A Multi-Agent Architecture for the Design of Hierarchical Interval Type-2 Beta Fuzzy System

Yosra Jarraya, Souhir Bouaziz, Hani Hagras, *Fellow, IEEE* and Adel M. Alimi, *Member, IEEE*

Abstract—**This paper presents a new methodology for building and evolving hierarchical fuzzy systems. For the system design, a tree-based encoding method is adopted to hierarchically link low dimensional fuzzy systems. Such tree structural representation has by nature a flexible design offering more adjustable and modifiable structures. The proposed hierarchical structure employs a type-2 beta fuzzy system to cope with the faced uncertainties, and the resulting system is called the Hierarchical Interval Type-2 Beta Fuzzy System (HT2BFS). For the system optimization, two main tasks of structure learning and parameter tuning are applied. The structure learning phase aims to evolve and learn the structures of a population of HT2BFS in a multi-objective context taking into account the optimization of both the accuracy and the interpretability metrics. The parameter tuning phase is applied to refine and adjust the parameters of the system. To accomplish these two tasks in the most optimal and faster way, we further employ a multi-agent architecture to provide both a distributed and a cooperative management of the optimization tasks. Agents are divided into two different types based on their functions: a structure agent and a parameter agent. The main function of the structure agent is to perform a multi-objective evolutionary structure learning step by means of the Multi-Objective Immune Programming algorithm (MOIP). The parameter agents have the function of managing different hierarchical structures simultaneously to refine their parameters by means of the Hybrid Harmony Search algorithm (HHS). In this architecture, agents use cooperation and communication concepts to create high-performance HT2BFSs. The performance of the proposed system is evaluated by several comparisons with various state of art approaches on noise-free and noisy time series prediction data sets and regression problems. The results clearly demonstrate a great improvement in the accuracy rate, the convergence speed and the number of used rules as compared with other existing approaches.**

Index Terms—**Beta function, hierarchical representation, interval type-2 fuzzy system, multi-agent architecture, multi-objective structure learning.**

I. Introduction

THe recent years have witnessed a growing interest in type-2 fuzzy logic systems due to their ability to handle high levels of uncertainties faced in dynamic real world and changing environments [1]. In fact, these uncertainties are present in most applications and can be a result of different sources such as the presence of noise in the training data, linguistic uncertainties, the uncertainty in input and output data as they usually contain inaccurate, incomplete, weak, and sometimes false information [2]. Type-2 fuzzy logic systems have been employed in various applications including pattern recognition [3], intelligent control [4], mobile robots [5], time series prediction [6], [7], function approximation [8], [9], classification [10], [11]. This work presents a new interval type-2 fuzzy system based on the Beta basis function [12], [13] for system modeling. The proposed system is termed interval type-2 Beta fuzzy system (IT2BFS).

In fuzzy logic systems, when the dimensionality and the complexity of the given applications increase, the number of used fuzzy rules will increase exponentially (the curse of dimensionality problem [14]) which can reduce the interpretability of the obtained rule base. As an alternative to solve this problem, hierarchical fuzzy design was suggested in the early 1990s by Raju and Zhou [15] to reduce the number of fuzzy rules from an exponential function of system variables to a linear one. In this case, instead of the use of a standard high-dimensional flat fuzzy system, a number of lower-dimensional sub-fuzzy models are linked in a hierarchical way. This method of hierarchical modeling allows the construction of fuzzy systems which are more interpretable (with fewer rules) as well as being relatively accurate with good approximation abilities. For example, suppose that the standard fuzzy system illustrated in Fig.1a has 4 input variables each represented by 5 fuzzy sets, then the total number of rules is equal to $5^4 = 625$ rules. However, in the case of the hierarchical fuzzy system of Fig.1b, each subfuzzy system (SFS) consists of 5^2 rules and, consequently, the total rules number is equal to $3 * 5^2 = 75$ rules. This shows the great rule reduction achieved by the hierarchical structure which makes it a good candidate to solve high-dimensional problems.

Recently, hierarchical fuzzy design has attracted increasing attentions and many works have been proposed to build or to optimize these systems [16–22]. However, most of the existing hierarchical systems employed type-1 fuzzy models. To the author’s knowledge, very few publications can be found in the literature that address the use of fuzzy type-2 hierarchical design [5], [23], [24]. In this paper, we will develop a novel hierarchical IT2BFS based on a tree structural representation called the hierarchical interval type-2 Beta fuzzy system (HT2BFS). Hence, instead of using a standard IT2BFS with high dimension, the input variables are distributed over different sub-interval type-2 fuzzy models having lower dimensions.

Y. Jarraya, S. Bouaziz and A. M. Alimi are with the Research Groups in Intelligent Machines (REGIM-Lab), University of Sfax, National Engineering School of Sfax (ENIS), BP 1173, Sfax, 3038, Tunisia (e-mail: yosra.jarraya@ieee.org; souhir.bouaziz@ieee.org; adel.alimi@ieee.org).

H. Hagras is with the Computational Intelligence Centre, School of Computer Science and Electronic Engineering, University of Essex,Colchester, CO4 3SQ, U.K.(e-mail: hani@essex.ac.uk).

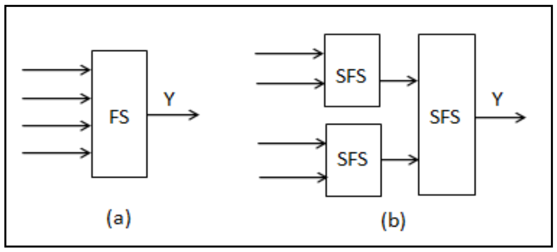


Fig. 1. An example of (a) usual flat standard fuzzy system, (b) hierarchical fuzzy system

Traditionally, most of the existing learning methods for type-2 fuzzy systems use single-objective learning techniques such as the gradient descent algorithms [25-27], least-squares methods [28], Evolutionary Algorithms (EAs) [29, 30], and other hybridizations [8], [31]. In fact, most of the existing approaches reported in the literature focused on improving only the accuracy of type-2 models while the interpretability was neglected. In order to optimize the model comprehensibility as well as the accuracy, Multi-Objective Evolutionary Algorithms (MOEAs) are employed in our research. Multi-Objective Evolutionary Algorithms (MOEAs) have widely spread over the past few years as an most effective tool to optimize type1 fuzzy systems [32–34], but until now, little works have exploited these algorithms to optimize type-2 fuzzy systems [9], [11], [35]. In this work, we will employ the Multi-Objective Immune Programming (MOIP) in order to evolve the HT2BFS structures.

This paper focuses on two main tasks of HT2BFS optimization which are a multi-objective structure learning process and a parameter tuning process. To accomplish these tasks in the most optimal and faster way, we further need a multi-agent architecture to provide both a distributed and a cooperative management of these optimization tasks. In fact, a multi-agent system is a coherent and interactive system formed by a set of agents with varied functions, which can share information and cooperate with each other to complete common goals [36]. An agent can be an abstract or a physical entity that has the aspect of initiative, cooperation and autonomy. In this study, a multi-agent architecture is proposed to provide a distributed coordinated environment of optimization. Based on their functions, agents are classified into two categories: a structure agent and a parameter agent. Indeed, the structure agent executes the proposed MOIP algorithm as a multi-objective structure optimization phase. The function of this agent is to learn the structures of a population of HT2BFSs with the objective of attending a good interpretability-accuracy trade-off. Once a set of optimal structures is obtained, a number of parameter agents are launched for further parallel tuning of the parameters encoded on these optimized structures. Each parameter agent will execute its own Hybrid Harmony Search (HHS) algorithm [37] for parameters adjustment. The tuned parameters are the interval type-2 Beta membership function parameters and the consequent parts of fuzzy rules. And then, we go back to improve the structures again by the structure agent. The loop continues until a stopping criterion is reached, and as a final result, an optimal HT2BFS is obtained. This new approach shows its efficiency in terms of high learning capacities, good convergence speed and a smaller rule base.

The rest of this paper is structured as follows: section II defines the proposed interval Type-2 Beta Fuzzy System. The MOIP and the HHS algorithms used in the training process are respectively presented in sections III and IV. The evolutionary HT2BFS is detailed in section V. Next, the employed multi-agent architecture for structure and parameter optimization processes is described in section VI. Simulation results with a comparative study are presented in section VII. And finally, the conclusion is drawn in section VIII.

II. The Interval Type-2 Beta fuzzy system

A. *Interval Type-2 Beta Membership Function*

A type-2 fuzzy set (T2FS) $\tilde{A}$ is characterized by a type-2 membership function $\mu_{\tilde{A}}(x,u)$ which is expressed by [38]:

$$\tilde{A} = \{(x,u), \mu_{\tilde{A}}(x,u) | x \in X \,, u \in [0,1]\} \tag{1}$$

where $\mu_{\tilde{A}}(x,u)$ is a type-1 fuzzy set called the secondary set with $0 \le \mu_{\tilde{A}}(x,u) \le 1$. J_x is the primary membership of $\tilde{A}$ denoted by [38]:

$$J_x = \{(x,u) | u \in [0,1], \mu_{\tilde{A}}(x,u) > 0\} \tag{2}$$

When all the secondary grades $\mu_{\tilde{A}}(x,u)$ equal 1, then the T2FS ($\tilde{A}$) is named an interval type-2 fuzzy set (IT2FS) [39]. The uncertainty in the primary MF is expressed by a bounded region named the footprint of uncertainty (FOU). The FOU provides additional degrees of freedom and it is delimited by two MFs called the Upper Membership Function (UMF), $\bar{\mu}_{\tilde{A}}(x)$, and the Lower Membership Function (LMF), $\underline{\mu}_{\tilde{A}}(x)$.

The choice of the shape of MFs is important since it has an impact on the performance of the fuzzy system. Different shapes of MFs are usually used in the fuzzy logic literature like triangular, gaussian, trapezoidal, etc. However, since the piece-wise linear MFs (like triangular and trapezoidal MFs) are formed from straight line segments, they are not smooth at the corner points specified by the parameters. Symmetric bell-shaped (such as gaussian) membership functions are also widely used since they present more smoothness, but they are unable to define asymmetric MFs. On the other hand, the Beta MF proposed by Alimi [12], [13] can generate richer forms than those functions. The Beta function has universal approximation proprieties and is able to approximate other usual functions such as triangular, gaussian or trapezoidal functions [40]. For example, [40] demonstrated the capacity of Beta function to approximate the Gaussian function and noted that the reverse is not true. In addition, the Beta function is characterized by its high flexibility and its ability to generate rich shapes (asymmetry, linearity, etc.). The Beta membership function is defined by:

$$\beta(x;c,\sigma,p,q) = \tag{3}$$

$$\begin{cases} \left[1+\frac{(p+q)(x-c)}{\sigma p}\right]^{p}\left[1-\frac{(p+q)(c-x)}{\sigma q}\right]^{q} & if\ x \in \left]c-\frac{\sigma p}{p+q},\ c+\frac{\sigma p}{p+q}\right[\\ 0 & elsewhere \end{cases}$$

where c is the center of the function and σ is its width. p and q are the form parameters, $p, q > 0$.

In this study, a Beta primary MF having an interval-valued secondary MF is employed and called the interval type-2 Beta membership function (IT2BMF). This function is characterized by a fixed center c, an uncertain width σ and uncertain form parameters p and q:

$$\begin{cases}\beta(x;c,\sigma,p,q) = \left[1+\frac{(p+q)(x-c)}{\sigma p}\right]^p \left[1-\frac{(p+q)(c-x)}{\sigma q}\right]^q \\ \quad \sigma \in [\sigma_L,\sigma_U],\ p \in [p_L,p_U]\ and\ q\ \in [q_L,q_U]\end{cases} \quad (4)$$

where $\sigma_L,\ \sigma_U,\ p_L,\ p_U,\ q_L\ and\ q_U$ are positive real values with $\sigma_L < \sigma_U,\ p_L < p_U\ and\ q_L < q_U$. The upper and the lower Beta MFs are respectively denoted by:

$$\begin{cases}\bar{\mu}_{\tilde{A}}(x) = \beta(x;c,\sigma_U,p_U,q_U) \\ \underline{\mu}_{\tilde{A}}(x) = \beta(x;c,\sigma_L,p_L,q_L)\end{cases} \quad (5)$$

The use of IT2BMFs provides flexibility and capacity to create more variant MF shapes. In comparison with the gaussian function, the Beta function relies on two additional form parameters (p and q) which allow a greater flexibility in the modeling of type-2 fuzzy sets. Hence, different shapes of FOUs can be created using the IT2BMF. Fig. 2 presents some examples of interval type-2 Beta MFs with uncertain σ, p and q having different FOU.

