Xfhl: A Tool for the Induction of Hierarchical Fuzzy Systems

Sergio Cala and Francisco J. Moreno-Velo

Abstract— **This paper presents a new tool, called *Xfhl*, for the automatic generation of hierarchical fuzzy systems. The tool takes a training dataset as input and finds the best hierarchical decomposition in terms of fuzzy modules with two inputs and one output. The behavior of *Xfhl* is based on an exhaustive search among all possible module decompositions. In order to evaluate each structure, *Xfhl* uses the mean square error on the structure after a process of parametric optimization. The tool is integrated into Xfuzzy 3, a development environment for fuzzy systems.**

I. Introduction

As far as fuzzy systems have been strengthened as a good way for modeling nonlinear behavior, the complexity of the problems in discussion has been increasing. Regarding standard fuzzy systems, a problem with *n* input variables and *m* linguistic labels for each one is described by a fuzzy system with m^n rules. This exponential increase results in a lack of interpretability when the number of variables is large. This problem, which is not unique to fuzzy systems, is known as "*the curse of dimensionality*" [1].

One way to reduce the number of rules and, therefore, to increase the interpretability, is to decompose the fuzzy system into a structure of simpler modules, which is known as a hierarchical fuzzy system (HFS). There are many proposals for the design of such systems [2]. Some of them consist in identifying common parts of the set of rules and create modules that generate these common parts, which is known as *clean* HFS [3] [4]. In other proposals the hierarchy level of each module refers to an increase in the granularity of the variables [5].

The most traditional kind of HFS is that in which each module is a complete fuzzy system relating a reduced set of variables. In this structure, the input variables of each module may be either global input variables or inner variables (which must be generated as outputs of other modules) [6]. For this kind of HFS the number of rules is

$$Number \quad of \quad rules = \sum_{i=1}^{L} m^{ci} , \qquad (1)$$

This work was supported in part by the Spanish CICYT Projects TEC2008-04920/TEC, and TIN2008-06681-C06-06, and by the Projects P07-TIC-03179 and P08-TIC-03674 from the Andalusian Regional Government.

S. Cala is a student of the Máster en Tecnologías Informáticas Avanzadas, Universidad de Huelva, Huelva, SPAIN (e-mail: sergio.cala@alu.uhu.es).
F. J. Moreno-Velo is with the Departamento de Tecnologías de la Información, Universidad de Huelva, Huelva, SPAIN (e-mail: francisco.moreno@dti.uhu.es).

where *L* is the number of modules and *ci* is the number of inputs for the i-th module. If different modules can share an input variable, then the HFS has the shape of an acyclic directed graph. Otherwise, the structure of the HFS is a tree. Focusing on systems with a single output (MISO) and tree shape, the input number of the modules must meet the following equation:

$$\sum_{i=1}^{L} ci = n + L - 1 \;, \qquad (2)$$

where *n* is the number of input variables of the HFS. If all the modules have the same number of inputs, *c*, the number of modules of the HFS is.

$$L = \frac{n-1}{c-1} \;, \qquad (3)$$

and, then, the number of rules of the HFS is

$$Number \quad of \quad rules = \left(\frac{n-1}{c-1}\right) \cdot m^{c} . \qquad (4)$$

This represents a linear grow in the number of rules in terms of the number of input variables of the system instead of the exponential grow that suffer the standard fuzzy systems. The minimum number of rules is reached for *c* = 2, i.e., when using modules with two input variables. These modules are usually known as *fuzzy logic units* (FLUs). The use of FLUs has another important feature. The behavior of these modules can be represented by a surface, so the interpretation of its behavior can be made both from a linguistic point of view and from a graphical point of view.

An important aspect of the HFS is their capability to approximate functions. It is well known that standard fuzzy systems are universal approximators [7]. Although there are some proposals of HFS that have been demonstrated to be universal approximators [8][9], these proposals refers to high order Takagi-Sugeno fuzzy systems (with an order similar to the number of inputs). So the problem with the dimensionality is not solved, but shifted from the antecedent of the rules on a standard fuzzy system to the consequent of the rules on the Takagi-Sugeno hierarchical fuzzy system [10]. On the other hand, Mamdani type hierarchical fuzzy systems are not universal approximators. In this case, the interest is now to know which functions can be approximated by a Mamdani type HFS. In [11], the concept of *continuous functions with natural hierarchical structure* is defined and it is proved that a Mamdani type HFS can approximate these functions correctly when the HFS has the same hierarchical structure as that of the function.

