for function approximation with outliers. IEEE Trans Fuzzy Syst 2001;9:810–821.
7. Chuang CC, Hsiao CC, Jeng JT. Adaptive fuzzy regression clustering algorithm for TSK fuzzy modeling. In: Proc 2003 IEEE Int Symp on Computational Intelligence in Robotics and Automation, Kobe, Japan; 2003. pp 201–206.
8. Linh TH, Osowski S, Stodolski M. On-line heart beat recognition using hermite polynomials and neuro-fuzzy network. IEEE Trans Instrum Meas 2003;52:1224–1231.
9. Höppner F, Klawonn F. Improved fuzzy partitions for fuzzy regression models. Int J Approx Reason 2003;32:85–102.
10. Babuška R. Fuzzy modeling for control. Boston, MA: Kluwer Academic; 1998.
11. Delgado M, Gómez-Skarmeta AF, Martín F. A fuzzy clustering based rapid-prototyping for fuzzy rule-based modeling. IEEE Trans Fuzzy Syst 1997;5:223–233.
12. Kumar A, Agrawal DP, Joshi SD. A GA-based method for constructing TSK fuzzy rules from numerical data. In: Proc 12th IEEE Int Conf on Fuzzy Systems (FUZZ'03); 2003. pp 131–136.
13. Ménard M. Fuzzy clustering and switching regression models using ambiguity and distance rejects. Fuzzy Set Syst 2001;122:363–399.
14. Ruspini EH. A new approach to clustering. Inform Contr 1969;15:22–32.
15. Bezdek JC. Fuzzy clustering. In: Ruspini EH, Bonisone PP, Pedrycz W, editors. Handbook of computation. Amsterdam: Institute of Physics Press; 1998.
16. Cordón O, Herrera F, Hoffmann F, Magdalena L. Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases. Singapore: World Scientific; 2001.
17. Cordón O, del Jesus MJ, Herrera F, Lozano M. MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. Int J Intell Syst 1999;14:1123–1153.
18. Pal N, Eluri VK, Mandal GK. Fuzzy logic approaches to structure preserving dimensionality reduction. IEEE Trans Fuzzy Syst 2002;10:277–286.
19. Yen J, Wang L. Simplifying fuzzy rule-based models using orthogonal transformation methods. IEEE Trans Syst Man Cybern B Cybern 1999;29:13–24.
20. Setnes M, Babuška R, Kaymak U, van Nauta Lemke HR. Similarity measures in fuzzy rule base simplification. IEEE Trans Syst Man Cybern B Cybern 1998;28:376–386.
21. Fiordaliso A. A constrained Takagi-Sugeno fuzzy system that allows for better interpretation and analysis. Fuzzy Set Syst 2001;118:307–318.
22. Yen J, Wang L, Gillespie CW. Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Trans Fuzzy Syst 1998;6:530–537.
23. Casillas J, Cordón O, Herrera F, Magdalena L, editors. Interpretability issues in fuzzy modeling. Studies in Fuzziness and Soft Computing, vol. 128. Heidelberg: Springer; 2003.
24. Casillas J, Cordón O, Herrera F, Magdalena L, editors. Accuracy improvements in linguistic fuzzy modeling. Studies in Fuzziness and Soft Computing, vol. 129. Heidelberg: Springer; 2003.
25. Pedrycz W, Reformat M. Rule-based models of multivariable functions. Fuzzy Set Syst 1997;90:235–253.
26. Hoffmann F, Nelles O. Genetic programming for model selection of TSK-fuzzy systems. Inform Sci 2001;136:7–28.
27. Jang JR. ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 1993;23:665–684.
28. Pomares H, Rojas I, González J, Prieto A. Structure identification in complete rule-based fuzzy systems. IEEE Trans Fuzzy Syst 2002;10:349–359.
29. Gustafson D, Kessel W. Fuzzy clustering with a fuzzy covariance matrix. In: Proc IEEE CDC, San Diego; 1979. pp 761–766.
30. Bäck T. Evolutionary algorithms in theory and practice. Oxford: Oxford University Press; 1996.