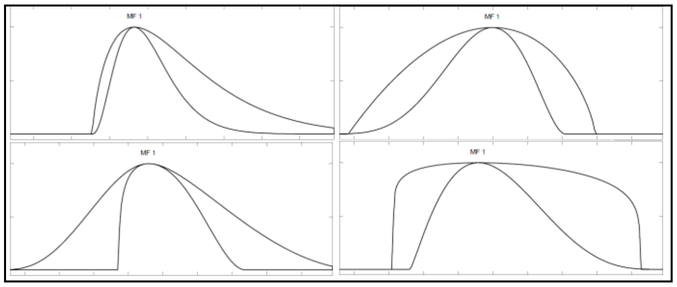


Fig. 2. Examples of Interval Type-2 Beta MFs with different FOU

B. Interval Type-2 Beta Fuzzy System

In this paper, the Interval A2-C1 TSK fuzzy model [41] is adopted and using IT2BMFs, the system is termed the interval type-2 Beta fuzzy system (IT2BFS). In the IT2BFS, the antecedent parts of each fuzzy rule are interval type-2 Beta fuzzy sets, while the consequent parts are of TSK nature having interval weights. Consider an IT2BFS with n inputs $x_i (i = 1,..,n)$, one output and M fuzzy rules, the j^{th} rule can be written as follows:

$$If\ (x_1\ is\ \tilde{A}_{1j})\ and\ \dots and\ (x_n\ is\ \tilde{A}_{nj})\ then\ Y_j = C_{0j} + C_{1j}x_1 + \dots + C_{nj}x_n \quad (6)$$

where $j = 1,\dots,M$; $\tilde{A}_{ij}$ are the antecedent fuzzy sets modeled by the IT2BMFs; C_{ij} are the consequent sets formed by interval type-1 fuzzy sets; Y_j is the j^{th} rule output. The j^{th} rule firing strengths are evaluated using the product t-norm operator:

$$\begin{cases} F_j(x) = \left[\underline{f_j}(x)\ ,\overline{f_j}(x)\right] \\ \underline{f_j}(x) = \prod_{i=1}^{n} \underline{\mu}\,(x_i),\ \overline{f_j}(x) = \prod_{i=1}^{n} \overline{\mu}\,(x_i)\end{cases} \quad (7)$$

where $\underline{f_j}(x)$ and $\overline{f_j}(x)$ are respectively the lower and the upper firing strengths.

The type-reduced set is an interval fuzzy set defined by its two end points, its left end point (y^l) and its right end point (y^r):

$$y = [y^l, y^r] = \int_{y_1} \dots \int_{y_M} \int_{f_1} \dots \int_{f_M} 1 \ / \ \left(\frac{\sum_{j=1}^{M} f_j\, y_j}{\sum_{j=1}^{M} f_j}\right) \quad (8)$$

where $y_j \in Y_j$, $y_j = [y_j^l, y_j^r]$ and $f_j \in F_j$. The type-reduced end points y^l and $y^r$are calculated through the KM algorithm using center of sets type-reduction [20]. Finally, the final output is defuzzified and calculated as follows:

$$y = (y^l + y^r)/2 \quad (9)$$

III. Multi-Objective Immune Programming algorithm: MOIP

A. Dominance and Pareto-Optimality

A minimization multi-objective problem has the following form:

$$Minf(x) = [f_1(x),..,f_k(x)] \quad (10)$$

subject to:

$$g_j(x) \le 0\ , j = 1, \dots, p \quad (11)$$

$$h_j(x) \le 0\ , j = 1, \dots, q \quad (12)$$

where k defines the number of objective functions $f_j: \mathbb{R}^n \to \mathbb{R}$. $x = [x_1, \dots, x_n]^T$ is the vector of decision variables. $g_j(x)$ and $h_j(x)$ are the functions representing the constraints of the problem. p and q are respectively the number of equality and inequality constraints. Unlike single objective optimization, multi-objective optimization considers that there is no unique optimum solution considering all objectives, but rather there are several solutions that provide different compromises between the objectives known as non-dominated or Pareto optimal solutions. Those solutions are generated using the Pareto dominance concept [42]. The main idea of dominance concept is that a given solution x can dominate another solution y if and only if:

- x is not worse than y in any of the objectives;
- x is strictly better than y in at least one of the objectives;

Solution x is named Pareto optimal if there is no solution in the search space that dominates it. In the objective space, the set of Pareto optimal solutions is called the Pareto optimal front or Pareto front.

B. Basic Single-Objective Algorithm: IP

The Immune Programming (IP) [43] is a population-based algorithm inspired from the clonal selection principle. It operates with a population of antibodies modeled by tree structures and uses the following three principal operators to evolve new generations:

- **Cloning operator:** allows the multiplication of the best candidates in the population. It presents more chance to explore a favorable region in the solution space.
- **Mutation operator:** applied to modify an antibody (tree) according to its fitness value. In this work, four mutation operators were employed which are: pruning (replace a randomly selected sub-tree by a random leaf node); growing (replace a randomly selected leaf node by a random sub-tree); modifying all leaf nodes randomly; modifying one leaf node randomly.
- **Replacement operator:** allows the replacement of an antibody of the population with another one generated at random. This operator is one of the most responsible factors of the population diversity.

C. Multi-Objective Algorithm: MOIP

The IP algorithm proved its efficiency in different studies, but it is still often used as a single optimization algorithm. In this work, we propose an extended multi-objective version of the IP algorithm called the Multi-Objective Immune Programming algorithm (MOIP). This algorithm is able to improve the structures of a given population of antibodies with the consideration of more than one objective function. To achieve such multi-objective optimization goals, the MOIP methodology combines the Pareto-dominance principles with IP operators and uses an elitist strategy in its evolution. This strategy makes use of an external elitist archive A (secondary population) in order to store the best non-dominated antibodies (solutions) found so far over the generations.

The main steps of the algorithm consists of initialization of population, evaluation, Pareto-dominance selection, applying IP operators, and reiterating the search on population until a near optimal Pareto front is obtained.

In addition, in the case of single objective optimization, a child is usually selected over its parent if it has better fitness value. In MOIP, the superiority is measured as a dominance relationship, and a child is selected over its parent only if this latter dominates its parent. As a result, as the search progresses, the different solutions move more closer to the true Pareto front.

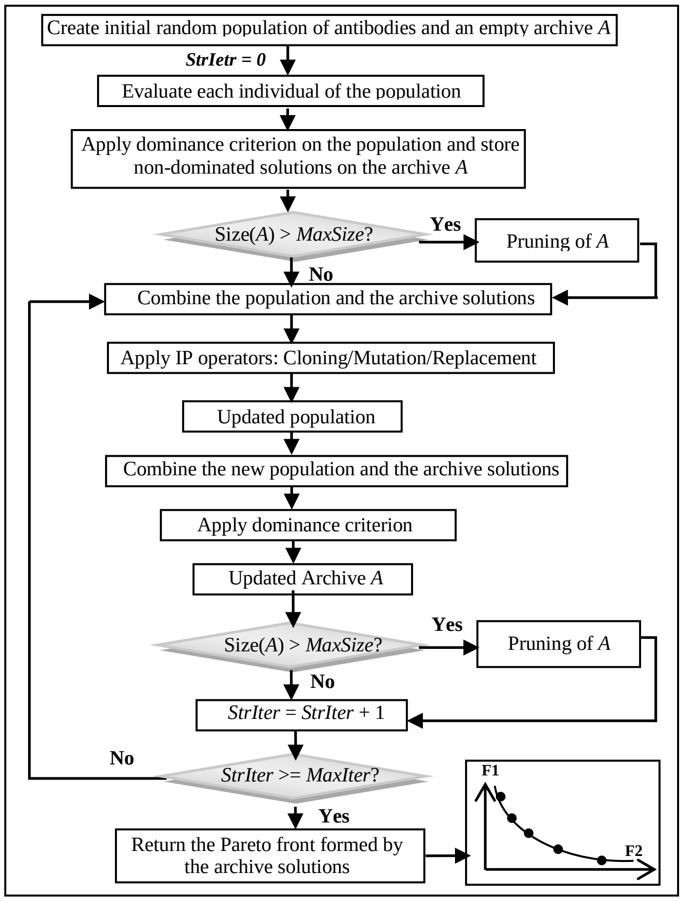
On the other hand, among the desirable characteristics of the obtained Pareto front is to have evenly spaced solutions covering the largest possible area of the front. Hence, we further use the crowding distance measure (applied in Non-dominated Sorting Genetic Algorithm II: NSGA-II [44]) to improve the diversity of solutions and to maintain a well-distributed front. In fact, the crowding distance gives a density estimation of solutions that surround one selected solution. A large average crowding distance allows a better diversity in the front. Suppose that $cd(x_i)$ represents the crowding distance of x_i (solution of the front). $cd(x_i)$ is evaluated by the following steps:

i) Initialization: $cd(x_i) = 0$;

ii) For each objective function f_j do:

- Sort the front's solutions along f_j ;
- $cd(x_i)= cd(x_i) + f_j$ (the solution preceding x_i in the ordered sequence) - f_j(the solution following x_i in the ordered sequence);

The flow chart of the MOIP algorithm is presented by Fig. 3. When applying the dominance criterion on a population, the antibodies are evaluated and checked for dominance relations among the population. Using the definition of Pareto dominance, an antibody is called a non-dominated antibody when it is not dominated by any other antibodies in the population. Then, the non-dominated solutions found are stored in the elitist archive A. The size of this archive is restricted to a predefined number. This restriction is imposed by a pruning process executed as follows: If the size of the archive (solutions number) is greater than *MaxSize*, then the crowding distances of all individuals of the archive are calculated and sorted in a descending order. The first *MaxSize* solutions are then selected to update the archive. Such pruning process aims to limit the archive size while preserving its diversity and spread along the front.



Create initial random population of antibodies and an empty archive A
StrIetr = 0
Evaluate each individual of the population
Apply dominance criterion on the population and store non-dominated solutions on the archive A
Size(A) > MaxSize?
Yes
Pruning of A
No
Combine the population and the archive solutions
Apply IP operators: Cloning/Mutation/Replacement
Updated population
Combine the new population and the archive solutions
Apply dominance criterion
Updated Archive A
Size(A) > MaxSize?
Yes
Pruning of A
No
StrIter = StrIter + 1
No
StrIter >= MaxIter?
Yes
Return the Pareto front formed by the archive solutions
F1
F2

Fig. 3. Flow chart of the MOIP algorithm

IV. The Hybrid Harmony Search Algorithm: HHS

The HHS algorithm [37] is an evolutionary music-inspired meta-heuristic algorithm inspired from the improvisation of music: a musician (decision variable) plays (creates) a note (value) to reach a good state of harmony (global optimum).