In order to find the structure of HFS which can approximate some given continuous function, some authors have proposed the use of genetic algorithms [12] [13]. This paper presents a tool that develops an exhaustive search among all the modular decomposition that can be done based on modules with two inputs (FLUs). The hierarchical structure generated by the tool is the one that minimize the mean square error after an optimization process.

II. The Modular Decomposition Algorithm

The proposed methodology consists in making an exhaustive search among all possible hierarchical structures with *n* input variables, based on FLUs. This section describes the algorithm used to generate all these structures. It is a recursive algorithm that generates all the structures of *n* input variables in terms of all the structures of (*n*-1) input variables.

The base case of the recursive algorithm is the structure of a system with two input variables, which is formed by a single FLU, as shown in Fig. 1.

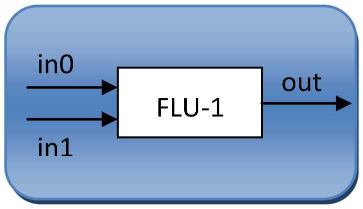


Fig 1. Base case for the recursive algorithm

The general case consists in generating the structures of *n* input variables using the structures of (*n*-1) input variables. Given a hierarchical structure with (*n*-1) inputs, it's possible to generate a set of structures with *n* inputs inserting a new FLU in each variable of the structure. Fig. 2 shows an example of the structures with 3 inputs, which can be generated through the base case. The structure of Fig. 2.a is obtained by including the new FLU in the position of the variable *in0*. Fig. 2.b is obtained including the new FLU in the position of the variable *in1*. The structure shown in Fig. 2.c is obtained including the new FLU in the position of the output variable.

Every hierarchical structure with *n* inputs contains (*n*-1) FLUs, so the total number of variables of the structure is (2n-1). Therefore, the number of structures with (*n*+1) inputs is

$$S(n+1) = (2n-1)\cdot S(n) = \prod_{i=1}^{n}(2i-1) \qquad (5)$$

Table 1 shows the values of the number of different hierarchical structures that can be generated using FLUs. It

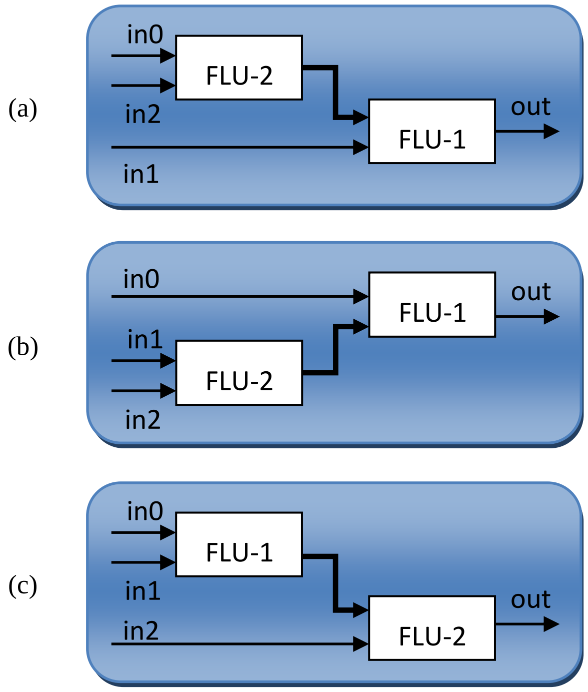


Fig. 2. Hierarchical structures for 3 input variables. The conection where the new module (FLU-2) is inserted is represented in bold.

can be seen that this number increase exponentially with respect to the number of inputs. For those problems with a number of input variables less than 7 or 8 the exhaustive search of the best structure is feasible. When the number of inputs is greater than 8, the number of structures is too high, leading to the use of random search techniques.

