Weighted Fuzzy Genetic Programming Algorithm for Structure and Parameters Selection of Fuzzy Systems for Nonlinear Modelling

Krystian Łapa and Krzysztof Cpałka

Abstract In this paper a weighted fuzzy genetic programming algorithm for selection of structure and parameters of fuzzy systems for nonlinear modelling is proposed. This method is based on fuzzy genetic programming and innovations in this method concern, among the others, using weights of fuzzy aggregation operators, using weights of fuzzy rules, using fitness function criteria designed for fuzzy genetic programming and using dynamic links between fuzzy rules and fuzzy rules base. The proposed method was tested with use of typical nonlinear modelling problems.

Keywords Genetic programming · Weights · Fuzzy system · Nonlinear modelling · Dynamic systems

1 Introduction

The aim of nonlinear modelling is to obtain model with behaviour as close as possible to the testing object. Nonlinear modelling can concern many area of interest, such as physics, engineering, biology, chemistry, etc. and it is an important topic in the literature [16]. One of the mostly used systems for nonlinear modelling are fuzzy systems [20]. These systems can achieve high accuracy and interpretable knowledge in a form of fuzzy rules [9]. Most papers in the literature concerns selecting parameters of fuzzy system with specified structure of the system. For example, genetic algorithms [14], population-based algorithms [5], differential evolution [8], etc. are used to achieve that. From the other hand, an interesting group of methods for solving nonlinear modelling are genetic programming methods [21].

K. Łapa (✉) · K. Cpałka
Institute of Computational Intelligence, Częstochowa University of Technology, Częstochowa, Poland
e-mail: krystian.lapa@iisi.pcz.pl

K. Cpałka
e-mail: krzysztof.cpalka@iisi.pcz.pl

L. Borzemski et al. (eds.), *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology—ISAT 2016—Part I*, Advances in Intelligent Systems and Computing 521, DOI 10.1007/978-3-319-46583-8_13

These methods allow obtaining system structures in a form of computer programs (trees). These systems can also achieve high accuracy. However, disadvantage of typical genetic programming methods is lack in interpretability.

1.1 Fuzzy Systems

Fuzzy systems [20] are based on fuzzy logic and fuzzy rules. Each fuzzy rule consists of fuzzy sets which can represent linguistic values '*low*', '*medium*', '*high*', etc. Fuzzy rules take interpretable form {IF … THEN …}. The interpretability can result not only from the low number of fuzzy rules and fuzzy sets, but also from the semantic of appropriate selected parameters of fuzzy sets [9]. The semantic clarifies understanding of how the systems (models [9], classifiers [12], control systems [13]) work. It is worth to mention that in the nonlinear modelling problems the interpretability is an important issue [9]. It emerges from possibilities of understanding how the current object works and it allows us to model the typical states of the object.

1.2 Genetic Programming

Genetic programming (GP) is, in a general, a computational intelligence method for designing systems (structures) in a form of computer programs for solving optimization problems [21]. These systems can be used as controllers, models of objects, classifiers, etc. The main difference between genetic programming and other computational intelligence methods is a possibility of creation tree structures with use of mathematical operators. GP and other evolutionary algorithms (like evolutionary strategies [20], evolutionary programming [17], genetic algorithm [7], etc.) rely on a population of solutions. These methods are based on a natural evolution (using mechanisms like natural selection, inheritance, survival, etc.) which gives them an advantage over other methods used for optimization problems like analytic methods, gradient methods and random methods (see e.g. [20]).

1.3 Genetic Programming Trees

Typical genetic programming tree contains nodes and leafs (both of them are noted as tree elements in further part of this paper). Each node contains mathematical operator which decides how node works and usually has two child tree elements (see Fig. 1). Each leaf contains a real number value or connection to real number value from the system input (see Fig. 2). The whole tree structure is represented by the root (main node) and child tree elements (see Fig. 3). The output of the system

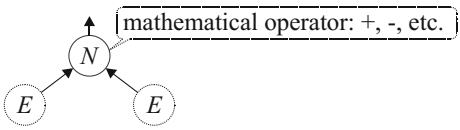

Fig. 1 GP node N and two connected tree elements E

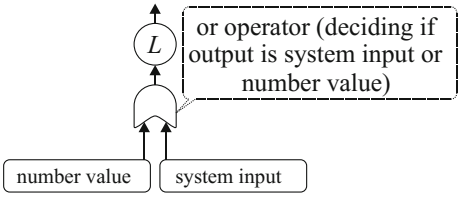

Fig. 2 GP leaf L connected with number value and system input

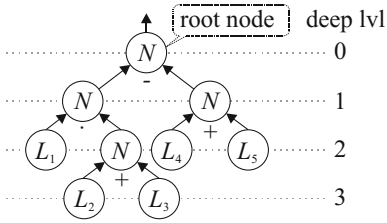

Fig. 3 An example of GP tree with randomly assigned structure and mathematical operators and output calculated as: $(L_1 \cdot (L_2 + L_3)) - (L_4 + L_5)$

(tree) is calculated recursively starting from the root node with use of defined mathematical operators. The mathematical operators are divided into: single argument operators (e.g. "cos(·)"), two argument operators (e.g. "+") and multi arguments operators (e.g. "avg(·)").

1.4 Fuzzy Genetic Programming

In the fuzzy genetic programming a mathematical operators are replaced by fuzzy operators (such as AND, OR, etc.) and leaves can be connected not to the system inputs but to the input of fuzzy sets. In [4] standard fuzzy operator AND was used, in [7] additional fuzzy operators: OR, NOT, '*greater*', '*lesser*' and '*near*' were used. In [14] operators AND, parent operators OR and fuzzy set operator NOT were used. In paper [18] a selected group of operators was used (with multiple versions of AND and OR operators).

1.5 Paper Aim

The aim of this paper is, among the others, to present impact of used weights in proposed weighted fuzzy genetic programing algorithm and to provide accurate and

interpretable fuzzy rules for considered simulation problems. The proposed method is based on fuzzy genetic programming. This method can be distinguished by: (a) use of flexible triangular norms with weights of arguments as fuzzy operators AND and OR, (b) use of weights of fuzzy rules, (c) use of operator NOT for fuzzy sets, (d) use of new encoding of the system, (e) use of fitness function with complexity of the system and new criteria of correct notations of fuzzy rules and (f) use of population algorithm adapted to the proposed system to select GP tree structure and parameters. The full description of the method and learning algorithm is presented in Sect. 2.