Inspite of its efficiency, the Harmony Search algorithm (HS) in its original version [45] contains some weaknesses. In fact, it is remarkable that, the harmony memory is usually stable and doesn't present changing values in the improvisation. Thus, in general, the standard HS algorithm has a small probability of providing new harmony vectors with good qualities. Therefore, there is a need to add a dynamic aspect allowing the creation of various values in memory with respect to their allowable ranges. This aspect is provided by the embedding of the Particle Swarm Optimization (PSO) algorithm which can generate after every iteration a new population totally different and nearer to the optimal solution. So, a hybridization between the HS and the PSO algorithm is proposed in [37] and called the Hybrid Harmony Search (HHS) algorithm. Indeed, the dynamic and stochastic aspects of particles velocities in PSO orientate the research to the right areas of the search space. In this case, the vectors of memory in HS are treated as particles taken from the swarm and the new values of memory for the new improvisation are supplied by the novel positions attained by the particles. For each particle j, the velocity v_j and the position x_j are calculated by the following equations:

$$v_j(t+1) = \Psi(t)v_j(t) + c_1\varphi_1\left(p_j(t) - x_j(t)\right) + c_2\varphi_2\left(p_g(t) - x_j(t)\right) \quad (13)$$

$$x_j(t+1) = x_j(t) + \left(1 - \Psi(t)\right) v_j(t+1) \quad (14)$$

where c_1 and c_2 are acceleration factors, j_1 and j_2 are random numbers in [0,1]. Ψ is the inertia factor. p_j is the local best position (attained by the j^{th} particle) and p_g is the global best position (attained by the swarm).

A simple global description of this algorithm is given by the following main steps:

– **Step1:** Formulation of the problem and initialization of the parameters which include:

• Harmony Memory Consideration Rate (*HMCR*): rate of the randomly selected values from the memory (0≤*HMCR*≤1);

• Harmony Memory Size (*HMS*): equivalent to population size;

• Pitch Adjustment Rate (*PAR*): rate of the altered values that was originally taken from the memory (0≤*PAR*≤1);

• Number of Improvisations (*NI*): the maximum number of generations;

• *FW* or *BW*: the width of the fret or bandwidth;

– **Step2:** Random initialization of the harmony memory (*HM*);

– **Step3:** Improvisation of a new harmony;

– **Step4:** Update of the harmony memory;

– **Step5:**

• Determines the best local and global positions;

• Calculates the particle velocity according to (13);

• Update of the particle position according to (14);

– **Step6:** Verification of the stopping criterion;

Readers may refer to [37] to get more details about this algorithm.

V. Evolution of the Hierarchical Interval Type-2 Beta Fuzzy System

A. *The Hierarchical Interval Type-2 Beta Fuzzy System: HT2BFS*

The hierarchical modeling of interval type-2 beta fuzzy systems is treated in this study. Thus, instead of designing a standard high dimensional IT2BFS, which is a common practice, the input variables are distributed over different sub-fuzzy models having lower dimensions. Consequently, each individual sub-fuzzy model having a moderate dimension will form a surface in all the hierarchy. For that, a tree-based encoding scheme is used to represent the hierarchical system. The reason for choosing the tree encoding method is that the tree has by nature a flexible hierarchical representation. Such encoding scheme can provide more adjustable and modifiable structures by means of existing or modified tree-based learning approaches, i.e., IP, Genetic Programming (GP), Ant Programming (AP), and so on. The proposed system is named the Hierarchical Interval Type-2 Beta Fuzzy System (HT2BFS). A possible tree structural representation (with 4 input variables and 4 hierarchical levels) and its corresponding HT2BFS are illustrated in Fig. 4a. The proposed HT2BFS is characterized by a set of non-leaf nodes *N* and leaf nodes *L*. Non-leaf nodes are formed by different sub-fuzzy models of IT2BFS type while leaf nodes are formed by original input variables. The node set *S* of the system is described as follows:

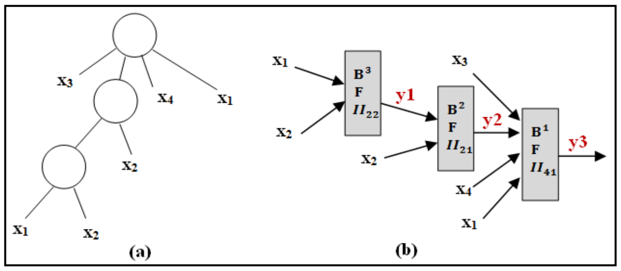


Fig. 4. a) A possible tree structural representation, (b) The corresponding HT2BFS: the tree node set *S*= { $BFII^1_{41}$, $BFII^2_{21}$, $BFII^3_{22}$, x_1, x_2, x_3, x_4}

$$S = N \cup L = \{BFII^l_{ci} / \, c \in \{2, \dots, NN\}, i \in \{1, \dots, T\}, l \in \{1, \dots, (ML-1)\}\} \cup \{x_1, \dots, x_M\} \quad (15)$$

where $BFII^l_{ci}$ represents a sub-fuzzy model of IT2BFS type formed by c inputs (children) and one evaluated output. *NN* defines the tree's maximal degree (nodes number), i is the index of the *BFII* having c children, T is the occurrence number of *BFII* having c offspring, l presents the level index of the tree and *ML* is the maximum number of levels (the tree's depth); $x_1, \dots, x_M$ are the original input variables illustrating the *L* leaf node set.

For the HT2BFS evaluation, each $BFII^l_{ci}$ receives c inputs and calculates one output. Some $BFII^l_{ci}$ calculate and generate their output to be exploited as inputs for other $BFII^l_{ci}$. The evaluation of the HT2BFS is done recursively from level to level (from left to right), and the root node generates finally the output of the whole tree-based system.

The rules at each non-leaf node were created as follows: Considering Fig. 4b as an exemple of a generated HT2BFS, the rules for each $BFII^l_{ci}$ are of TSK nature taking the following format:

$$BFII^3_{22}:\ R^{l=3}_i : If\ (x_1\ is\ \tilde{A}^3_{1i}\ )\ and\ (x_2\ is\ \tilde{A}^3_{2i})\ then\ Y^3_i = C^3_{0i} + C^3_{1i}x_1 + C^3_{2i}x_2 \quad (16)$$

$$BFII^2_{21}:\ R^{l=2}_j : If\ \left(y1\ is\ \tilde{A}^2_{1j}\right)\ and\ (x_2\ is\ \tilde{A}^2_{2j})\ then\ Y^2_j = C^2_{0j} + C^2_{1j}\, y1 + C^2_{2j}x_2 \quad (17)$$

$$BFII^1_{41}:\ R^{l=1}_k : If\ (x_1\ is\ \tilde{A}^1_{1k})\ and\ \left(y2\ is\ \tilde{A}^1_{2k}\right)\ and\ (x_3\ is\ \tilde{A}^1_{3k})\ and\ (x_4\ is\ \tilde{A}^1_{4k})\ then\ Y^1_k = C^1_{0k} + C^1_{1k}x_1 + C^1_{2k}\, y2 + C^1_{3k}x_3 + C^1_{4k}x_4 \quad (18)$$

where:

- $i = 1, \dots, M3$; *M3* presents the rules number of $BFII^3_{22}$; $\tilde{A}^3_{1i}$ and $\tilde{A}^3_{2i}$ are the antecedent fuzzy sets modeled by the IT2BMFs; $C^3_{0i}, C^3_{1i}\ and\ C^3_{2i}$ are the consequent sets formed by interval type-1 fuzzy sets; Y^3_i is the i^{th} rule output.

- $j = 1, \dots, M2$; *M2* presents the rules number of $BFII^2_{21}$; $\tilde{A}^2_{1j}$ and $\tilde{A}^2_{2j}$ are the antecedent fuzzy sets modeled by the IT2BMFs; $C^2_{0j}, C^2_{1j}\ and\ C^2_{2j}$ are the consequent sets formed by interval type-1 fuzzy sets; Y^2_j is the j^{th} rule output.

- $k = 1, \dots, M1$; *M1* presents the rules number of $BFII^1_{41}$; $\tilde{A}^1_{1k}, \tilde{A}^1_{2k}, \tilde{A}^1_{3k}$ and $\tilde{A}^1_{4k}$ are the antecedent fuzzy sets modeled by the IT2BMFs; $C^1_{0k}, C^1_{1k}, C^1_{2k}, C^1_{3k}$ and C^1_{4k} are the consequent sets formed by interval type-1 fuzzy sets; Y^1_k is the k^{th} rule output.

- x_1, x_2, x_3 and x_4 are original input variables; *y1*, *y2* and *y3* are respectively the outputs of $BFII^3_{22}$, $BFII^2_{21}$and $BFII^1_{41}$and are calculated by (9).

- *y3* is the output of the whole HT2BFS.

B. Initialization of HT2BFSs Population

In general, initial fuzzy rule generation is usually considered as a time-consuming and a difficult task since it needs expert knowledge information. One way of solving this difficulty is to use a clustering technique allowing an automatic extraction of an initial rule base. Clustering methods have been frequently used in the literature for the identification of both type-1 and type-2 fuzzy systems [46- 49]. In the same context, the subtractive clustering algorithm is applied in this study to derive the initial rules of each sub-fuzzy model from the available data and to determine the initial MFs locations. The use of such technique allows the optimization processes applied afterwards to converge in a shorter time.

The subtractive clustering algorithm is a fast and unsupervised algorithm used to divide the input data into smaller and meaningful subgroups named clusters, so that the items in the same cluster are as homogenous as possible. For this algorithm, the number of clusters is automatically defined based on a measure of data density in space. Hence, the obtained clusters centers will define the centers of MFs, and each center of a cluster will be transformed into a fuzzy rule. Based on this concept, the subtractive clustering algorithm is applied in the initialization step for the generation of an initial population of HT2BFSs.

To do this, first of all, a random population of initial trees having random structures is created; i.e. with random number of levels in [3, *LMax*] and with random number of nodes in [2, *NMax*], where *LMax* is the maximum level number and *NMax* is the maximum number of child nodes for each non-leaf node (degree of the tree). *NMax* and *LMax* are fixed according to the studied problem. In fact, the generation process of a tree structure is realized with a random and recursive way. It's an automated random process done recursively from top to bottom and from left to right. Non-leaf nodes are randomly distributed over the levels. Concerning the way of arrangement of original inputs in the different levels, original input variables are randomly chosen to be assigned for non-leaf nodes. The minimum size of each tree is equal to 5 and its maximum size is calculated as following:

$$SizeMax = \frac{NMax^{LMax}-1}{NMax-1} \tag{19}$$

After the initial generation of random population of trees, each tree is examined separately, and starting from the lowest level, the subtractive clustering algorithm is applied recursively by depth-first method. Consequently, for each non-leaf node, its child nodes are clustered in order to create the corresponding interval type-2 Beta sub-fuzzy model $BFII_{ci}^{l}$. The number of rules, the rule base in each $BFII_{ci}^{l}$ and the MFs locations are automatically defined by the clustering algorithm.

The embedding of such initialization clustering step allows both an automatic extraction of fuzzy rules from input data and also creates a better distribution of the Beta MFs centers. Consequently, the initial population will be composed of relatively good solutions of HT2BFSs, and this can save many generations of evolutionary search later. More details about this clustering method are presented in [46].

C. The Evolutionary HT2BFS: E_HT2BFS

The Evolutionary HT2BFS (E_HT2BFS) is a systematic design method of HT2BFSs. The HT2BFS evolution is considered as a search problem in both structure and parameter spaces. So, starting with a random population of trees having different structures, an initial population of HT2BFSs is derived by recursively clustering leaf nodes of input variables of each tree. Next, two main optimizations processes are iteratively applied: the HT2BFS structure learning and the HT2BFS parameter tuning. The structure learning phase is applied in a multi-objective context considering two objectives which are the accuracy maximization (by minimizing the error) and the interpretability maximization (by reducing the rules number).