TABLE I
NUMBER OF HIERARCHICAL STRUCTURES

Input variables	Number of structures
2	1
3	3
4	15
5	105
6	945
7	10395
8	135135
9	2027025

III. PARAMETRIC OPTIMIZATION OF HIERARCHICAL FUZZY SYSTEMS

From all the hierarchical structures in which a system with *n* input variables can be decomposed, the proposed methodology selects the one that can adjust the best to a training dataset. In order to evaluate the approximation capability of each structure, the methodology uses a parametric optimization algorithm based on the Gradient-Descent method. The use of algorithms based on Gradient-Descent method for tuning hierarchical fuzzy systems is also proposed in [14]. The proposal in our methodology is the use of the RPROP algorithm [15] and Levenberg-Marquardt algorithm [16], which present a higher convergence speed. These algorithms (and many others) are included in the optimization tool of the Xfuzzy environment [17]. This feature has greatly eases the development of the Xfhl tool described in this work.

The hierarchical structures are evaluated by the mean square error (MSE) on the training dataset. This function is

$$MSE=\frac{1}{N}\cdot\sum_{i=1}^{N}\left(\tilde{y}_i-y(\tilde{x}_i)\right)^2 \quad , \qquad (6)$$

where N is the number of instances in the dataset, $(x_i, \tilde{y}_i)$ are the input-output instances of the data set and $y(x)$ represents the behavior of the system.

The behavior of the system depends on the hierarchical structure and on the behavior of the different FLUs. The behavior of each FLU depends on the parameters that define the membership functions of its input variables and the parameters using in the calculation of its output. The goal of the optimization algorithm is to tune the values of all these parameters in order to minimize the value of the MSE.

The optimization algorithms proposed in this methodology are based on the gradient of the MSE, that is, on the derivative of the MSE with respect to the different parameters in the system. These derivatives are computed using the chain rule. For example, the derivative of the MSE with respect to the parameters of the FLU-1 module in Fig. 3 is given by the expression

$$\frac{\partial MSE}{\partial p}=\sum_{i=1}^{N}\left(\frac{\partial MSE}{\partial y(\tilde{x}_i)}\right)\cdot\left(\frac{\partial y(\tilde{x}_i)}{\partial a_1(\tilde{x}_i)}\right)\cdot \left(\frac{\partial a_1(\tilde{x}_i)}{\partial a_0(\tilde{x}_i)}\right)\cdot\left(\frac{\partial a_0(\tilde{x}_i)}{\partial p}\right) \qquad (7)$$

In the expression above, the derivative of the MSE with respect of the output of the system for the i-th instance is

$$\frac{\partial MSE}{\partial y(\tilde{x}_i)}=\frac{2}{N}\cdot\left(y(\tilde{x}_i)-\tilde{y}\right) \qquad (8)$$

In order to compute the rest of the partial derivatives, it is necessary to know the behavior of the FLUs. If, into these modules, the conjunction operator is assigned to the product, and the fuzzy mean is used as defuzzification method, then the behavior of the FLU is the following:

$$y(x_1,x_2)=\frac{\sum_{i=1}^{m}\sum_{j=1}^{m}\mu_{1i}(x_1)\cdot\mu_{2j}(x_2)\cdot c_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m}\mu_{1i}(x_1)\cdot\mu_{2j}(x_2)} \qquad (9)$$

where $\mu_{1i}(x_1)$ are the membership functions related to the

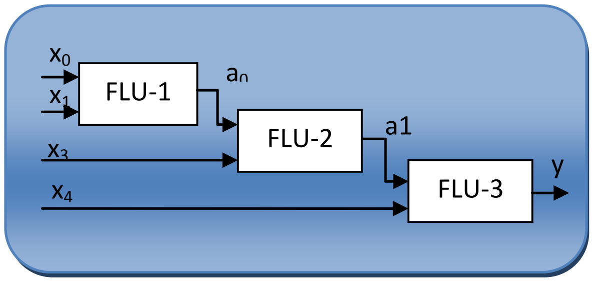


Fig. 3. Example of a hierarchical structure

input variable x_1, $\mu_{2j}(x_j)$ are the membership functions related to the input variable x_2, m is the number of membership functions of each variable, and c_{ij} is the centroid for the conclusion of the rule relating "x_1=*label-i*" and "x_2 = *label-j*".