1.6 Paper Structure

The structure of the proposed paper consists of: Sect. 2 with description of the proposed method, Sect. 3 with presentation of simulation results and Sect. 4 with conclusions.

2 Proposed Method Description

2.1 Description of Fuzzy System

In this paper a Mamdani type fuzzy system [20] was used, where fuzzy rules can be defined as:

$$R_k : \begin{pmatrix} \text{IF}\left(\bar{x}_1 \text{ IS}[\text{NOT}]A_{1,k}\right) \text{ AND/OR} \ldots \text{AND/OR}\left(\bar{x}_n \text{ IS}[\text{NOT}]A_{1,k}\right) \\ \text{THEN}\left(y_1 \text{ IS } B_{1,k}\right), \ldots, \left(y_m \text{ IS } B_{m,k}\right) \end{pmatrix}, \quad (1)$$

where n is the number of inputs, m is the number of outputs, $\bar{\mathbf{x}} = [\bar{x}_1, \ldots, \bar{x}_n] \in \mathbf{X}$, $\mathbf{y} = [y_1, \ldots, y_m] \in \mathbf{Y}$, $A_{1,k}, \ldots, A_{n,k}$ are input fuzzy sets and $B_{1,k}, \ldots, B_{m,k}$ are output fuzzy sets. In the proposed method fuzzy rules (1) are represented by GP trees (see Fig. 4). Using genetic programming trees creates the need of using fuzzy sets' base, which allows fuzzy sets to connect to the leaves of trees. Proposed fuzzy sets' base $\mathbf{C}$ (stored for each input or output) is defined as:

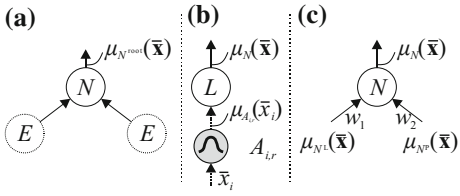

Fig. 4 Proposed structure of: **a** tree, **b** leaf, **c** node

Table 1 Parameters of proposed tree elements ('-' stands for parameters not used in the current type of tree element)

Tree element	Set of parameters							
E (element)	l	o	i	r	w_1	w_2	N^{L}	N^{P}
L (leaf)	$l = 1$	o	i	r	–	–	–	–
N (node)	$l = 0$	o	i	–	w_1	w_2	N^{L}	N^{P}

Table 2 Output of proposed tree elements (leaves and nodes)

Output value $(\mu_N(\bar{\mathbf{x}}))$	o	l	Operator name
$\mu_{A_{i,r}}(\bar{x}_i)$	0	1	IS
$1 - \mu_{A_{i,r}}(\bar{x}_i)$	1	1	IS NOT
$T^*\{\mu_{N^{\mathrm{L}}}(\bar{\mathbf{x}}), \mu_{N^{\mathrm{P}}}(\bar{\mathbf{x}}); w_1, w_2\}$	0	0	AND
$S^*\{\mu_{N^{\mathrm{L}}}(\bar{\mathbf{x}}), \mu_{N^{\mathrm{P}}}(\bar{\mathbf{x}}); w_1, w_2\}$	1	0	OR

$$\mathbf{C} = \left\{A_{1,1}, \ldots, A_{1,R}, \ldots, A_{n,1}, \ldots, A_{n,R}, B_{1,1}, \ldots, B_{1,R}, \ldots, B_{m,1}, \ldots, B_{m,R}\right\}. \quad (2)$$

Each input fuzzy set $A_{i,r}$ from fuzzy set base (2) is represented by the membership function $\mu_{A_{i,r}}(\bar{x})$, while each output fuzzy set $B_{j,r}$ is represented by the membership function $\mu_{B_{j,r}}(\bar{y})$. Each element of the tree (see Table 1) is described by a set of parameters: $l, o, i, r, w_1, w_2, N^{\mathrm{L}}$ and N^{P}. The parameter l decides how the element is treated (as node or leaf—see Table 1). The parameter o indicates operator of a given element: $o = 0$ for $l = 1$ stands for '*IS*', $o = 1$ for $l = 1$ stands for '*IS NOT*', $o = 0$ for $l = 0$ stands for '*AND*' and $o = 1$ for $l = 0$ stands for '*OR*' (see Table 2). The parameter i (for leaf) stands for input index of associated fuzzy set. The parameter r (for leaf) stands for index of associated fuzzy set $A_{i,r}$ (see Fig. 4b). The parameter w_1 stands for weight of left child node, w_2 stands for weight of right child node (see Fig. 4c), N^{L} stands for left child node and N^{P} stands for right child node. Taking into consideration mentioned parameters the output of any element of the tree can be calculated according to Table 2 (in these calculations a $T^*(\cdot)$ triangular t-norm with weights of arguments and $S^*(\cdot)$ triangular t-conorm with weights of arguments [20] were used).

The activation (firing) level of each fuzzy rule based on the structure presented in Fig. 4 is calculated as follows:

$$\tau_k(\bar{\mathbf{x}}) = \mu_{N_k^{\mathrm{root}}}(\bar{\mathbf{x}}), \quad (3)$$

where N_k^{root} stands for root of the tree of kth fuzzy rule ($k = 1, \ldots, K$, K stands for the number of fuzzy rules). Crisp output values of the system for each output j can be calculated (for example) with center of area method [20]:

$$\bar{y}_j(\bar{\mathbf{x}}) = \frac{\sum_{r=1}^{R} y_{j,r}^B \cdot \underset{k=1}{\overset{K}{S}} \left\{ T^* \left\{ \tau_k(\bar{\mathbf{x}}), \mu_{k,j}\left(y_{j,r}^B\right) \right\}; w_k^{\text{rule}} \right\}}{\sum_{r=1}^{R} \underset{k=1}{\overset{K}{S}} \left\{ T^* \left\{ \tau_k(\bar{\mathbf{x}}), \mu_{k,j}\left(y_{j,r}^B\right) \right\}; w_k^{\text{rule}} \right\}}, \tag{4}$$

where $y_{j,r}^B$ are centres of output fuzzy sets $B_{j,r}$, w^{rule} stands for fuzzy rule weight and $\mu_{k,j}\left(y_{j,r}^B\right)$ stands for membership function value of output fuzzy set $B_{j,k}$ calculated for discretization point $y_{j,r}^B$. This value can be calculated (using proposed encoding of the system) as:

$$\mu_{k,j}\left(y_{j,r}^B\right) = \mu_{B_{j,n_{j,k}^B}}\left(y_{j,r}^B\right), \tag{5}$$

where $n_{j,k}^B$ stands for index connecting k-th fuzzy rule with jth output fuzzy set (for example $n_{j=1,k=2}^B = 3$ means that the second fuzzy rule is associated with the third set of the first output $B_{1,3}$).