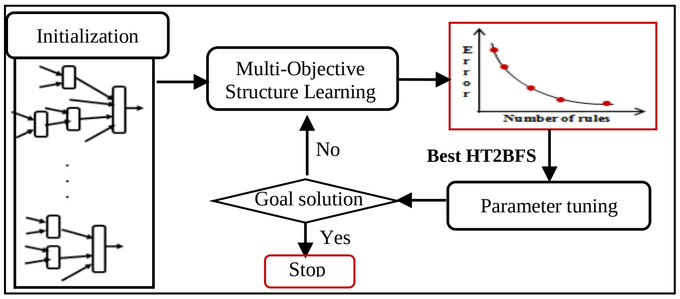


Fig. 5. The flowchart of the E_HT2BFS

The MOIP is used in the multi-objective structure learning phase, while the HHS is employed in the parameter tuning phase. The two algorithms are alternately applied until an optimal HT2BFS is obtained. The flowchart of the E_HT2BFS is illustrated in Fig. 5. As we can see from this figure, the multi-objective structure learning phase generates a Pareto-optimal front of non-dominated HT2BFSs, and then the most suitable solution (having a good trade-off between the two objectives) is selected to undergo the next parameter tuning phase. The stopping criterion here is to find a near-optimal HT2BFS or to reach the maximum number of global iterations. If the stopping criterion is not validated, another round of structure optimization is performed. In this case, the new population is formed by the best found HT2BFS having tuned parameters concatenated with a set of random generated individuals.

VI. MULTI-AGENT ARCHITECTURE FOR HT2BFS EVOLUTION: MA_HT2BFS

A multi-agent system (MAS) is considered as one of the most important branches in the distributed intelligent area. In general, a multi-agent architecture ensures a global organization between autonomous and coordinated agents in a distributed way. This architecture allows agents to interact together in order to accomplish common aims and to break the complexity of the given tasks [36]. In this study, our goal is to perform an optimization process for a population of HT2BFSs in the most efficient and fastest way. A key limitation of the E_HT2BFS (presented in section V.C) is its concentration on the parameter optimization of only one solution causing the loss of the other solutions of the pareto front. Indeed, all of the front's solutions have optimized structures and their exploitation can improve a lot the optimization process and can reduce the training time. Although the search for an optimal solution using the E_HT2BFS gives good results, it requires many iterations of learning since it usually relies on random populations in the structure learning rounds. To overcome this drawback, a multi-agent architecture is proposed in this study to efficiently parallelize the optimization task between different agents with the goal of adjusting and exploiting all of the front's solutions in order to contribute to the optimization process. Based on their functions, agents in this architecture are partitioned into two different types: a structure agent and a parameter agent. The functions of these agents are detailed in the next two subsections, and the third subsection will detail the negotiation protocol and how the communication and the cooperation between those agents are realized.

A. Structure Agent Description

In general, the modeling of a fuzzy logic system requires the consideration of two important metrics which are the accuracy and the interpretability. The accuracy reflects the fuzzy system's capability of representing the real system in a faithful way. However, the interpretability refers to the ability of presenting the designed system in an understandable way. Although the accuracy and the interpretability objectives are generally in conflict, MOEAs can approximate a set of solutions named Pareto optimal solutions having various tradeoffs of these objectives. In the same context, a structure agent is created for multi-objective structure optimization purpose. Its principle function is to undergo an exploration step of the search space. The structure agent executes the proposed MOIP algorithm and takes into consideration the enhancement of both the accuracy and the interpretability metrics. The predictive performance of the system (accuracy) is expressed by the Root Mean Squared Error (RMSE) or the Mean Squared Error (MSE). The RMSE is used as objective function in the case of testing time series problems, while the MSE is used as objective function in the case of regression problems:

$$Objective\ 1: RMSE = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(y_t^j - y_{out}^j\right)^2} \tag{20}$$

$$Objective\ 1: MSE = \frac{1}{2*m}\sum_{j=1}^{m}\left(y_t^j - y_{out}^j\right)^2 \tag{21}$$

where m defines the samples number, y_t^j and y_{out}^j are respectively the desired output and the calculated output. The rule base complexity (interpretability) is expressed by the number of fuzzy rules:

$$Objective2: Interpretability = R \tag{22}$$

where R defines the rules number.

The structure agent firstly takes as input a population of different structures of HT2BFSs (antibodies) and then executes the proposed MOIP algorithm taking into account the following points:

- Learn and evolve the structures of the population and consider both the accuracy and the interpretability metrics as two objectives to optimize by the multi-objective algorithm.
- Use of the three immune programming operators (cloning, mutation and replacement) combined with a dominance concept to guide the search through an optimal Pareto-front of non-dominated solutions of HT2BFSs.
- Use of an elitist strategy based on the exploitation of an external archive of population in order to store elite solutions.
- Ensure a diversity maintenance mechanism and keep a well-distributed front based on the crowding distance procedure.
- Generate as output an optimal set of evolved population (Pareto optimal solutions). These solutions have different structures and they illustrate the obtained set of HT2BFSs with different accuracy-interpretability tradeoff.

B. Parameter Agent Description

The parameter agent is an autonomous agent created for parameter tuning purpose to refine existing solutions. Its main function consists of applying the HHS algorithm as a hybrid evolutionary optimization algorithm to perform a parameter tuning phase. The selected parameters for adjustment are the interval type-2 Beta MF parameters (c, σ_L, σ_U, p_L, p_U, q_L, q_U) and the consequent parts of fuzzy rules. To do this, the parameter agent takes firstly as input a given HT2BFS structure, and then it encodes the parameters of this selected HT2BFS in a matrix representation and initializes the rest of the population at random. Next, it executes the HHS algorithm to evolve the population and generates finally an optimum matrix of tuned parameters. The parameter agent encodes at the end the best parameters found in the fixed structure to be its final output. It should be noted that the RMSE previously defined is used by this agent as an objective function.

C. Multi-Agent Architecture: Communication between the Structure Agent and the Parameter Agents

The MA_HT2BFS is a multi-agent system capable of distributing and organizing the optimization task between the structure agent (the initiator) and a number of parameter agents (the participants). In such system, cooperation and interaction between agents take place to reach a common goal which is the generation of an optimal HT2BFS in a reduced time and with a less cost. An HT2BFS solution is called optimal or near optimal if it has the optimum structure with the optimum set of parameters. That means that this solution has the best distribution of nodes by levels in such a way that its evaluation meets the two desired objectives (accuracy and interpretability features).

To reach this goal, a negotiation protocol is needed to organize the communication and to guarantee the information exchange among agents. In fact, different negotiation protocols have been presented in the literature, the first and the most known one is the contract net protocol [50]. The main idea of this protocol is to decompose the problem into sub-problems by a central agent or a manager. This latter announces the sub-problems to the other system's agents, and then it collects their propositions to solve the problem. Here, the central agent is responsible for supervising the tasks execution and the treatment of their execution results. This protocol is more useful in conditions where all worker agents cooperate to attain the same goal.

In this work, we have used a negotiation protocol similar to the contract net protocol. In this protocol, the structure agent plays the role of an initiator agent or a manager, while the parameter agents are presented as participant agents. Fig. 6 illustrates the flowchart of the MA_HT2BFS, where *G* and *Iter* correspond respectively to the number of generations and the global number of iterations. *StrIter* and *PramIter_i* define respectively the structure iteration number and the parameter iteration number of agent *i*. The following description gives more details about the communication scenario assured by the negotiation protocol.

After the execution of the MOIP on a population of HT2BFSs by the structure agent, a Pareto front of non-dominated solutions is generated. Indeed, this Pareto contains a set of HT2BFSs solutions having different structures. Here, the main next task is to undergo a parameter tuning phase to all these structures. In fact, this task is difficult to do by one agent as the solutions have different structures. Therefore, the structure agent who acts as an initiator and a central agent, decomposes the main task to several sub-tasks in order to break its complexity. So, the initiator starts a negotiation session and sends a call to all the participants (parameter agents) announcing the beginning of a communication session. Then, the initiator sends each solution of the Pareto front to a participant agent. It should be noted that the number of the front's solutions is equal to the number of the called participant agents. At this level, participant agents answer by an approval message; they execute in parallel a parameter optimization step to refine solutions and then they send their proposals to the structure agent. This latter evaluates the received solutions based on their levels of accuracy and interpretability. And according to the validation of the stopping criterion, the structure agent decides if it will continue the learning process or not. The stopping criterion here is to find among the obtained high-quality HT2BFSs a sufficient solution representing the best trade-off between the objectives or to reach the maximum number of global iterations.

If one of the stopping criteria is attained, then the initiator sends an 'accept-proposal' message to the winner participant agent and takes its solution as the best final solution. The initiator also sends a 'reject-proposal' message to the other participants and closes the negotiation session.

If the stopping criterion is not yet reached, the initiator exposes another novel population for further structure optimization. This population is formed by the solutions proposals of the participant agents concatenated with the population already optimized by the structure agent in the previous round. As a result of this step, another Pareto front of optimal solutions is generated and will be sent for further parameter tuning, and in this case three alternatives are possible:

- If the same number of participant agents is needed (in comparison with the previous round), in this case the initiator sends a 'counter-proposal' message to all the existing participants containing the proposed structure to optimize (taken from the front).
- If the number of needed participant agents is less than the previous round (the number of the front's solutions is reduced), in this case the initiator will reject the extra agents by sending them a 'Quit' message. And, a set of 'counter-proposals' containing the new structures are sent to the rest of needed agents.
- If the number of needed participant agents is more than the previous round (the number of the front's solutions is increased), in this case, the initiator will create new participant agents to receive the extra solutions.

By this manner, instead of choosing just one solution from the front and tuning it, our multi-agent system aims to give the same chance to all the solutions to contribute to the learning process. This will prevent the other non-dominated solutions from being lost and enable their exploitation in the next round. As a result, the search space is enlarged, and the fact that all the optimized solutions will join the next population for further structure optimization, this will speed up the whole optimization process and will avoid the extra computations.

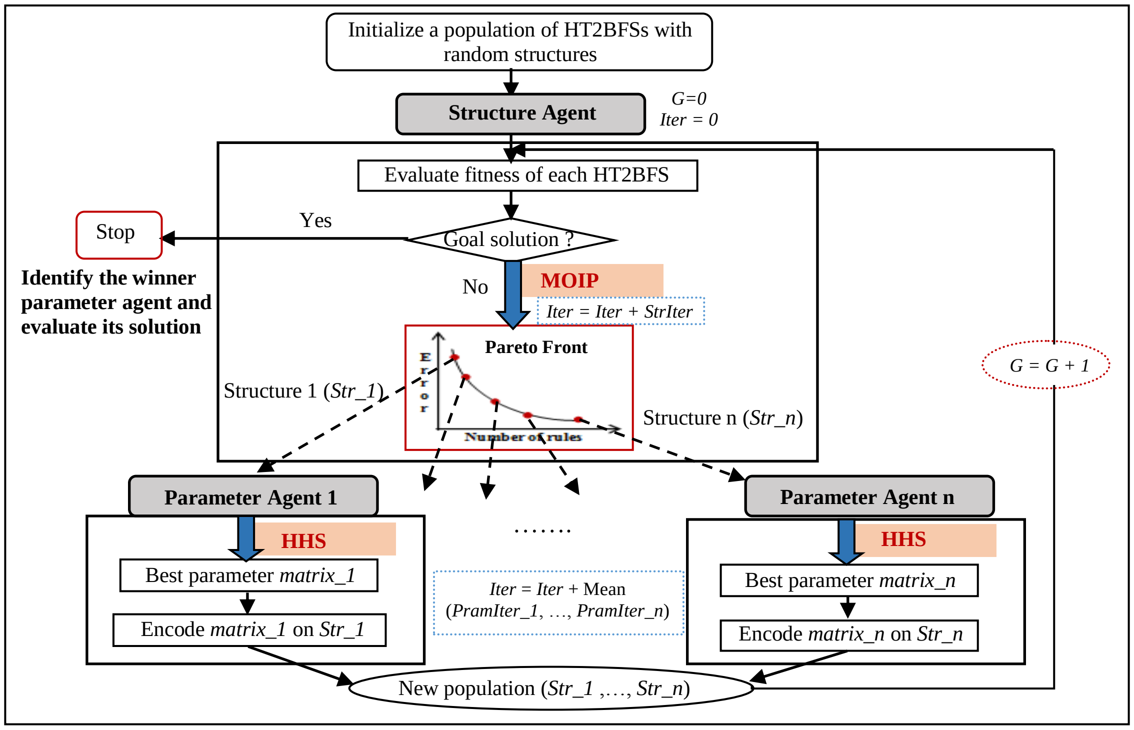


Fig. 6. The flowchart of the MA_HT2BFS

VII. Simulation results

In this section, the performance of the MA_HT2BFS is evaluated under both noise-free and noisy environments. The simulations include three kinds of forecasting time series problems. We also studied the impact of artificial additive noise for two cases of time series experiments. The experiments were also conducted over large-scale real-world regression problems.