Given the expression above, it is possible to compute the partial derivative of the output of a FLU with respect to the centroids of the rules:

$$\frac{\partial y(x_1,x_2)}{\partial c_{ij}}=\frac{\mu_{1i}(x_1)\cdot\mu_{2j}(x_2)}{\sum_{k=1}^{m}\sum_{l=1}^{m}\mu_{1k}(x_1)\cdot\mu_{2l}(x_2)} \quad . \qquad (10)$$

Equation (9) can be used also to compute the partial derivative of the output of a FLU with respect to the activation degree of each membership function:

$$\frac{\partial y(x_1,x_2)}{\partial \mu_{1i}(x_1)}=\frac{\sum_{j=1}^{m}\mu_{2j}(x_2)\cdot c_{ij}}{\sum_{k=1}^{m}\sum_{l=1}^{m}\mu_{1k}(x_1)\cdot\mu_{2l}(x_2)} -\frac{\left(\sum_{j=1}^{m}\mu_{2j}(x_2)\right)\cdot\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\mu_{1i}(x_1)\cdot\mu_{2j}(x_2)\cdot c_{ij}\right)}{\left(\sum_{k=1}^{m}\sum_{l=1}^{m}\mu_{1k}(x_1)\cdot\mu_{2l}(x_2)\right)^2} \qquad (11)$$

Finally, the partial derivative of the membership functions with respect to their parameters must be considered. For example, for a Gaussian membership function,

$$\mu(x,a,b)=\exp\left(-\left(\frac{x-a}{b}\right)^2\right) \quad , \qquad (12)$$

the derivative with respect to the variable x is

$$\frac{\partial\mu(x,a,b)}{\partial x}=-\frac{2}{b}\cdot\left(\frac{x-a}{b}\right)\cdot\exp\left(-\left(\frac{x-a}{b}\right)^2\right) \quad ; \qquad (13)$$

the derivative with respect to the parameter a is

$$\frac{\partial \mu(x,a,b)}{\partial a} = \frac{2}{b} \cdot \left(\frac{x-a}{b}\right) \cdot \exp\left(-\left(\frac{x-a}{b}\right)^2\right) \quad ; \quad (14)$$

and the derivative with respect to the parameter *b* is

$$\frac{\partial \mu(x,a,b)}{\partial b} = \frac{2}{b} \cdot \left(\frac{x-a}{b}\right)^2 \cdot \exp\left(-\left(\frac{x-a}{b}\right)^2\right) . \qquad (15)$$

Considering all these equations it is possible to compute the gradient of the MSE as shown in (7) and use it in the selected parametric optimization algorithm. In the case that the behavior of FLUs is based on a distinct conjunction operator, a distinct defuzzification method, or distinct membership functions, the equations (8) to (15) need to be recalculated.

IV. The Xfhl Tool

Xfhl is a new tool integrated into the Xfuzzy development environment for fuzzy system. The Xfhl tool implements the methodology for the induction of hierarchical fuzzy systems discussed in the preceding paragraphs. The tool works by generating all hierarchical structures of *n* inputs and one output that can be built with modules of two inputs and one output (FLUs). The selected structure is that having the minimum mean square error after a parametric optimization process based on a gradient descent algorithm (RProp or Levenberg-Marquardt).

The generation of the set of hierarchical structures for *n* input variables is based on the algorithm presented in the second section. The implementation of the algorithm uses an iterator based on a stack of structures. Each stack level corresponds to a number of inputs, starting from the base structure with two inputs. The top of the stack contains the structure of *n* inputs generated by the iterator. On each iteration, the algorithm pops the last structure and generates the following structure from the structure of (*n*-1) inputs stored on the stack. When all the structures of *n* inputs have been generated for the structure on the (*n*-1) level, the algorithm pops the structure of (*n*-1) inputs and generates a new structure of (*n*-1) inputs based on the structure of (*n*-2) entries stored on the stack, and so on. Thanks to this iterator, there is no need to store all hierarchical structures in memory at once.

The tool performs a parametric optimization process on all parameters of each structure. In order to do this, the tool computes the gradient of the mean square error as described in Section 3 and applies the chosen gradient descent algorithm.