2.2 Encoding of the System

In the proposed approach encoding of the system (4) is based on the encoding of a tree elements $\mathbf{N}$ (see Table 1 and Fig. 4) as sets of parameters:

$$\mathbf{N} = \left\{ l, o, i, r, w_1, w_2, \mathbf{N}^{\text{L}}, \mathbf{N}^{\text{P}} \right\}, \tag{6}$$

where $\mathbf{N}^{\text{L}}$ and $\mathbf{N}^{\text{P}}$ encodes recursively child tree elements (these values are set to "no value" in case of leaves). The encoding of fuzzy system (4) is defined as:

$$\mathbf{X}_{ch} = \left\{ \mathbf{X}_{ch}^{\text{fsets}}, \mathbf{X}_{ch}^{\text{rules}} \right\}. \tag{7}$$

The part $\mathbf{X}_{ch}^{\text{fsets}}$ encodes parameters of fuzzy sets' base (represented by Gaussian membership functions) (2):

$$\mathbf{X}_{ch}^{\text{fsets}} = \left\{ \begin{array}{c} x_{1,1}^A, \sigma_{1,1}^A, \ldots, x_{1,R}^A, \sigma_{1,R}^A, \ldots, x_{n,1}^A, \sigma_{n,1}^A, \ldots, x_{n,R}^A, \sigma_{n,R}^A, \\ y_{1,1}^B, \sigma_{1,1}^B, \ldots, y_{1,R}^B, \sigma_{1,R}^B, \ldots, y_{m,1}^B, \sigma_{m,1}^B, \ldots, y_{m,R}^B, \sigma_{m,R}^B \end{array} \right\}, \tag{8}$$

thus the part $\mathbf{X}_{ch}^{\text{rules}}$ encodes fuzzy rules:

$$\mathbf{X}_{ch}^{\text{rules}} = \left\{ \mathbf{N}_1^{\text{root}}, n_{1,1}^B, \ldots, n_{m,1}^B, w_1^{\text{rule}}, \mathbf{N}_2^{\text{root}}, n_{1,2}^B, \ldots, n_{m,2}^B, w_2^{\text{rule}}, \ldots, \mathbf{N}_K^{\text{root}}, n_{1,K}^B, \ldots, n_{m,K}^B, w_K^{\text{rule}} \right\}, \tag{9}$$

where $\mathbf{N}_k^{\text{root}}$ is a root of the tree of k-th rule, $n_{j,k}^B$ is an index connecting k-th fuzzy rule with fuzzy set of j-th output, w_k^{rule} is weight of the rule. In the proposed method the part $\mathbf{X}_{ch}^{\text{fsets}}$ encoding parameters of fuzzy sets is processed by a genetic algorithm and the part $\mathbf{X}_{ch}^{\text{rules}}$ encoding fuzzy rules is processed by a genetic programming (for details see Sect. 2.5).

2.3 Initialization of the System

The parameters of fuzzy sets encoded in $\mathbf{X}_{ch}^{\text{fsets}}$ are initialized randomly with adjustments to the considered simulation problems. Next, the number of fuzzy rules $K \in \left[K^{\min}, K^{\max}\right]$ is chosen randomly. After the number of fuzzy rules is chosen, parameters of part $\mathbf{X}_{ch}^{\text{rules}}$ are initialized as follows:

$$\mathbf{X}_{ch}^{\text{rules}} = \left\{ \begin{array}{c} init\left(\mathbf{N}_1^{\text{root}}, 0, 0\right), U_c(1,R), \ldots, U_c(1,R), U_r(0,1), \\ init\left(\mathbf{N}_2^{\text{root}}, 0, 0\right), U_c(1,R), \ldots, U_c(1,R), U_r(0,1), \ldots, \\ init\left(\mathbf{N}_K^{\text{root}}, 0, 0\right), U_c(1,R), \ldots, U_c(1,R), U_r(0,1) \end{array} \right\}, \tag{10}$$

where $U_c(a,b)$ returns random integer value from the set $\{a, \ldots, b\}$, $U_r(a,b)$ returns random number value from the range $[a,b]$, $init(\mathbf{N}, d, e)$ (d stands for "deep lvl" of the tree—see Fig. 3, e stands for type of tree element) is an initialization function for tree elements. The function $init(\cdot)$ for nodes initialization (when $e = 0$ and $d < d^{\max}$, where $d^{\max}$ is maximum "deep lvl" of the tree) takes the following form:

$$init(\mathbf{N}, d, 0) = \left\{ \begin{array}{c} 0, U_c(0,1), U_c(1,n), U_c(1,R), U_r(0,1), U_r(0,1), \\ init\left(\mathbf{N}\{\mathbf{N}^{\text{L}}\}, d+1, U_c(0,1)\right), init\left(\mathbf{N}\{\mathbf{N}^{\text{P}}\}, d+1, U_c(0,1)\right) \end{array} \right\}, \tag{11}$$

where notation $\mathbf{N}\{a\}$ refers to gene a encoded in $\mathbf{N}$. It is worth to mention that this function initializes child elements of the tree recursively with increased value of deep of the tree (d). This process can stop if new element is a leaf or when deep of the tree reaches maximum lvl $d^{\max}$. The function $init(\cdot)$ for leaves initialization (when $e = 1$ or $d = d^{\max}$) takes the following form:

$$init(\mathbf{N}, d, 1) = \{1, U_c(0,1), U_c(1,n), U_c(1,R), U_r(0,1), U_r(0,1), null, null\}, \tag{12}$$

where *null* stands for genes with no assigned values.