The proposed system is implemented using the Matlab platform: the parallel computing toolbox and the distributed computing toolbox are exploited for the modeling of the multi-agent architecture, while the implementation of interval type-2 fuzzy logic systems was performed by the use of interval type-2 fuzzy logic toolbox [51]. The employed trees have degrees between 2 and 5 and depths between 2 and 4 (as a minimum and maximum). In addition, we used for each MF of each input a different FOU. For the MOIP training algorithm, the parameters are initialized as follows: population size = 20, probability of cloning *Pc* = 0.7 and probability of replacement *Pr* = 0.5. For the HHS tuning algorithm, the parameters are initialized with the following values: size of population = 20, *PARmin* = 1e-05, *PARmax* = 1, *HMCR* = 0.9, *c1*=0.2 and *c2*=0.7. Results are generated after 10 runs and then are averaged. To evaluate the efficiency of the MA_HT2BFS system, several comparisons with state of the art fuzzy/neural learning methods are made taking into account the accuracy (measured via RMSE), convergence speed (measured via the number of Function Evaluations (*NFEs*) and the global number of Iterations (*Iter*)) and Interpretability (measured based on the rule base complexity and fuzzy rules number(R)).

A. Mackey-Glass Chaotic Time Series

1) Case 1: Noise-free Mackey Glass time series

The Mackey-Glass chaotic time series (MG) [52] is a widely known benchmark problem usually adopted for performance comparison with different approaches. The MG is derived from the following differential equation:

$$\frac{d(x(t))}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t) \qquad (23)$$

Note that, if $\tau > 16.8$, the series has a chaotic behaviour. To make a meaningful comparison with related works, we use the same initial conditions as in these works. Hence, we choose a = 0.2, b = 0.1, c = 10, τ = 17 and $x(0)$ = 1.2. Two cases of input variables number are treated for this series. In the first case, we used 4 input variables for the prediction of MG at $x(t+6)$. These inputs are $x(t)$, $x(t-6)$, $x(t-12)$ and $x(t-18)$. In the second case, the $x(t+6)$ is predicted using 19 inputs which are $x(t)$, $x(t-1)$, $x(t-2)$, $x(t-3)$, ..., $x(t-18)$. 1000 observations were generated by applying the fourth order Runge-Kutta method in (23). The first 500 data points are exploited for training while the remaining 500 data points are exploited for testing.

After performing 8 global iterations and 173 *NFEs*, the obtained RMSE values for training and testing data are respectively 6.9421e-16 and 6.7523e-16 (in the case of 4 input variables). Tables I and II illustrate the simulation results for the two cases and make comparisons between our proposed model and other approaches from the literature. The training and testing RMSE are respectively given by $RMSE_{tr}$ and $RMSE_{ts}$ in the tables. The results indicate that the MA_HT2BFS can notably achieve better performance in the two cases of 4 and 19 inputs and outperforms other existing models.

Note that the MA_HT2BFS is compared with different Type-1 FLSs, Type-2 FLSs and neural network learning approaches. For Type-2 FLSs, our system is principally compared with the SA-IT2FLS [53] which is an interval type-2 fuzzy system optimized by the simulated annealing algorithm, and with the memetic-T2FS [54] which uses a variable-length genetic algorithm with a gradient descent technique for the structure and parameters learning of the interval type-2 fuzzy system. Our system is also compared to the support vector-based interval type-2 fuzzy system: TSK-SVR II [55] and to a general type-2 fuzzy system that uses vertical-slices centroid type-reduction method: GT2FLS-VSCTR [56].

For the TSK-SVR II [55] and the SA-IT2FLS [53] approaches, the used number of rules is respectively 32 and 16 rules. It is remarkable that the MA_HT2BFS with fewer rules (6 rules) could yield smaller error than its competitors. This is due to the hierarchical nature of the system and the use of a multi-objective optimization process which has a great impact on the reduction of the resulting rule base without affecting the system's prediction performance.

TABLE I. Comparison results of Mackey-Glass time-series in the case of 4 inputs

Method	$RMSE_{tr}$	$RMSE_{ts}$
ADANN-EDA [59]	1.2e-02	-
FLNFN-CCPSO [60]	8.2e-03	8.4e-03
HMDDE-BBFNN [61]	9.4e-03	1.7e-02
LNF [62]	7.0e-04	7.9e-04
NARMA [63]	6.3e-04	6.2e-04
FBBFNT [57]	9.9e-07	2.0e-06
MA_EFBBFNT [58]	4.1e-11	4.1e-11
GT2FLS-VSCTR [56]	3.9e-02	3.9e-02
TSK-SVR II [55]	-	7.0e-03
Memetic-T2FS [54]	3.1e-03	-
SA-IT2FLS [53]	9.0e-03	8.9e-03
MA_HT2BFS	**6.9e-16**	**6.7 e-16**

TABLE II. Comparison results of Mackey-Glass time-series in the case of 19 inputs

Method	$RMSE_{tr}$	$RMSE_{ts}$	$NFEs$
FNT [64]	2.7e-03	2.7e-03	-
FBBFNT_EGP&OPSO [65]	2.5e-05	2.5e-05	5,213,935
MA_EFBBFNT [58]	2.0e-06	2.0e-06	290,379
MA_HT2BFS	**5.5e-13**	**9.4e-13**	**414**

Other comparisons with existing neural network learning approaches are also discussed, and we can remark from Table I that the evolutionary neural system FBBFNT [57] can generate high rates of accuracy but using a huge number of

NFEs (more than 800,000). In the case of the MA_EFBBFNT [58] which is an extended version of FBBFNT that integrates a multi-agent architecture for training, this system succeeds to generate similar low training and testing errors and with much fewer number of function evaluations. In spite of this great improvement in *NFEs*, the function evaluations number of MA_EFBBFNT is still high (more than 158,000). As compared with the results of these systems, the proposed MA_HT2BFS seems to have better performance in terms of reaching comparable errors with minimum *NFEs* (173).

In fact, the huge decrease in the number of function evaluation is due to different reasons: firstly, the use of a clustering technique in the initialization step allows the generation of an initial population composed by relatively good solutions. This initialization process combined with the use of the powerful reasoning capacities of type-2 fuzzy modeling yield to more quality outputs, and this could save many iterations of optimization. Note that the FBBFNT based systems [57], [58] use an initial random population with totally random parameters which requires more generations of evolutionary optimization. On the other hand, the multi-agent architecture has also a powerful effect on the convergence speed of our algorithm. The parallel training and cooperation of several agents accelerate a lot the whole optimization process.

2) *Case 2: Noisy Mackey Glass time series*

Since type-2 fuzzy systems are supposed to handle higher uncertainty levels in comparison to their counterparts, the MA_HT2BFS has been tested for the Mackey glass time series when an additive noise is present in the training and/or testing data and was compared with other existing systems. The original data were affected by six different levels of Gaussian noise with zero mean and STDs (σ) equal to 0.04, 0.08, 0.1, 0.2, 0.3 and 0.4. To allow a fair comparison with other works, we use the same initial conditions as in [8], [31], [66], [67] and we adopt $\tau = 30$ and $x(0) = 1.2$. The same set of input variables is used for all comparison models. In this experimentation, $x(t-24)$, $x(t-18)$, $x(t-12)$ and $x(t-6)$ are used as four past values to predict $x(t)$. A total of 1000 data pairs were generated from the interval $t \in [124;1123]$. The first 500 data points are used for training while the remaining 500 data points are used for testing.

Based on different noise levels, we studied a comparison between the MA_HT2BFS presented in this work and another modified version of this system. The second considered system for comparison is the multi-agent hierarchical Beta fuzzy system (MA_HBFS) which employs type-1 Beta sub-fuzzy models. It should be noted that the same initial conditions and parameters were used for the two proposed systems. In this sense, Fig. 7 illustrates the MA_HT2BFS and the MA_HBFS prediction testing results in terms of training with noise level σ=0.1 and noise-free for test.

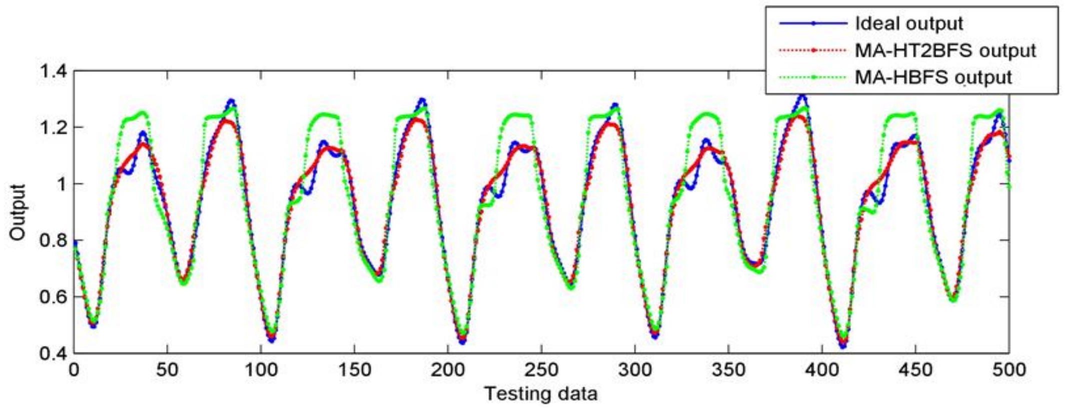


Fig. 7. Prediction results of MA_HT2BFS ($\text{RMSE}_{ts} = 0.031$) and MA_HBFS ($\text{RMSE}_{ts} = 0.078$) trained with noise level $\sigma = 0.1$ and noise-free for test

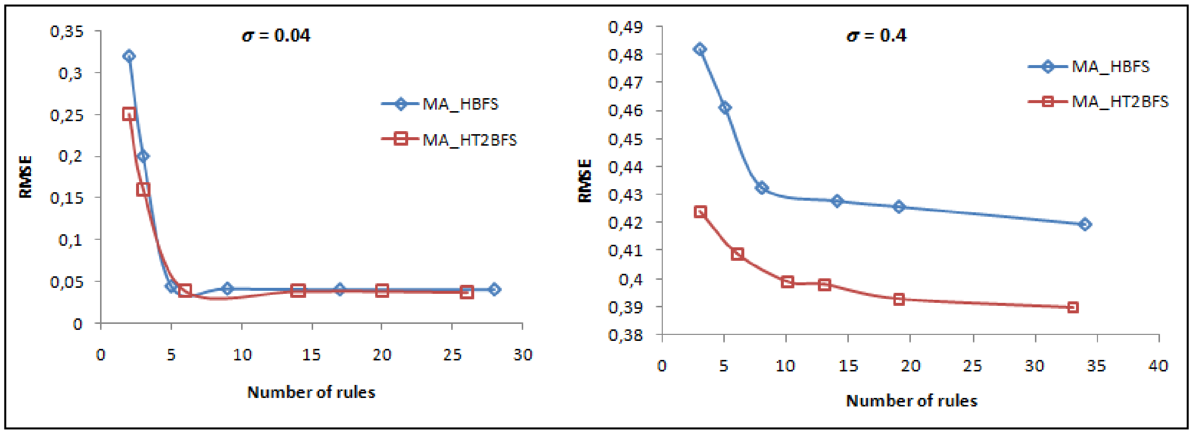


Fig. 8. Pareto fronts generated by MA_HT2BFS and MA_HBFS (left) when the noise in the training data is $\sigma = 0.04$ and (right) when the noise in the training data is $\sigma = 0.4$

Furthermore, we analyze the results generated by the MA_HT2BFS and the MA_HBFS when the training data are affected by low and high levels of noise. Fig. 8 presents the obtained Pareto fronts by the MA_HT2BFS and the MA_HBFS under noise levels σ = 0.04 and σ = 0.4. It is noticed from Fig. 8 that when the level of noise is low (σ = 0.04), the two systems achieve similar Pareto fronts. However, when the level of noise grows to σ = 0.4, the MA_HT2BFS generates a better Pareto front that dominates the front of the MA_HBFS. Therefore, we can conclude from the experiments that type-2 fuzzy sets have better noise tolerance than their type-1 counterparts. On the other hand, the robustness of the MA_HT2BFS over the MA_HBFS is also shown by training the MA_HT2BFS and the MA_HBFS using noise-free training data. Next, two levels of low and high Gaussian noise (σ=0.04 and σ=0.4) were added to the testing data in order to verify the robustness of the resulting hierarchical fuzzy systems. The results of this experiment are shown in Table III. The results show a better performance and tolerance to the noise of the MA_HT2BFS in comparison with the other type-1 model in the case of very noisy testing data (σ=0.4).