Since the optimization and evaluation of a structure can be performed independently of the others, the search process is easily parallelizable. The Xfhl tool has been programmed to take advantage of parallelization capabilities of current systems (multicore architectures, multiprocessor and hyperthreading technology) so that the tuning process of the different hierarchical structures is performed in parallel in multiple execution threads.

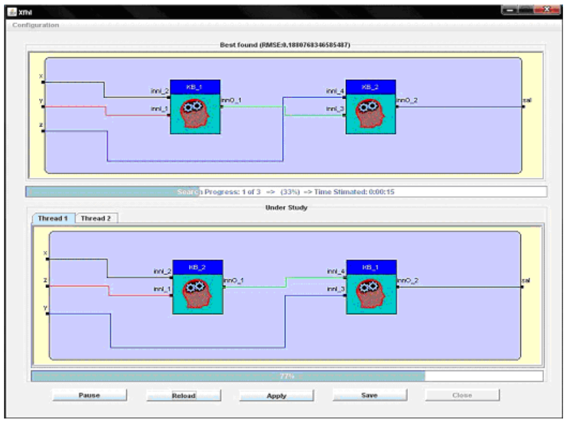
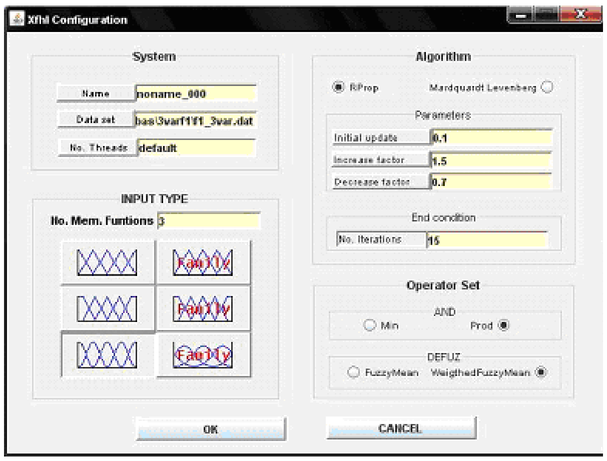


Fig. 4. The main window of Xfhl

Fig. 4 shows the main window of Xfhl. The configuration menu is located at the top of the window. This menu allows the selection of the different run options of the tool. Under the menu bar, there is an area with the best fuzzy system found to date and a progress bar indicating the percentage of structures studied so far. Under this bar is a panel which shows the status of each thread of the tool. For each thread, the panel includes the structure representation under study and a progress bar of the optimization process. In the bottom of the window there are controls buttons dedicated to run the process, stop, save the best result obtained so far and close the tool.

Fig. 5 shows the configuration window of the tool. This configuration requires the selection of the following fields:

- Training dataset.
- Number of concurrent threads to execute. The value "Default" specifies that the number of threads is equal to the number of cores in the system.
- Number of membership functions for each input variable.
- Style of the membership functions for the input


Xfhl Configuration

System
Name: noname_000
Data set: bas\3varf1f1_3var.dat
No. Threads: default

INPUT TYPE
No. Mem. Funtions: 3

Algorithm
[✓ RProp] Marquardt Levenberg
Parameters
Initial update: 0.1
Increase factor: 1.5
Decrease factor: 0.7
End condition
No. Iterations: 15

Operator Set
AND: Min [✓ Prod]
DEFUZ: FuzzyMean [✓ WeigthedFuzzyMean]

Fig. 5. The configuration window of Xfhl

variables. This style may be one of the following types: Free Triangles, Free Shouldered Triangles, Free Gaussians, Triangles Family, Shouldered-Triangular Family, and B-Splines Family.

- Function associated to the conjunction operator. It can be the minimum or the product.
- Defuzzification method used in the modules. The options are the fuzzy mean method and the weighted fuzzy mean method.
- The parametric optimization algorithm used to evaluate each structure. The options are RProp and Levenberg-Marquardt. The control parameters of the algorithm must be configured and the number of iterations in the tuning process must be introduced.