2.4 System Evaluation

For evaluation of the system (4) the following fitness function was used:

$$\mathrm{ff}(\mathbf{X}_{ch}) = \underset{f=1}{\overset{F}{T^*}} \left\{ \mathrm{ffc}_f(\mathbf{X}_{ch}), w_f^{\mathrm{ffc}} \right\}, \tag{13}$$

where F stands for the number of fitness function components, component $\mathrm{ffc}_1(\mathbf{X}_{ch}) = \mathrm{ffacc}(\mathbf{X}_{ch})$ specifies the accuracy of the system (4), component $\mathrm{ffc}_2(\mathbf{X}_{ch}) = \mathrm{ffcom}(\mathbf{X}_{ch})$ specifies complexity of the system (4), component $\mathrm{ffc}_3(\mathbf{X}_{ch}) = \mathrm{ffsam}(\mathbf{X}_{ch})$ stands for a penalty for using the same fuzzy set multiple times by fuzzy rules (which is non-desired), component $\mathrm{ffc}_4(\mathbf{X}_{ch}) = \mathrm{ffmul}(\mathbf{X}_{ch})$ stands for a penalty for using the same input multiple times by single fuzzy rule (with is non-desired), w_f^{facc} $(f = 1, \ldots, F)$ are weights of components, $T^*\{\cdot\}$ is a n-argument extension of algebraic triangular norm with weights of arguments. The components aliases (ffcom, ffacc, ffsam, ffmul) were used to increase readability of the paper and presentation of the results.

The component $\mathrm{ffc}_1(\mathbf{X}_{ch})$ of function (13) is defined as:

$$\mathrm{ffc}_1(\mathbf{X}) = \frac{1}{m} \sum_{j=1}^{m} \frac{\frac{1}{Z} \sum_{z=1}^{Z} \left| d_{z,j} - \bar{y}_{z,j} \right|}{\max\limits_{z=1,\ldots,Z} \{d_{z,j}\} - \min\limits_{z=1,\ldots,Z} \{d_{z,j}\}}, \tag{14}$$

where Z is the number of rows of a learning sequence, $d_{z,j}$ is the desired output value of output j for input vector z $(z = 1, \ldots, Z)$, $\bar{y}_{z,j}$ is the real output value j calculated for the input vector $\bar{\mathbf{x}}_z$. Equation (14) takes into account the normalization of errors at different outputs of the system (4), which allows us to use function (14) in triangular norm used in function (13) (Table 3).

The component $\mathrm{ffc}_2(\mathbf{X}_{ch})$ of function (13) is defined as:

$$\mathrm{ffc}_2(\mathbf{X}_{ch}) = \frac{1}{K} \sum_{k=1}^{K} \frac{\mathrm{cm}\left(\mathbf{N}_k^{\mathrm{root}}\right)}{2^{lvl^{\max}} - 1}, \tag{15}$$

where $\mathbf{N}_k^{\mathrm{root}}$ is by default a part of structure $\mathbf{X}_{ch}^{\mathrm{rules}}$ of $\mathbf{X}_{ch}$ (this notation will be used in further part of this paper), denominator stands for maximum number of tree elements (Mersenne's number [15]), numerator stands for actual number of tree elements calculated according to the Table 4.

Table 3 Output of the $\mathrm{cm}(\cdot)$ function

$\mathrm{cm}(\mathbf{N})$	$\mathbf{N}\{l\}$
$1 + \mathrm{cm}\left(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\}\right) + \mathrm{cm}\left(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\}\right)$	0
1	1

Table 4 Output of the $\mathrm{sa}(\cdot)$ function

$\mathrm{sa}(\mathbf{N}, i, r)$	$\mathbf{N}\{l\}$	$\mathbf{N}\{i\}$	$\mathbf{N}\{r\}$
1	0	i	r
0	0	not i or not r	
$\mathrm{sa}(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\}, i, r) + \mathrm{sa}(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\}, i, r)$	1	–	–

The component $\mathrm{ffc}_3(\mathbf{X}_{ch})$ of function (13) is defined as:

$$\mathrm{ffc}_3(\mathbf{X}_{ch}) = \frac{\left(\begin{array}{l}\sum_{i=1}^{n}\sum_{r=1}^{R}\max\left(0, \sum_{k=1}^{K}\mathrm{sa}\left(\mathbf{N}_k^{\mathrm{root}}, i, r\right) - 1\right) + \\ + \sum_{j=1}^{m}\sum_{r=1}^{R}\max\left(0, \sum_{k=1}^{K}\mathrm{sb}\left(n_{j,k}^{B}, j, r\right) - 1\right)\end{array}\right)}{K \cdot \left(2^{d^{\max}-1} + m\right)} \tag{16}$$

where denominator stands for maximum number of leaves and output fuzzy sets for all K fuzzy rules, numerator stands for penalty for using specified fuzzy set more than 1 time by any fuzzy rule, function $\mathrm{sa}(\mathbf{N_k}, i, r)$ stands for number of used input fuzzy set $A_{i,r}$ by kth rule, function $\mathrm{sb}(n^B, j, r)$ stands for number of used output fuzzy set $B_{j,r}$ by k-th rule. The output of function $\mathrm{sa}(\mathbf{N}, i, r)$ from Eq. (16) is calculated according to Table 4 and the function $\mathrm{sb}(n^B, j, r)$ output is calculated as follows:

$$\mathrm{sb}\left(n^B, j, r\right) = \begin{cases} 1 & \text{for} \quad n^B = r \\ 0 & \text{for} \quad n^B \neq r \end{cases}. \tag{17}$$

The component $\mathrm{ffc}_4(\mathbf{X}_{ch})$ of function (13) was defined with assumption that one fuzzy rule cannot use multiple fuzzy sets which are connected to the same input:

$$\mathrm{ffc}_4(\mathbf{X}_{ch}) = \frac{1}{n \cdot K}\left(\sum_{i=1}^{n}\max\left(0, \sum_{k=1}^{K}\mathrm{mul}\left(\mathbf{N}_k^{\mathrm{root}}, i\right) - 1\right)\right), \tag{18}$$

where function $\mathrm{mul}(\mathbf{N}, i)$ stands for penalty for multiple use of fuzzy sets connected to i-th input calculated according to Table 5. The penalty resulting from using OR operator in minimization of fitness function (13) is smaller than penalty for using AND operator. Using the OR operator (see $\mathbf{N}\{o\} = 1$ in Table 5) for the same inputs is acceptable (opposed to AND operator), but it complicates readability of fuzzy rules.

Table 5 Output of the $\mathrm{mul}(\cdot)$ function

$\mathrm{mul}(\mathbf{N}, i)$	$\mathbf{N}\{l\}$	$\mathbf{N}\{i\}$	$\mathbf{N}\{o\}$
$\mathrm{mul}(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\}, i) + \mathrm{mul}(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\}, i)$	0	–	0
$\frac{1}{2}\left(\mathrm{mul}(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\}, i) + \mathrm{mul}(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\}, i)\right)$	0	–	1
0	1	not i	–
1	1	i	–

The aim of fitness function is to minimize values of all fitness function components which allow us to obtain accurate system (ffc_1) with simple structure (ffc_2) and consistent interpretable fuzzy rules (ffc_3 and ffc_4).