In addition, Fig. 9 shows the evolution of the testing error ($RMSE_{ts}$) as the noise level increases for the two types of FLSs. It should be noted that to make a fair comparison, we compare in Fig. 9 only solutions having the same number of rules (6 rules). We can observe that the impact of noise on the $\mathrm{RMSE_{ts}}$ values is not the same for the two systems. For low levels of noise, the MA_HT2BFS and the MA_HBFS systems give similar $RMSE_{ts}$, as the noise level increases, the MA_HT2BFS produce much lower $\mathrm{RMSE_{ts}}$ compared to its type-1 counterpart.

Table IV shows a comparison between our technique and other state of art techniques applied to noisy Mackey-Glass. For the training part, the training set is created by adding Gaussian noise with zero mean and STDs (σ) equal to 0.1 to the original data $x(t)$. Three sets are generated for testing: clean, σ = 0.1, and σ = 0.3. The best values having the lowest error are marked in bold. As shown in Table IV, the MA_HT2BFS outperforms the competing methods where although the SIT2FNN outperforms the other techniques in training and testing data, the MA_HT2BFS gives the best $RMSE_{ts}$ over testing data (with less or similar number of rules) where the difference to competing techniques increase when increasing the noise (with σ = 0.3).

TABLE III. Performance comparison in terms of noise-free training data and noisy testing data for Mackey-Glass time-series

Method	$RMSE_{tr}$	$RMSE_{ts}$			
	clean	clean	σ=0.04	σ=0.4	#R
MA_HBFS	7.9e-16	7.6e-16	0.038	0.301	6
MA_HT2BFS	1.2e-16	1.2e-16	0.035	0.182	6

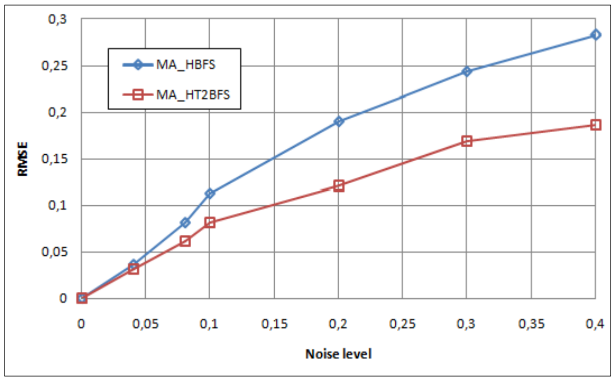


Fig. 9. Evolution of the $RMSE_{ts}$ values as the noise level increases for testing data in the case of Mackey-Glass time-series

TABLE IV. Comparison results of Mackey-Glass time-series in the case of noise level σ = 0.1

	Method	$RMSE_{tr}$	$RMSE_{ts}$			
		σ=0.1	clean	σ=0.1	σ=0.3	#R
Type-1	MA_HBFS	0.152	0.065	0.113	0.228	5
	SONFIN [66]	0.113	0.054	0.108	0.256	10
Type-2	IT2FNN-SVR [8]	0.127	0.046	0.088	0.215	6
	eT2FIS [67]	0.120	0.059	0.107	0.214	-
	SEIT2FNN [31]	0.123	0.049	0.097	0.212	5
	SIT2FNN [68]	**0.088**	0.041	0.087	0.215	5
	$T2HFIT^{M}$ [69]	0.123	0.042	0.135	0.365	-
	MA_HT2BFS	0.118	**0.039**	**0.082**	**0.181**	5

B. *Lorenz chaotic time series prediction*

1) Case 1: Noise-free Lorenz time series

The Lorenz system is a model of fluid motion between a hot surface and a cool surface [70]. This series is generated by the following ordinary nonlinear differential equations:

$$\begin{cases} \dot{x} = \sigma(y-x) \\ \dot{y} = -y - xz + rx \\ \dot{z} = xy - bz \end{cases} \qquad (24)$$

The x-coordinate of the equations is employed as the time series. The parameters in (24) are most commonly selected to be σ= 10, r = 28 and b = 8/3. The data are obtained by solving the described equations. For the prediction, $(t-4)$,$x(t-3)$, $x(t-2)$ and $x(t-1)$ are used as the inputs of the system while x(t) is the output. 1000 observations are generated, the first 500 data pairs are employed for training and the other 500 are used for the test.

Table V shows the comparison for MA_HT2BFS against other techniques where the MA_HT2BFS gives the best compromise between solution quality, the convergence speed and the rule base complexity.

TABLE V. COMPARISON RESULTS OF LORENZ TIME-SERIES

Method	$RMSE_{tr}$	$RMSE_{ts}$	*Iter*	*NFEs*	*#R*
LNF [62]	3.9e-03	8.1e-03	-	-	-
RBLM-RNN [71]	1.8e-02	3.0e-02	1000	-	-
FBBFNT [57]	7.4e-08	1.0e-07	3872	204,911	-
MA_HT2BFS	**3.3e-16**	**2.1e-16**	**6**	**106**	**5**

2) *Case 2: Noisy Lorenz time series*

We have also tested with noisy Lorenz time series. Different levels of Gaussian noise with zero mean and σ-deviation are added to the training and testing data, i.e., $\sigma = \{0.04, 0.08, 0.1, 0.2, 0.3, 0.4\}$.

The robustness of the MA_HT2BFS under a noisy environment is compared with that of the MA_HBFS as shown below. For the training part, the two systems were trained using clean training data (no noise was added), while for the testing part, two levels of Gaussian noise were added including testing with σ = 0.04 and σ = 0.4 respectively. The results of this experiment are shown in Table VI where the MA_HT2BFS can clearly outperform its type-1 counterpart specifically in the case of very noisy testing data (σ = 0.4). In Fig. 10, we show the impact of increasing the levels of noise on the $RMSE_{ts}$values for the two proposed typ-1 and type-2 hierarchical fuzzy models. We can remark from the figure that the MA_HT2BFS achieves significantly lower errors in comparison with the MA_HBFS as the noise increases. This affirms the noise resilience abilities of the MA_HT2BFS as compared with its type-1 counterpart.

TABLE VI. PERFORMANCE COMPARISON IN TERMS OF NOISE-FREE TRAINING DATA AND NOISY TESTING DATA FOR LORENZ TIME-SERIES

Method	$RMSE_{tr}$	$RMSE_{ts}$			
	clean	clean	σ=0.04	σ=0.4	*#R*
MA_HBFS	8.1e-16	9.4e-16	0.036	0.419	7
MA_HT2BFS	7.6e-16	7.8e-16	0.033	0.282	7

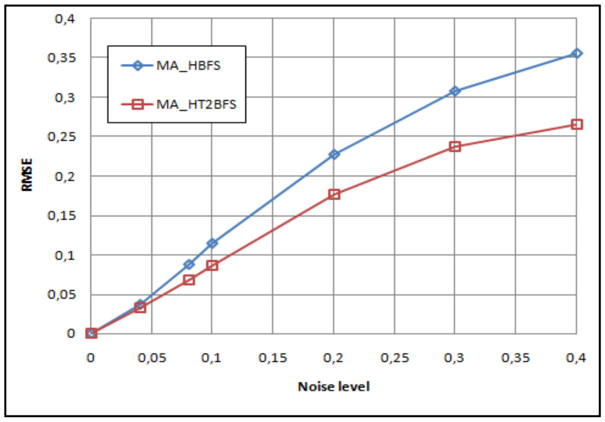


Fig. 10. Evolution of the $RMSE_{ts}$ values as the noise level increases for testing data in the case of Lorenz time series

C. *Sunspot time series*

We have also employed the Sunspot time series data set which presents a real world non-stationary and highly-complex time series showing the annual average relative number of observed sunspot [72]. The dataset is recorded between years 1700-1979. The training data are formed by data points between 1700 and 1920, the testing data are divided into two sets, the first set is from 1921 to 1955 and the second is from 1956 to 1979. The inputs of the system are $y(t-4)$, $y(t-3)$, $y(t-2)$ and $y(t-1)$ and the output is $y(t)$. The dataset is available from: http://www.ngdc.noaa.gov/stp/solar/ssndata.html.

Based on the different performance measures, Table VII lists the simulation results of our system and makes a comparison with other related works. After accomplishing two generations (G=2), an optimal HT2BFS having 6 rules is generated with 2.3195e-16 value for training RMSE. The actual time series and the predicted output are illustrated through Fig. 11. From the results table, we can notice a significant improvement when applying the MA_HT2BFS in the different measures of performance in comparison with the other methods. For example, although the fuzzy wavelet neural system FWNN [73] shows an improvement in the number of global iterations (*Iter*=200) as compared with the FBBFNT neural system [57] (*Iter*=3821), but it still uses many rules in the prediction (16 rules). In our case, our model can reach better rates of accuracy (*RMSE*$_{ts2}$) in less time (*Iter*=6 and *NFEs*=98) and using less complex rule base (6 rules).

TABLE VII. COMPARISON RESULTS OF SUNSPOT NUMBER TIME-SERIES

Method	$RMSE_{tr}$	$RMSE_{ts1}$	$RMSE_{ts2}$	*Iter*	*NFEs*	*#R*
RFNN [74]	-	7.4e-02	2.1e-01	-	-	-
FWNN-S [73]	2.5e-01	3.3e-01	5.2e-01	200	-	16
FWNN-R [73]	2.3e-01	3.3e-01	6.8e-01	200	-	16
FWNN-M [73]	2.4e-01	3.1e-01	6.0e-01	200	-	16
FBBFNT [57]	3.1e-08	7.2e-07	8.0e-07	3821	631,075	-
FBBFNT_EIP &HBFOA [43]	1.9e-10	4.1e-10	7.2e-10	-	-	-
MA_HT2BFS	**2.3e-16**	**5.4e-16**	**3.2e-16**	**6**	**98**	**6**

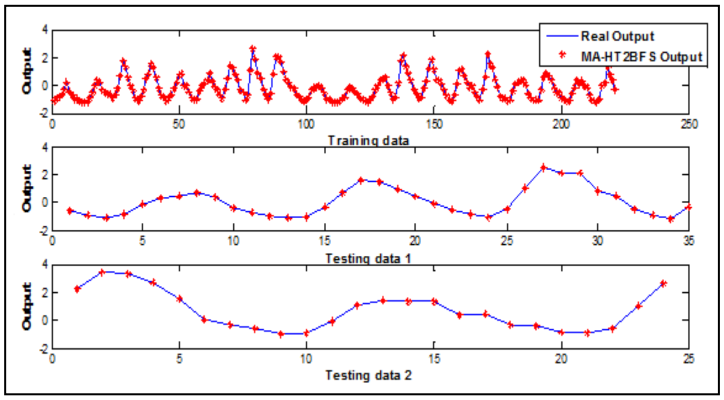


Fig. 11. The desired output and the predicted output for the training, test 1 and test 2 data in the case of sunspot number time series

D. *MA_HT2BFS for High-Dimensional Regression Problems*

In order to analyze the performance of the MA_HT2BFS in high-dimensional problems, we have employed four large-scale real-world regression problems from the KEEL project repository [75]. Table VIII shows the characteristics of the data sets which have been selected from the most complex problems of the KEEL project webpage (Available at http://www.keel.es/). In fact, the considered problems present an important challenge for the proposed system because of the high number of data and features (input variables). The data sets cover a range of input variables from 8 to 40 and a range of examples from 13750 to 22784.