V. Test

In order to demonstrate how the Xfhl tool works, the following continuous function with 4 input variables and separable hierarchical structure has been considered:

$$f(w,x,y,z)=\left(w^2+z^2\right)\cdot\cos\left(\frac{\pi}{3}\cdot(x-2y)\right) \qquad (16)$$

The training dataset has been generated considering that the 4 variables are defined in the interval [-1.0, 1.0]. All possible instances have been generated, sweeping the values of the variables in steps of 0.25, producing a dataset with a total of 6561 instances.

The process have been configured considering a division of variables into three Gaussian membership functions and choosing the product as conjunction operator and weighted fuzzy mean as defuzzification method. The parametric optimization has been carried out through 40 iterations of the Rprop algorithm. The execution was carried out with two threads on Intel Core2 Duo 2.0 GHz machine. The process spends approximately 3 minutes to finish. The best structure found by the tool (from the set of 15 possible decompositions) is shown in Fig. 6.a. This structure corresponds to the natural hierarchical structure of the target function. The structure has initially a RMSE = 2.9%. A later parametric optimization with RProp get a better approximation to a RMSE = 1.4%.

Fig. 6 shows the graphical representation of the behavior of the different modules of the hierarchical structure obtained. As shown in Figure 6b, the behavior of FLU-1 module approximates the cosine function on a linear relationship of the variables x and y. (Actually the function is displaced, since the output range is between 0 and 1). A more detailed study can even deduce this relationship. Regarding the behavior of the FLU-2 module, the Figure 6.c shows that the relation between the variables w and z correspond to a parabola, that is to say, to the function w^2+z^2 multiplied by a negative factor. In the case of FLU-3 module, the behavior is more difficult to interpret because this module is responsible not only for calculating the

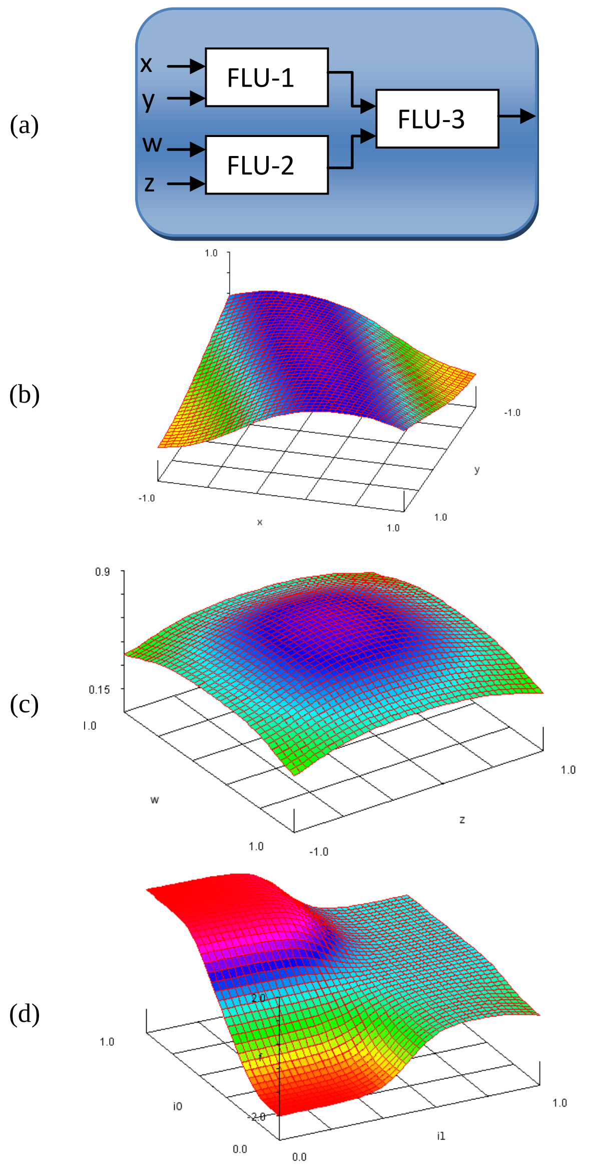


Fig. 6. Results of the Xfhl run for the target function. (a) best HFS found; (b) behavior of FLU-1 module; (c) behavior of FLU-2 module; (d) behavior of FLU-3 module.

product between the two parts of the function but also to compensate the offset and scale factors of the modules above.