2.5 Description of Learning Algorithm

The learning algorithm purpose is to select parameters of the fuzzy sets stored in base (2) and to select the structure of the fuzzy rules. The taken into consideration algorithm designed for proposed system structure and encoding works according to the following steps:

- Step 1. In this step N^{pop} individuals of the population $\mathbf{P}$ are initialized according to description from Sect. 2.3.
- Step 2. This step involves evaluation of the individuals of the population $\mathbf{P}$ by fitness function (13).
- Step 3. In this step N^{pop} of child individuals are generated and stored in the temporary population $\mathbf{P}$. Genes $\mathbf{X}_{ch}^{\text{fsets}}$ of these individuals are initialized with use of genetic algorithm crossover operator. The individuals for crossover are selected by roulette wheel method from the population $\mathbf{P}$. The genes $\mathbf{X}_{ch}^{\text{rules}}$ of these individuals are initialized by choosing randomly genes $n^{B_{j,k}}$, weights w_k^{rule} and root nodes from preselected parents.
- Step 4. This step purpose is to mutate individuals from the population $\mathbf{P}$ (each individual is mutated with probability $p_{m1} \in (0,1)$). Genes $\mathbf{X}_{ch}^{\text{fsets}}$ are mutated (with probability $p_{m2} \in (0,1)$) with use of standard genetic mutation operator. Genes $\mathbf{X}_{ch}^{\text{rules}}$ are mutated (with probability $p_{m3} \in (0,1)$). This mutation is based on random changes of parameters $\mathbf{N}\{i\}$, $\mathbf{N}\{r\}$ and $n_{j,r}^{B}$. Independent mutation probabilities $p_{m1} \neq p_{m2} \neq p_{m3}$ (where $p_{m1} \gg p_{m2} > p_{m3}$) balance the mutation in the following way: (a) mutation should be processed on the greater part of the population $\mathbf{P}'$ (p_{m1}) which provides a proper diversity of the population, (b) from the other hand, genes mutation probability (p_{m2}) cannot be high due to degeneration of the population, (c) changes in connection between leafs and nodes (p_{m3}) should be rarely performed, because too intense changes in relationships between the fuzzy rules and fuzzy sets could hinder the convergence of the algorithm.
- Step 5. Next, the individuals from the population $\mathbf{P}'$ are pruned. This process is based on replacing randomly selected node of each genetic programming tree (with probability $p_x \in (0,1)$) by randomly generated leaf ($\text{init}(\mathbf{N}, 0, 1)$).
- Step 6. In this step extension of genetic programming trees from population $\mathbf{P}'$ is performed. This process is based on replacing randomly selected leaf of each genetic programming tree (with probability $p_l \in (0,1)$) by randomly generated node ($\text{init}(\mathbf{N}, lvl, 0)$). The lvl stands for actual height of leaf, which prevents excessive growth of the tree.

- Step 7. In this step for each individual from the population $\mathbf{P}'$ a new fuzzy rule is added (with probability p_d and only when $K < K^{\max}$) or existing randomly chosen fuzzy rule is removed (with probability p_u and when $K > K^{\min}$).
- Step 8. After modification of individuals from the population $\mathbf{P}'$ (Steps 3–7) each individual is evaluated by fitness function (13).
- Step 9. Next, the individuals from populations $\mathbf{P}$ and $\mathbf{P}'$ are merged and only N^{pop} best individuals are chosen to replace the population $\mathbf{P}$.
- Step 10. In the last step of the algorithm the purpose is to check if stop condition is met (for example if the number of executed iterations of algorithm reaches specified value). If this condition is met, the algorithm stops. Otherwise, algorithm goes back to the Step 3.

2.6 Fuzzy Rules Notation

As it was mentioned earlier, in the proposed system (4) a varied fuzzy operators were used to aggregate antecedences of fuzzy rules and to process the fuzzy sets. Due to this and using genetic programming tree structure, the notation of fuzzy rules is defined as:

$$R_k : \left(\text{IF} \overbrace{\text{zp}\left(\mathbf{X}_{ch}^{\text{rules}}\left\{\mathbf{N}_k^{\text{root}}\right\}\right)}^{\text{definitionbyfunction}} |\mathbf{X}_{ch}^{\text{rules}}\left\{w_k^{\text{rule}}\right\} \text{THEN} \begin{pmatrix} y_1 \text{IS} B_{1,\mathbf{X}_{ch}^{\text{rules}}\left\{n_{1,k}^B\right\}}, \ldots, \\ y_m \text{IS} B_{m,\mathbf{X}_{ch}^{\text{rules}}\left\{n_{m,k}^B\right\}} \end{pmatrix} \right), \quad (19)$$

where function $\text{zp}(\mathbf{N})$ defines antecedences of fuzzy rules according to the Table 6. It is worth to mention that values of weights w_1 and w_2 from Eq. (6) and w^{rule} from Eq. (9) can be replaced by their linguistic equivalents: n (not important) for values lower than 0.25, i (important) for values from the range $[0.25, 0.75]$ and v (very important) for values higher than 0.75. The fuzzy rule notation may be written as the following example:

$$R_1 : \left(\text{IF} \left(\begin{pmatrix} x_4 \text{ISNOT} A_{4,5} | v \\ \text{AND} \\ x_6 \text{IS} A_{6,2} | n \end{pmatrix} | n \text{AND} \begin{pmatrix} x_1 \text{IS} A_{1,4} | i \\ \text{OR} \\ x_2 \text{IS} A_{2,2} | n \end{pmatrix} | i \right) | i \text{THEN} \left(y_1 \text{IS} B_{1,4}\right). \right), \quad (20)$$

or in a shorter form as:

$$R_1 : \text{IF} \left(\begin{pmatrix} (x_4 \text{ISNOT} A_{4,5} | v \text{AND} x_6 \text{IS} A_{6,2} | n) | n \\ \text{AND}(x_1 \text{IS} A_{1,4} | i \text{OR} x_2 \text{IS} A_{2,2} | n) | i \end{pmatrix} | i \right) \text{THEN} \left(y_1 \text{IS} B_{1,4}\right). \quad (21)$$