TABLE VIII. DATA SETS CHARACTERISTICS

Problem	Abbr.	Variables	cases
Ailerons	AIL	40	13750
California Housing	CAL	8	20640
Elevators	ELV	18	16559
House-16H	HOU	16	22784

For all the problems, a 5-fold cross validation method was performed. Therefore, we divided each data set into 5 equal groups of samples where 4 groups are used for training and one group is used for the test. For each of the five partitions, six runs are executed resulting in a total of 30 runs per data set. The final results are averaged over the 30 runs. This experimentation does not aim to generate the lowest mean square error (MSE) in comparison with other works, but it aims to obtain in the same time the most accurate solution having a reduced number of rules and with the least number of function evaluations. The MA_HT2BFS is also compared to three state-of-the-art fuzzy systems for regression problems. Table IX presents the average rules number (*#R*), the average training MSE (MSE_{tr}), the average testing MSE (MSE_{ts}) and the average used *NFEs* for all the data sets. The best values having the lowest error and the minimum number of rules are marked in bold. Analyzing the results presented in Tables IX, we can see that the MA_HT2BFS presents competitive results in training and testing errors as compared with the other GFSs. Focusing on the used number of rules, it is remarkable that the proposed approach has the advantage of reaching competitive errors using fewer number of rules. We can see that despite the high number of examples, the number of rules generated by the MA_HT2BFS is the lowest in most of the cases. This is due to the great effect of the multi-objective immune programming mechanism which is able to significantly minimize the rules number while decreasing the error of the system. Regarding the number of executed function evaluations, the MA_HT2BFS succeeds to reach the desired compromise between the accuracy and the interpretability using a reduced number of *NFEs*. This is due to the hierarchical structure of the system and due to the parallel optimization process offered by the multi-agent architecture which reduces the needed learning iterations.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a hierarchical representation of interval type-2 beta fuzzy systems through a tree encoding scheme. Hierarchical type-2 fuzzy modeling has been considered in this study as a search problem and an optimization task in both structure and parameter spaces. For that, innovative hybrid stages of structure learning and parameter tuning tasks were applied in order to obtain a near-optimal system. To accomplish these tasks in the most optimal way, we have further employed a multi-agent architecture to efficiently parallelize and distribute the optimization tasks between a structure agent and a set of parameter agents. Both types of agents communicate and coordinate in order to generate an optimal hierarchical fuzzy system in a reduced time and with less cost. To do this, the structure agent applied a multi-objective structure learning phase by means of the MOIP algorithm which aims to obtain a set of improved structures of HT2BFSs. This phase is presented in a multi-objective context where the accuracy and the interpretability were considered as two main objectives to reach. For the tuning task, a number of parameter agents applied a parameter tuning phase by means of the HHS algorithm in order to adjust the parameters of the evolved structures.

The proposed hierarchical fuzzy design was implemented for both type-1 and type-2 FISs and the robustness of the MA_HT2BFS compared with the MA_HBFS was studied. The two types of systems were applied for time series prediction problems in the cases of absence of noise and under noisy environments. A comparative analysis was presented showing that when the used data are noise-free, or when the level of noise is low, the two types of FLSs gave close results. Therefore, the type-1 MA_HBFS is recommended to be used in such situations offering simpler computation and comparable error results. However, it has been observed that in the cases of higher noise levels (large amounts of uncertainty), the difference between the two systems becomes more evident. For example, in the case of MG time series, when the injected noise level is high (σ=0.4), the MA_HBFS achieved an $RMSE_{ts}$ of 0.284 in comparison to the MA_HT2BFS which gave an $RMSE_{ts}$ of 0.186 (i.e the MA_HT2BFS offered about 35% improvement over its type-1 counterpart).

Additionally, we presented many comparisons with other

TABLE IX. Average results of the different algorithms. Results in this table (MSE_{tr} and MSE_{ts}) should be multiplied by 10^{-8}, 10^{+9}, 10^{-6} or 10^{+8} in the case of AIL, CAL, ELV or HOU respectively

DATA SET	MA_HT2BFS				$FS_{\text{MOGFS}^{\text{e}}} + \text{TUN}^{\text{e}}$ [76]			FRULER [78]		$METSK-HD^{e}$ [77]		
	#R	MSE_{tr}	MSE_{ts}	*NFEs*	*#R*	MSE_{tr}	MSE_{ts}	*#R*	MSE_{ts}	*#R*	MSE_{tr}	MSE_{ts}
$AIL_{40/13750}$	**8.4**	**1.393**	**1.400**	397.6	15	1.955	2.000	8.5	1.404	48.4	**1.39**	1.51
$CAL_{8/20640}$	**6.3**	2.934	2.965	257.2	8.4	2.94	2.95	15.4	2.110	55.8	**1.64**	**1.71**
$ELV_{18/16559}$	7.3	**2.821**	**2.911**	326.5	8	9.00	9.00	**5.4**	2.934	34.9	6.75	7.02
$HOU_{16/22784}$	**6.4**	9.421	9.512	367.1	11.7	9.35	9.40	12.1	**8.005**	30.5	**8.29**	8.64

methods taken from the literature. The comparisons include recent works of neural systems and type-1/type-2 fuzzy or neuro-fuzzy systems. In most of the cases (noisy or noise free time series), the proposed MA_HT2BFS provided better testing error using similar or lower number of rules as compared with the other state of the art methods. Simulations on time series prediction problems showed good results and proved that the MA_HT2BFS outperforms the other competing methods even under a noisy environment.

Moreover, the performance of the proposed system was also examined in the case of high-dimensional problems. For this purpose, we have tested some large-scale regression problems and we compared the results to other well-known existing fuzzy systems. It is clear from the results that the MA_HT2BFS with much fewer rules could yield smaller or competitive error than the other existing GFSs.

Finally, for our future work, we will look to consider the proposed system for real world applications areas.

References

[1] J. M. Mendel, “Uncertain rule-based fuzzy logic systems: introduction and new directions,” Upper Saddle River, NJ, USA: Prentice-Hall, 2001, pp.131–184.

[2] J. M. Mendel, “Uncertainty, fuzzy logic, and signal processing,” *Signal Process.*, vol. 80, no. 6, pp. 913–933, Jun. 2000.

[3] O. Mendoza, P. Melín, and O. Castillo, “Interval type-2 fuzzy logic and modular neural networks for face recognition applications,” *Appl. Soft. Comput.*, vol. 9, no. 4, pp. 1377–1387, Sep. 2009.

[4] E. A. Jammeh, M. Fleury, C. Wagner, H. Hagras, and M. Ghanbari, “Interval type-2 fuzzy logic congestion control for video streaming across ip networks,” *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1123–1142, Oct. 2009.

[5] H. Hagras, “A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots,” *IEEE Trans.Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, Aug. 2004.

[6] N. N. Karnik and J. M. Mendel, “Applications of type-2 fuzzy logic systems to forecasting of time-series,” *Inf. Sci.*, vol. 120, no. 1, pp. 89–111, Nov. 1999.

[7] N. S. Bajestani and A. Zare, “Application of optimized type 2 fuzzy time series to forecast taiwan stock index,” in *2009 2nd International Conference on Computer, Control and Communication*, Pakistan, Feb. 2009, pp. 1-6.

[8] C. F. Juang, R. B. Huang, and W. Y. Cheng, “An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems,” *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 686–699, Aug. 2010.

[9] A. B. Cara, C. Wagner, H. Hagras, H. Pomares, and I. Rojas, “Multiobjective optimization and comparison of non singleton type-1 and singleton interval type-2 fuzzy logic systems,” *IEEE Trans. Fuzzy Syst.*,
vol. 21, no. 3, pp. 459–476, Jun. 2013.

[10] M. Tang, X. Chen, W. Hu, and W. Yu, “A Fuzzy Rule-Based Classification System Using Interval Type-2 Fuzzy Sets,” in *Integrated Uncertainty in Knowledge Modelling and Decision Making.*, Springer Berlin Heidelberg, 2011, pp. 72–80.

[11] M. Antonelli, D. Bernardo, H. Hagras, and F. Marcelloni, “Multi-objective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification,” *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 2, pp. 249-264, Apr. 2017.

[12] A. M. Alimi, R. Hassine, and M. Selmi, “Beta fuzzy logic systems: Approximation properties in the siso case,” Int. *J. Appl. Math. Comput. Sci.*, vol. 10, no. 4, pp. 857–875, 2000.

[13] A. M. Alimi, R. Hassine, and M. Selmi, “Beta fuzzy logic systems: Approximation properties in the mimo case,” *Int. J. Appl. Math. Comput. Sci.*, vol. 13, no. 2, pp. 225–238, 2003.

[14] M. L. Lee, H. Y. Chung, and F. M. Yu, “Modeling of hierarchical fuzzy systems,” *Fuzzy Sets Syst.*, vol. 138, no. 2, pp. 343–361, Sep. 2003.

[15] G. V. S Raju and J. Zhou, “Adaptive hierarchical fuzzy controller,” *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 4, pp. 973–980, Jul. 1993.

[16] L. X. Wang, “Analysis and design of hierarchical fuzzy systems,” *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 5, pp. 617–624, Oct. 1999.

[17] Y. Jarraya, S. Bouaziz, A. M. Alimi, and A. Abraham, “Evolutionary multi-objective optimization for evolving hierarchical fuzzy system,” in *Proc. IEEE Cong. Evol. Comput.*, Sendai, Japan, May 2015, pp. 3163–3170

[18] M. G. Joo and T. Sudkamp, “A method of converting a fuzzy system to a two-layered hierarchical fuzzy system and its run-time efficiency,” *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 1, pp. 93–103, Feb. 2009.

[19] K. Balázs, J. Botzheim, and L. T. Kóczy, “Hierarchical fuzzy system modeling by genetic and bacterial programming approaches,” in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, Sep. 2010, pp. 1–6.

[20] X. J. Zeng and J. A. Keane, “Approximation capabilities of hierarchical fuzzy systems,” *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 5, pp. 659–672, Oct. 2005.

[21] F. L. Chung and J. C. Duan, “On multistage fuzzy neural network modeling,” *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 2, pp. 125–142, Apr. 2000.

[22] P. Salgado, “Rule generation for hierarchical collaborative fuzzy system,” *Appl. Math. Model.*, vol. 32, no. 7, pp. 1159–1178, Jul. 2008.

[23] Z. Liu, C. L. P. Chen, Y. Zhang, and H. Li, “Type-2 hierarchical fuzzy system for high-dimensional data-based modeling with uncertainties,” *Soft Comput.*, vol. 16, no. 11, pp. 1945–1957, Nov. 2012.

[24] Y. Jarraya, S. Bouaziz, A. M. Alimi, and A. Abraham, “Evolutionary hierarchical fuzzy modeling of interval type-2 beta fuzzy systems,” in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, Oct. 2016, pp. 003481–003486.

[25] J. M. Mendel, “Computing derivatives in interval type-2 fuzzy logic systems,” *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, Feb. 2004.

[26] R. H. Abiyev and O. Kaynak, “Identification and control of time-varying plants using type-2 fuzzy neural system,” in *Int. Jt. Conf. Neural Netw.*, Atlanta, GA, USA, Jun. 2009, pp. 13–19.