31. Cordón O, Herrera F. A two-stage evolutionary process for designing TSK fuzzy rule-based systems. IEEE Trans Syst Man Cybern B Cybern 1999;29:703–715.
32. Papadakis SE, Theocharis JB. A GA-based fuzzy modeling approach for generating TSK models. Fuzzy Set Syst 2002;131:121–152.
33. Wai RJ. Robust fuzzy neural network control for nonlinear motor-toggle servomechanism. Fuzzy Set Syst 2003;139:185–208.
34. Bastian A. How to handle the flexibility of linguistic variables with applications. Int J Uncertainty Fuzziness Knowl Base Syst 1994;2:463–484.
35. Lee MA, Takagi H. Integrating design stages of fuzzy systems using genetic algorithms. In: Proc Second IEEE Int Conf on Fuzzy Systems (FUZZ-IEEE'93), San Francisco; 1993. pp 613–617.
36. Cordón O, Herrera F. A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases. Int J Approx Reason 1997;17:369–407.
37. González A, Pérez R. Completeness and consistency conditions for learning fuzzy rules. Fuzzy Set Syst 1998;96:37–51.
38. Herrera F, Lozano M, Verdegay JL. A learning process for fuzzy control rules using genetic algorithms. Fuzzy Set Syst 1998;100:143–158.
39. Cordón O, Herrera F. Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems. Fuzzy Set Syst 2001;118: 235–255.
40. Ishibuchi H, Nozaki K, Yamamoto N, Tanaka H. Selecting fuzzy if-then rules for classification problems using genetic algorithms. IEEE Trans Fuzzy Syst 1995;9:260–270.
41. Ishibuchi H, Murata T, Türksen IB. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Set Syst 1997;89: 135–150.
42. Beasley D, Bull DR, Martin RR. A sequential niche technique for multimodal function optimization. Evol Comput 1993;1:2:101–125.
43. Baker JE. Reducing bias and inefficiency in the selection algorithm. In: Grefenstette JJ, editor. Proc Second Int Conf on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum; 1987. pp 14–21.
44. Herrera F, Lozano M, Verdegay JL. Tackling real-coded genetic algorithms: Operators and tools for the behaviour analysis. Artif Intell Rev 1998;12:265–319.
45. Michalewicz Z. Genetic algorithms + data structures = evolution programs. Berlin: Springer-Verlag; 1996.
46. Cordón O, Herrera F, Sánchez L. Solving electrical distribution problems using hybrid evolutionary data analysis techniques. Appl Intell 1999;10:5–24.
47. Foukal PV. The variable sun. Sci Am 1990;262:34–41.
48. Demuth H, Beale M. Neural network toolbox for use with MATLAB. User's guide. Version 3.0. Natick, MA: The MathWorks, Inc.; 1998.
49. Wasserman PD. Advanced methods in neural computing. New York: Van Nostrand Reinhold; 1993. pp 155–161.
50. Wang LX, Mendel JM. Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 1992;22:1414–1427.
51. Bárdossy A, Duckstein L. Fuzzy rule-based modeling with application to geophysical, biological and engineering systems. Boca Raton, FL: CRC Press; 1995.
52. Herrera F, Lozano M, Verdegay JL. Tuning fuzzy logic controllers by genetic algorithms. Int J Approx Reason 1995;12:299–315.
53. Marple SL. Digital spectral analysis with applications. Englewood Cliffs, NJ: Prentice-Hall; 1987.
54. Angeline PJ. Evolving predictors for chaotic time series. In: Proc SPIE: Application and Science of Computational Intelligence (SPIE'98), Bellingham, WA; 1998. pp 170–180.
55. Weigend A, Huberman B, Rumelhart D. Predicting the future: A connectionist approach. Int J Neur Syst 1990;7:403–430.
56. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. Parallel Distr Process 1986;1:318–362.

APPENDIX: LEARNING METHODS CONSIDERED IN THE EXPERIMENTAL STUDY

A.1. Ad Hoc Learning Methods

Four ad hoc methods have been developed to study some interesting characteristics of the proposed learning process:

- WM$_{\rightarrow}$TSK: A two-stage GFRBS composed of Wang and Mendel's method[50] as a rapid identification of global fuzzy prototypes, the linear consequent derivation proposed in this article (see Step 5 of the local process for identifying prototypes in Section 5), and the global semantics-based TSK genetic tuning process presented in Ref. 31.
- WCA$_{\rightarrow}$TSK: A two-stage GFRBS composed of the Weighted Counting Algorithm[51] as a rapid identification of local fuzzy prototypes, the linear consequent derivation proposed in this article (see Step 5 of the local process for identifying prototypes in Section 5), and the local semantics-based TSK genetic tuning process proposed in this article (see Section 6.2).
- WM + TUN: A two-stage GFRBS to obtain global semantics-based Mamdani fuzzy rules. First, the primary rule in importance is obtained in each fuzzy input subspace considering Wang and Mendel's well-known ad hoc generation method.[50] Then, the global semantics-based genetic tuning process presented in Ref. 36 (a real-coded GA) is used to adjust the parameters of the membership functions. The number of labels for each variable is determined by the user.
- WCA + TUN: A two-stage GFRBS to obtain local semantics-based Mamdani fuzzy rules. In a way similar to the previous algorithm, the Weighted Counting Algorithm[51] is considered to obtain a preliminary set of rules, and the local semantics-based genetic tuning process presented in Ref. 52 (a real-coded GA) adjusts the parameters of the membership functions.

A.2. Learning Methods from Other Authors

Eight methods from other authors have been considered for comparison (in most cases with accuracy purposes):

- TS: This was the first approach to learning global semantics-based TSK FRBSs.[3] This method assumes some input variables and some initial premise parameters, which are iteratively obtained by a heuristic combinatorial search. Consequent parameters are optimally adjusted with respect to these premises by means of a least squares estimation algorithm and, finally, the premise parameters are readjusted. This is accomplished considering the simplex algorithm, which is based on a nonlinear optimization method. The maximum number of rules is determined by the user.
- LT: A single-stage GFRBS following a global semantics approach.[35] It is based on a binary-coded GA that jointly performs the derivation of the premise parameters and the learning of the consequent linear parameters, the number of rules, and the number of labels per variable. However, in our implementation we make use of a real coding scheme because it is more efficient when real values are considered. The maximum number of labels in the variables is defined by the user.
- ANFIS: A neural FRBS presented by Jang[27] to obtain global semantics-based TSK FRBSs. The TSK fuzzy inference system is represented in a layered, feedforward network structure. Jang proposed a two-pass algorithm to adjust the parameters using a modified error-backpropagation optimization algorithm. In the forward pass, the premise parameters are held fixed and the consequent parameters are adjusted by least squares estimation. In the backward pass, the network error is backpropagated through the network and the

premise parameters are adjusted by gradient descent while the consequent parameters are held fixed. The number of labels in the variables is defined by the user. In our implementation we consider triangular membership functions. Nevertheless, the method behavior is very similar to the original one considering Gaussian membership functions.

- WNET: This method, presented by Weigend et al.,[55] builds a feedforward network with backpropagation (the error backpropagation algorithm of Rumelhart et al.[56]) and weight elimination.
- FMID: This method builds input–output TSK fuzzy models from data by means of the Gustafson-Kessel fuzzy clustering method.[10,29] It uses fuzzy clustering in the product space of the premises and the consequents to approximate a nonlinear system by a collection of local linear models. Each local model then corresponds to a fuzzy rule of the TSK type. The number of clusters (local semantics-based TSK fuzzy rules) is chosen by the user.
- GRNN: A method to perform function approximation with a generalized regression neural network,[48,49] those having a radial basis layer and a special linear layer. It finds the smallest network that can solve the problem within a given error goal. The *spread* of the radial basis functions is considered as a learning factor. The larger the *spread*, the smoother the function approximation will be. To fit data closely, a *spread* smaller than the typical distance between input vectors should be considered. To fit the data more smoothly, a larger *spread* should be set. GRNN sets the first layer weights to the transposed input vectors (training data premises), and the first layer biases are all set to 0.8326/*spread*. The second layer weights are set to the target vectors (training data consequents).
- M-SCL: A two-stage GFRBSs based on the Soft Constrained Learning presented in Ref. 36 to obtain local semantics-based Mamdani fuzzy rules. The first stage is composed of an *evolutionary generation process* very close to the local identification algorithm proposed in this article—without the linear consequent derivation. The second stage (postprocessing) is composed of a genetic niching-based selection process and a genetic tuning stage.
- M-TSK: A two-stage GFRBS presented in Ref. 31 to obtain global semantics-based TSK rules. It is based on an iterative rule generation process and a postprocessing stage performing a genetic niching-based selection and a global semantics-based genetic tuning. The number of labels considered in the premises is determined by the user.