VI. Conclusion

Hierarchical fuzzy systems are an interesting way to model a complex behavior while keeping a good interpretability. The better results on interpretability are obtained for HFSs composed by FLUs, that is, modules with two input variables. This work proposes a design methodology for HFSs based on an exhaustive search over the whole set of modular decompositions based on FLUs. The goodness of each HFS is evaluated after a tuning process based on the RPROP algorithm or the Levenberg-

Marquardt algorithm. This methodology has been implemented in a design tool called Xfhl, which has been included into the Xfuzzy development environment.

The methodology has demonstrated to be appropriate when the number of input variables of the system is less than 7 or 8. If the number of input variables is higher, the size of the search space is too large and the exhaustive search is non-viable. In these cases, the methodology should be adapted to the use of metaheuristics. This should lead to a hybrid algorithm combining an evolutionary algorithm for searching the HFS structure and a gradient-based algorithm for evaluating each structure.

References

[1] R. Bellman, *Adaptive Control Processes*, Princeton University Press, 1966.

[2] V. Torra, "A Review of the Construction of hierarchical Fuzzy Systems," *International Journal of Intelligent Systems*, vol. 17, 2002, pp. 531-543.

[3] S. Aja-Fernández and C. Arbeloa-López, "Matrix modeling of hierarchical fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 16, n. 3, 2008, pp. 585-599.

[4] M.L. Lee, H.Y. Chung, and F.M. Yu, "Modeling of hierarchical fuzzy systems," *Fuzzy Sets and Systems*, vol. 138, 2003, pp. 343-361.

[5] O. Cordón, F. Herrera, and I. Zwir, "A hierarchical knowledge-based environment for linguistic modeling: models and iterative methodology," *Fuzzy Sets and Systems*, vol. 138, 2003, pp. 307-341.

[6] G.V.S. Raju, J. Zhou, and R.A. Kisner, "Hierarchical fuzzy control," *International Journal of Control*, vol. 54, 1991, pp. 1201-1216.

[7] J.L. Castro, "Fuzzy logic controllers are universal approximators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, n. 4, 1995, pp. 629-635.

[8] M.G. Joo and J.S. Lee, "Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule," *Fuzzy Sets and Systems*, vol. 130, n. 2, 2002, pp. 175-188.

[9] L.X. Wang, "Universal approximation by hierarchical fuzzy systems," *Fuzzy Sets and Systems*, vol. 93, 1998, pp. 223-230.

[10] L.X. Wang, "Analysis and Design of Hierarchical Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, n. 5, 1999.

[11] X.J. Zeng and J.A. Keane, "Approximation Capabilities of Hierarchical Fuzzy Systems," *IEEE Transactions on Fuzzy Systems*, vol. 13, n. 5, 2005, pp. 659-672.

[12] A.D. Benítez and J. Casillas, "Aprendizaje Evolutivo de Sistemas Difusos Jerárquicos en Serie," in *Actas del VI Congreso Español sobre Metaheurística, Algoritmos Evolutivos y Bioinspirados (MAEB'09)*, 2009, pp. 255-262. (in spanish)

[13] Y. Chen, B. Yang, A. Abraham, and L. Peng, "Automatic Design of Hierarchical Takagi-Sugeno Type Fuzzy Systems Using Evolutionary Algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 15; n. 3, 2007, pp. 385-397.

[14] D. Wang, X.J. Zeng, and J. Keane, "Learning for Hierarchical Fuzzy Systems Based on the Gradient-Descent Method," in *Proc. 2006 IEEE International Conference on Fuzzy Systems*, 2006, pp. 92-99.

[15] M. Riedmiller and H. Braun, "RPROP: A fast and robust backpropagation learning strategy," in *Proc. 4th Australian Conference on Neural Networks*, 1993, pp. 169-72.

[16] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, Ltd., 1986.

[17] F.J. Moreno-Velo, I. Baturone, A. Barriga, and S. Sanchez-Solano, "Automatic tuning of complex fuzzy systems with Xfuzzy," *Fuzzy Sets and Systems*, vol. 158, 2007, pp. 2026-2038.