Table 6 Notation output of the $\mathrm{zp}(\cdot)$ function

$\mathrm{zp}(\mathbf{N}, i)$	$\mathbf{N}\{l\}$	$\mathbf{N}\{o\}$
$(\mathrm{zp}(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\})\|\mathbf{N}\{w_1\}\mathrm{ANDzp}(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\})\|\mathbf{N}\{w_2\})$	0	0
$(\mathrm{zp}(\mathbf{N}\{\mathbf{N}^{\mathrm{L}}\})\|\mathbf{N}\{w_1\}\mathrm{ORzp}(\mathbf{N}\{\mathbf{N}^{\mathrm{P}}\})\|\mathbf{N}\{w_2\})$	0	1
$x_{\mathbf{N}\{i\}}\mathrm{ISA}_{\mathbf{N}\{i\},\mathbf{N}\{r\}}$	1	0
$x_{\mathbf{N}\{i\}}\mathrm{ISNOTA}_{\mathbf{N}\{i\},\mathbf{N}\{r\}}$	1	1

Table 7 Considered simulation problems

Problem	Label	Inputs	Outputs	Rows
Airfoil self-noise	ASN	5	1	1503
Box and Jenkins gas furnace	BJG	6	1	290
chemical plant problem	CPP	3	1	70
concrete slump test	CST	7	3	103
servo data set	SDS	4	1	167

3 Simulation Results

Simulation was performed using the following benchmarks (for details see Table 7): airfoil self-noise problem [3] (ASN), Box Jenkins gas furnace problem [2] (BJG), chemical plant problem [22] (CPP), concrete slump test [23] (CST), servo data set [19] (SDS). The simulations were executed for four cases:

- case 1—case without using weights (in this case all weights values w_1, w_2 and w^{rule} were set to 1).
- case 2—case with using rule weights (in this case weights values w_1, w_2 were set to 1).
- case 3—case with using fuzzy operators weights (in this case weights values w^{rule} were set to 1).
- case 4—case with using rule weights and fuzzy operators weights.

This way of testing allowed precise determination of the impact of using weights in the system (4) on the results.

3.1 Simulation Parameters

Values of parameters of the algorithm were experimentally selected as follows: number of fuzzy sets in fuzzy sets' base (2) for each input and output $R = 5$, minimum number of fuzzy rules $K^{\min} = 3$, maximum number of fuzzy rules $K^{\max} = 5$, maximum height of the tree $lvl^{\max} = 5$, weights of fitness function

components (13) $w_{\text{ffacc}} = 1.0$, $w_{\text{ffcom}} = 0.5$, $w_{\text{ffsam}} = 0.2$, $w_{\text{ffmul}} = 0.1$, number of individuals in population $N^{\text{pop}} = 100$, number of algorithm iterations $Nstep = 1000$, individual mutation probability $p_{m1} = 0.7$, genes mutation probability $p_{m2} = 0.2$, rules mutation probability $p_{m3} = 0.1$, pruning of tree probability $p_x = 0.3$, extending of tree probability $p_l = 0.2$, adding new fuzzy rule probability and removing fuzzy rule probability to $p_u = 0.3$. For each benchmark and case, simulations were repeat 100 times and results were averaged.

3.2 Obtained Results

Obtained results for all simulation problems are presented in Table 9. The normalized and averaged results for all simulation problems are presented in Fig. 5 and in Table 8. The example of obtained fuzzy rules and fuzzy sets are presented in Fig. 6 and in Table 10.

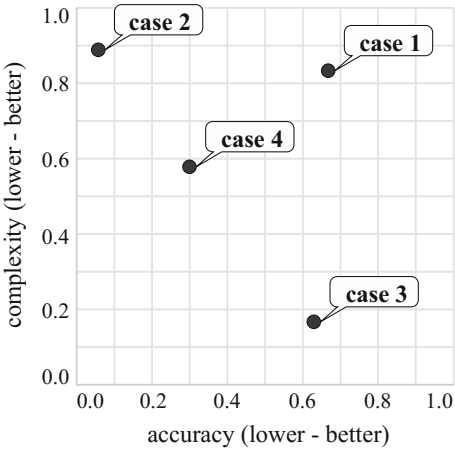

Fig. 5 Obtained accuracy and complexity normalized and averaged for all considered simulation problems

Table 8 Obtained accuracy and complexity, normalized and averaged for all considered simulation problems (see also Fig. 5)

Rule weights	Norm weights	ffacc	ffcom
No	No	0.6675	0.8334
Yes	No	0.0574	0.8888
No	Yes	0.6292	0.1671
Yes	Yes	0.2994	0.5785