[27] T. Kumbasar and H. Hagras, “A gradient descent based online tuning mechanism for pi type single input interval type-2 fuzzy logic controllers,” in in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Istanbul, Turkey, Aug. 2015, pp. 1–6.

[28] G. M. Mendez and M. A. Hernandez, “Interval Type-1 Non-Singleton Type-2 TSK Fuzzy Logic Systems Using the Hybrid Training Method RLS-BP,” in *2007 IEEE Symposium on Foundations of Computational Intelligence*, Honolulu, HI, USA, Apr. 2007, pp.–374.

[29] N. Baklouti and A. M. Alimi, “Real time PSO based adaptive learning type-2 fuzzy logic controller design for the iRobot Create robot,” in *2013 International Conference on Individual and Collective Behaviors in Robotics*, Sousse, Tunisia, Dec. 2013, pp. 15–20.

[30] O. Castillo, P. Melin, A. Alanis, O. Montiel, and R. Sepulveda, “Optimization of interval type-2 fuzzy logic controllers using evolutionary algorithms,” *Soft Comput.*, vol. 15, no. 6, pp. 1145–1160, Jun. 2011.

[31] C. F. Juang and Y. W. Tsao, “A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning,” IEEE *Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, Dec. 2008.

[32] F. Jiménez, G. Sánchez, and P. Vasant, “A multi-objective evolutionary approach for fuzzy optimization in production planning,” *J. Intell. Fuzzy Syst.*, vol. 25, no. 2, pp. 441–455, 2013.

[33] M. Cococcioni, P. Ducange, B. Lazzerini, and F. Marcelloni, “A pareto-based multi-objective evolutionary approach to the identification of mamdani fuzzy systems,” *Soft Comput.*, vol. 11, no. 11, pp. 1013–1031, Sep. 2007.

[34] H. Ishibuchi and Y. Nojima, “Evolutionary multiobjective optimization for the design of fuzzy rule-based ensemble classifiers,” *Int. J. Hybrid Intell. Syst.*, vol. 3, no. 3, pp. 129–145, Aug. 2006.

[35] A. Starkey, H. Hagras, S. Shakya, and G. Owusu, “A multi-objective genetic type-2 fuzzy logic based system for mobile field workforce area optimization,” *Inf. Sci.*, vol. 329, pp. 390 – 411, Feb. 2016. Special issue on Discovery Science.

[36] G. Weiss, “Multiagent systems: a modern approach to distributed artificial intelligence,” MIT press, Cambridge, MA, USA, 1999.

[37] M. Ammar, S. Bouaziz, A. M. Alimi, and A. Abraham, “Hybrid harmony search algorithm for global optimization,” in *2013 World Congress on Nature and Biologically Inspired Computing*, Fargo, USA, Oct. 2013, pp. 69–75.

[38] J. M. Mendel, M. R. Rajati, and P. Sussner, “On clarifying some definitions and notations used for type-2 fuzzy sets as well as some recommended changes,” *Inf. Sci.*, vol. 340, pp. 337–345, May 2016.

[39] J. M. Mendel, R. I. John, and F. Liu, “Interval type-2 fuzzy logic systems made simple,” *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 808–821, Dec. 2006.

[40] A. M. Alimi, “Beta neuro-fuzzy systems,” *TASK Quarterly Journal, Special Issue on Neural Networks*, vol. 7, no. 1, pp. 23–41, 2003.

[41] Q. Liang and J. M. Mendel, “An introduction to type-2 tsk fuzzy logic systems,” in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Seoul, South Korea, Aug. 1999, vol. 3, pp. 1534–1539.

[42] K. Deb, “Multi-objective optimization using evolutionary algorithms,” John Wiley & Sons, Inc. New York, NY, USA , vol. 16, 2001.

[43] S. Bouaziz, A. M. Alimi, and A. Abraham, “Evolving flexible beta basis function neural tree for nonlinear systems,” in *Int. Jt. Conf. Neural Netw*., Dallas, USA, Aug. 2013, pp. 1–8.

[44] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm formulti-objective optimization: Nsga-ii,” in *International Conference on Parallel Problem Solving From Nature*, Springer, 2000, pp. 849–858.

[45] Z. W. Geem, J. H. Kim, and G. V. Loganathan, “A new heuristic optimization algorithm: harmony search,” *Simulation*, vol. 76, no. 2, pp. 60–68, Feb. 2001.

[46] S. L. Chiu, “Fuzzy model identification based on cluster estimation,” *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 267–278, May. 1994.

[47] B. De Baets H. Vernieuwe and N. E. C Verhoest, “Comparison of clustering algorithms in the identification of takagi–sugeno models: A hydrological case study,” *Fuzzy Sets Syst.*, vol. 157, no. 21, pp. 2876–2896, Nov. 2006.

[48] Q. Ren, L. Baron, and M. Balazinski, “Type-2 takagi-sugeno-kang fuzzy logic modeling using subtractive clustering,” in *2006 Annual Meeting of the North American Fuzzy Information Processing Society*, Montreal, Canada, Jun. 2006, pp. 120–125.

[49] N. K. Kasabov and Qun Song, “Denfis: dynamic evolving neural-fuzzy inference system and its application fortime-series prediction,” *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[50] R. G. Smith and R. Davis, “Frameworks for cooperation in distributed problem solving,” *IEEE Trans. Syst., Man,Cybern.*, vol. 11, no. 1, pp. 61–70, Jan. 1981.

[51] J. Castro, O. Castillo, P. Melin, L. Martínez, S. Escobar, and I. Camacho, “Building fuzzy inference systems witht he interval type-2 fuzzy logic toolbox,” in *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, *Advances in Soft Computing*, Berlin, Heidelberg, Springer, 2007, vol. 41, pp. 53–62.

[52] M. C. Mackey and L. Glass, “Oscillation and chaos in physiological control systems,” *Science*, vol. 197, no. 4300, pp. 287–289, Jul. 1977.

[53] M. Almaraashi and R. John, “Tuning of type-2 fuzzy systems by simulated annealing to predict time series,” in *Proceedings of the World Congress on Engineering*, London, U.K , Jul. 2011, vol. 2, pp. 976–980.

[54] I. Castro Leon and P. C. Taylor, “Memetic type-2 fuzzy system learning for load forecasting,” in *2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology* (IFSA-EUSFLAT-15), Gijon, Spain., Jun. 2015.

[55] V. Uslan, H. Seker, and R. John, “A support vector-based interval type-2 fuzzy system,” in *IEEE Int. Conf. Fuzzy Syst.*, Beijing, China, Jul. 2014, pp. 2396–2401.

[56] M. Almaraashi, R. John, A. Hopgood, and S. Ahmadi, “Learning of interval and general type-2 fuzzy logic systems using simulated annealing: Theory and practice,” *Inf. Sci.*, vol. 360, pp. 21–42, Sep. 2016.

[57] S. Bouaziz, H. Dhahri, A. M. Alimi, and A. Abraham, “Evolving flexible beta basis function neural tree using extended genetic programming & hybrid artificial bee colony,” *Appl. Soft. Comput.*, vol. 47, pp. 653–668, Oct. 2016.

[58] M. Ammar, S. Bouaziz, A. M. Alimi, and A. Abraham, “Multi-agent evolutionary design of flexible beta basis function neural tree,” in *Int. Jt. Conf. Neural Netw.*, Beijing, China, Sep. 2014, pp. 1265–1271.

[59] J. P. Donate, X. Li, G. G. Sánchez, and A. S. de Miguel, “Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm,” Neural Comput Appl., vol. 22, no. 1, pp. 11–20, Jan. 2013.

[60] C. J. Lin, C. H. Chen, and C. T. Lin, “A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications,” *IEEE Trans. Syst., Man, Cybern., Part C (Applications and Reviews)*, vol. 39, no. 1, pp. 55–68, Jan. 2009.

[61] H. Dhahri, A. M. Alimi, and A. Abraham, “Hierarchical multi-dimensional differential evolution for the designof beta basis function neural network,” *Neurocomputing*, vol. 97, pp. 131–140, Nov. 2012.

[62] A. Miranian and M. Abdollahzade, “Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[63] C. G. da Silva, “Time series forecasting with a non-linear model and the scatter search meta-heuristic,” *Inf. Sci.*, vol. 178, no. 16, pp. 3288–3299, Aug. 2008. Including Special Issue: Recent advances in granular computing.

[64] Y. Chen, B. Yang, J. Dong, and A. Abraham, “Time-series forecasting using flexible neural tree model,” *Inf. Sci.*, vol. 174, no. 3, pp. 219–235, Aug. 2005.

[65] S. Bouaziz, H. Dhahri, A. M. Alimi, and A. Abraham, “A hybrid learning algorithm for evolving flexible beta basis function neural tree model,” *Neurocomputing*, vol. 117, pp. 107–117, Oct. 2013.

[66] C. F. Juang and C. T. Lin, “An online self-constructing neural fuzzy inference network and its applications,” *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.

[67] S. W. Tung, C. Quek, and C. Guan, “et2fis: An evolving type-2 neural fuzzy inference system,” *Inf. Sci.*, vol. 220, pp. 124–148, Jan. 2013.

[68] Y. Y. Lin, S. H. Liao, J. Y. Chang, and C. T. Lin, “Simplified interval type-2 fuzzy neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 959–969, May 2014.

[69] V. OJHA, V. Snasel, and A. Abraham, “Multiobjective programming for type-2 hierarchical fuzzy inference trees,” *IEEE Trans. Fuzzy Syst.*, vol. PP, no. 99, pp. 1–1, Apr. 2017.

[70] E. N. Lorenz, “Deterministic Nonperiodic Flow,”, *J. Atmos. Sci.*, vol. 20, no. 2, pp. 130-141, Mar. 1963.

[71] D. T. Mirikitani and N. Nikolaev, “Recursive bayesian recurrent neural networks for time-series modeling,” *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 262–274, Feb. 2010.

[72] A J Izeman, “J.r. wolf and the zurich sunspot relative numbers,” *Math. Intell.*, vol. 7, no. 1, pp. 27–33, Mar. 1985.

[73] S. Yilmaz and Y. Oysal, “Fuzzy wavelet neural network models for prediction and identification of dynamical systems,” *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1599–1609, Oct. 2010.

[74] R. A. Aliev, B. G. Guirimov, B. Fazlollahi, and R. R. Aliev, “Evolutionary algorithm-based learning of fuzzy neural networks. part 2: Recurrent fuzzy neural networks,” *Fuzzy Sets Syst.*, vol. 160, no. 17, pp. 2553-2566, Sep. 2009.

[75] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit,V. M. Rivas, J. C. Fernández, and F. Herrera, “Keel: a software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput.*, vol. 13, no. 3, pp. 307-318, Feb. 2009.

[76] R. Alcala, M. J. Gacto, and F. Herrera, “A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems,” *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 666–681, Aug. 2011.

[77] M. J. Gacto, M. Galende, R. Alcalá, and F. Herrera, “Metsk-hde: A multiobjective evolutionary algorithm to learn accurate tsk-fuzzy systems in high-dimensional and large-scale regression problems,” *Inf. Sci.*, vol. 276, pp. 63–79, Aug. 2014.

[78] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, “Fruler: Fuzzy rule learning through evolution for regression,” *Inf. Sci.*, vol. 354, pp. 1-18, Aug. 2016.

Appendix A: Nomenclature

IT2BMF: Interval Type-2 Beta Membership Function.
IT2BFS: Interval Type-2 Beta Fuzzy System.
HT2BFS: Hierarchical Interval Type-2 Beta Fuzzy System.
E_HT2BFS: Evolutionary Hierarchical Interval Type-2 Beta Fuzzy System.
MA_HT2BFS: Multi-Agent Hierarchical Interval Type-2 Beta Fuzzy System.
MA_HBFS: Multi-Agent Hierarchical Beta Fuzzy System.
MOIP: Multi-Objective Immune Programming.
HHS: Hybrid Harmony Search.