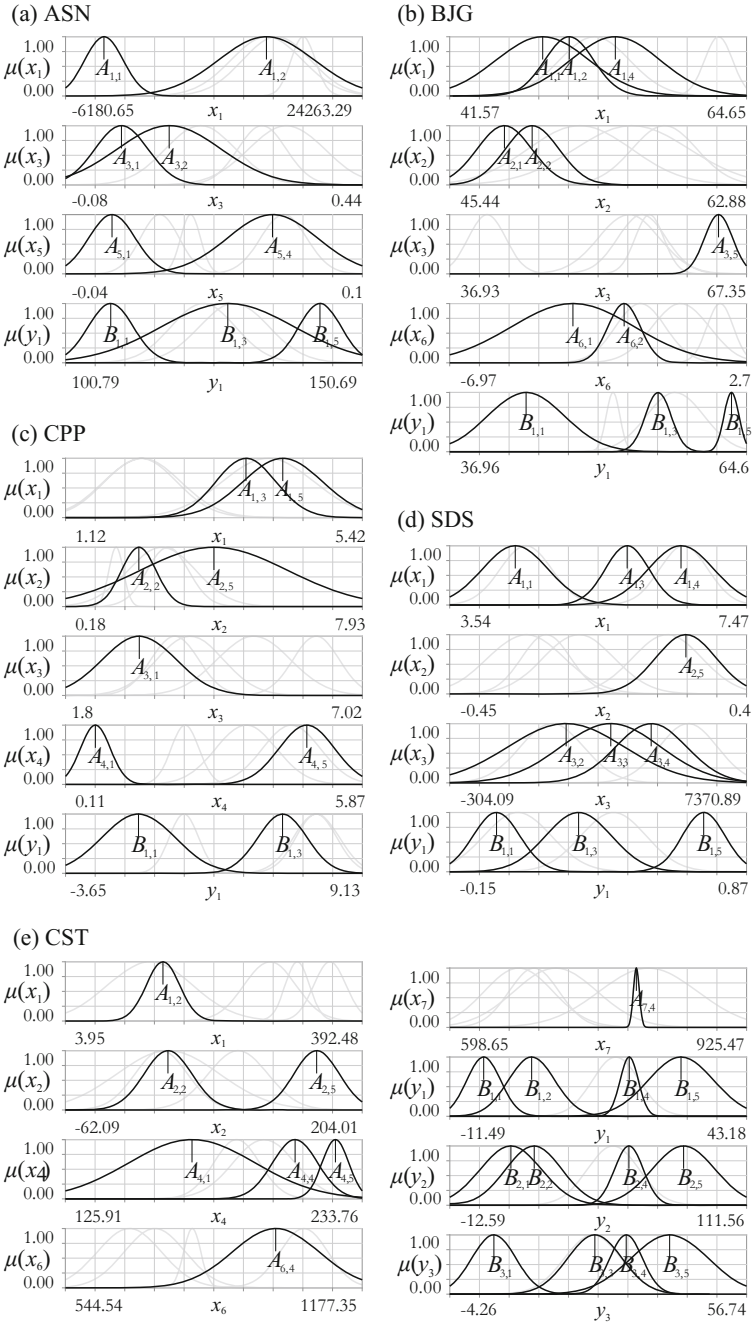

Fig. 6 Fuzzy sets obtained for simulation problems corresponding to fuzzy rules presented in Table 10. Grey fuzzy sets stands for fuzzy sets from fuzzy set base not used by any fuzzy rules

Table 9 Obtained simulation results in comparison with the best results (aimed on accuracy) obtained by other authors [1, 6, 10, 11]

Label	Rule weights	Norm weights	avg.ff	avg.K	avg. RMSE	RMSE	Best RMSE (other authors)
ASN	No	No	0.1601	3.7778	4.4576	4.1256	From 2.4280 to 5.270
	Yes	No	0.1585	4.0000	4.3578	3.9965	
	No	Yes	0.1529	3.5556	4.3882	3.8559	
	Yes	Yes	0.1566	3.6000	4.3459	4.1694	
BJG	No	No	0.0831	4.4500	0.4315	0.3720	From 0.2190 to 0.4490
	Yes	No	0.0854	4.7500	0.4123	0.3633	
	No	Yes	0.0746	3.7000	0.5124	0.4470	
	Yes	Yes	0.0778	4.0588	0.4894	0.4036	
CPP	No	No	0.0676	4.6000	0.0087	0.0065	From 0.0042 to 0.0092
	Yes	No	0.0719	4.5882	0.0082	0.0069	
	No	Yes	0.0631	4.0500	0.0085	0.0071	
	Yes	Yes	0.0770	4.7000	0.0081	0.0065	
CST	No	No	0.1193	3.1500	13.9072	12.8881	From 11.9410 to 15.3440
	Yes	No	0.1193	3.1579	13.7611	12.1798	
	No	Yes	0.1186	3.0588	13.7612	12.6257	
	Yes	Yes	0.1159	3.0588	13.7616	13.3177	
SDS	No	No	0.1016	3.8000	0.4188	0.2976	From 0.1177 to 0.7480
	Yes	No	0.0961	3.5500	0.4007	0.3280	
	No	Yes	0.1098	3.4167	0.5248	0.3858	
	No	No	0.1601	3.7778	4.4576	4.1256	

3.3 Simulation Conclusions

The simulation conclusions are following: (a) using rule weights allowed us to obtain better accuracy and similar complexity in a comparison to case without using weights (see Tables 8, 9 and case 2 on Fig. 5), (b) using fuzzy operators weights allowed us to obtain lower complexity and similar accuracy in a comparison to case without using weights (see Tables 8, 9 and case 3 on Fig. 5), (c) using fuzzy operators weights and fuzzy rules weights allowed us to obtain both the lower complexity and better accuracy in a comparison to case without using weights (see Table 8, 9 and case 4 in Fig. 5), (d) obtained results do not differ from the results of other authors (see Table 9). It is worth to mention that other authors' results are concentrated mostly on accuracy or on using more complex systems (see e.g. [1, 10]), (e) proposed approach is characterized by clear and interpretable fuzzy rules (see Fig. 6 and Table 10).

Table 10 Obtained examples of fuzzy rules for all simulation problems

Label	Fuzzy rules notation	RMSE
ASN	$\begin{cases} R_1 : \text{IF}\left(\begin{matrix} x_1 \text{IS} A_{1,1} \mid v \\ \text{AND}\, x_3 \text{ IS} A_{3,1} \mid n \end{matrix}\right) \mid i \text{ THEN } \left(y_1 \text{IS } B_{1,5}\right) \\ R_2 : \text{IF}\left(x_1 \text{ IS } A_{1,1} \mid v \text{ AND } x_5 \text{ IS } A_{5,1} \mid n\right) \mid v \text{ THEN } \left(y_1 \text{ IS } B_{1,3}\right) \\ R_3 : \text{IF}\left(\left(\begin{matrix} x_5 \text{ IS } A_{5,4} \mid v \\ \text{AND } x_3 \text{ IS NOT } A_{3,2} \mid n \end{matrix}\right) \mid v \text{ OR } x_1 \text{ IS } A_{1,2} \mid n\right) \mid v \text{ THEN } \left(y_1 \text{ IS } B_{1,1}\right) \end{cases}$	4.2719
BJG	$\begin{cases} R_1 : \text{IF}\left(x_6 \text{ IS } A_{6,1} \mid v \text{ AND} \left(\begin{matrix} x_3 \text{ IS } A_{3,5} \mid i \\ \text{OR } x_1 \text{ IS } A_{1,2} \mid n \end{matrix}\right) \mid v\right) \mid i \text{ THEN } \left(y_1 \text{ IS } B_{1,5}\right) \\ R_2 : \text{IF}\left(x_1 \text{ IS } A_{1,1} \mid v \text{ OR } x_2 \text{ IS } A_{2,1} \mid i\right) \mid v \text{ THEN } \left(y_1 \text{ IS } B_{1,1}\right) \\ R_3 : \text{IF}\left(x_1 \text{ IS } A_{1,4} \mid v \text{ OR} x_6 \text{ IS } A_{6,2} \mid i\right) \mid v \text{ THEN } \left(y_1 \text{ IS } B_{1,3}\right) \\ R_4 : \text{IF}\left(x_1 \text{ IS } A_{1,1} \mid v \text{ OR } x_2 \text{ IS } A_{2,2} \mid i\right) \mid v \text{ THEN} \left(y_1 \text{ IS } B_{1,1}\right) \end{cases}$	0.4725
CPP	$\begin{cases} R_1 : \text{IF}\left(x_3 \text{ IS } A_{3,4} \mid i \text{ AND } x_3 \text{ IS } A_{3,3} \mid i\right) \mid v \text{ THEN} \left(y_1 \text{ IS } B_{1,5}\right) \\ R_2 : \text{IF}\left(\left(x_3 \text{ IS } A_{3,2} \mid v \text{ OR } x_1 \text{IS } A_{1,4} \mid i\right) \mid v \text{ OR } x_1 \text{ IS } A_{1,3} \mid i\right) \mid i \text{ THEN} \left(y_1 \text{ IS } B_{1,1}\right) \\ R_3 : \text{IF}\left(\left(x_3 \text{ IS } A_{3,3} \mid i \text{OR } x_3 \text{ IS } A_{3,4} \mid i\right) \mid i \text{OR } x_2 \text{ IS } A_{2,5} \mid n\right) \mid i \text{ THEN} \left(y_1 \text{ IS } B_{1,3}\right) \\ R_4 : \text{IF}\left(x_1 \text{ IS } A_{1,1} \mid i \text{ AND } x_3 \text{ IS } A_{3,3} \mid i\right) \mid v \text{ THEN} \left(y_1 \text{ IS } B_{1,5}\right) \end{cases}$	0.0082
CST	$\begin{cases} R_1 : \text{IF}\left(x_7 \text{ IS } A_{7,4} \mid i \text{ OR } x_2 \text{ IS } A_{2,5} \mid v\right) \mid v \text{ THEN} \left(y_1 \text{ IS } B_{1,1}, y_2 \text{IS } B_{2,2}, y_3 \text{IS} B_{3,4}\right) \\ R_2 : \text{IF}\left(x_1 \text{IS} A_{1,2} \mid i \text{AND} x_4 \text{IS} A_{4,5} \mid v\right) \mid i \text{THEN} \left(y_1 \text{IS} B_{1,4}, y_2 \text{IS} B_{2,4}, y_3 \text{ IS } B_{3,1}\right) \\ R_3 : \text{IF}\left(x_6 \text{ IS } A_{6,4} \mid i \text{ OR } x_4 \text{ IS } A_{4,4} \mid n\right) \mid i \text{ THEN} \left(y_1 \text{ IS } B_{1,2}, y_2 \text{ IS } B_{2,1}, \ y_3 \text{ IS } B_{3,3}\right) \\ R_4 : \text{IF}\left(x_4 \text{ IS NOT } A_{4,1} \mid i \text{ AND } x_2 \text{ IS } A_{2,2} \mid i\right) \mid v \text{ THEN} \left(\begin{matrix} y_1 \text{ IS } B_{1,5}, \ y_2 \text{ IS} \\ B_{2,5}, \ y_3 \text{ IS } B_{3,5} \end{matrix}\right) \end{cases}$	13.4402
SDS	$\begin{cases} R_1 : \text{IF}\left(x_3 \text{ IS NOT } A_{3,1} \mid v \text{ AND } x_1 \text{ IS } A_{1,5} \mid n\right) \mid i \text{ THEN} \left(y_1 \text{ IS } B_{1,1}\right) \\ R_2 : \text{IF} \left(\begin{matrix} x_3 \text{ IS NOT } A_{3,1} \mid v \\ \text{AND} \left(\begin{matrix} x_1 \text{ IS } A_{1,3} \mid v \\ \text{OR} \left(\begin{matrix} x_4 \text{ IS } A_{4,5} \mid i \\ \text{AND } x_2 \text{ IS } A_{2,2} \mid n \end{matrix}\right) \mid v \end{matrix}\right) \mid n \end{matrix}\right) \mid i \text{ THEN} \left(y_1 \text{ IS } B_{1,1}\right) \\ R_3 : \text{IF}\left(x_2 \text{ IS NOT } A_{2,5} \mid i \text{ OR } x_4 \text{IS NOT } A_{4,1} \mid n\right) \mid n \text{ THEN} \left(y_1 \text{ IS } B_{1,3}\right) \end{cases}$	0.4251

The corresponding fuzzy sets are shown in Fig. 6

4 Conclusions

In this paper a weighted fuzzy genetic programming algorithm for selection of the structure and the parameters of the fuzzy systems for nonlinear modelling is presented. In presented approach fuzzy rules take the form of binary trees where nodes of these trees decide on aggregation operators (AND/OR) and the leaves of these trees are connected to the input fuzzy sets. The proposed method allows us to obtain accurate fuzzy systems with clear and interpretable fuzzy rules. The obtained accuracy is similar to the accuracy obtained by other authors, achieved using systems which usually do not take into account interpretability. The use of the system weights shown possibilities in increasing the system accuracy (with use of rule weights), decrease the system complexity (with use of fuzzy operators'

weights) or improve both accuracy and complexity (with use of both weights). The proposed approach was tested on typical nonlinear modelling benchmarks and it can be said that obtained results are satisfying.

Acknowledgment The project was financed by the National Science Centre (Poland) on the basis of the decision number DEC-2012/05/B/ST7/02138.

References

1. Bosnic, Z., Kononenko, I.: Correction of regression predictions using the secondary learning on the sensitivity analysis outputs. Comput. Inform. **20**, 1–17 (2001)
2. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco (1970)
3. Brooks, T.F., Pope, D.S., Marcolini, A.M.: Airfoil self-noise and prediction. Technical report, NASA RP-1218 (1989)
4. Carmona, C.J., Ruiz-Rodado, V., del Jesus, M.J., Weber, A., Grootveld, M., González, P., Elizondo, D.: A fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans. Inf. Sci. **298**, 180–197 (2015)
5. Cheng, S., Shi, Y., Qin, Q., Zhang, Q., Bai, R.: Population diversity maintenance in brain storm optimization algorithm. J. Artif. Intell. Soft Comput. Res. **4**(2), 83–97 (2014)
6. Cpałka, K., Łapa, K., Przybył, A., Zalasiński, M.: A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects. Neurocomputing **135**, 203–217 (2014)
7. Edmonds, A.N., Kershaw, P.S.: Genetic programming of Fuzzy logic production rules with application to financial trading. In: Proceedings of the IEEE World Conference on Computational Intelligence, Orlando, Florida (1994)
8. Gabryel, M., Woźniak, M., Damaševičius, R.: An application of differential evolution to positioning queueing systems. Lect. Notes Comput. Sci. **9120**, 379–390 (2015)
9. Gacto, M.J., Alcalá, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures. Inf. Sci. **181**(20), 4340–4360 (2011)
10. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
11. Łapa, K.: Algorithms for extracting interpretable expert knowledge in nonlinear modeling issues. Ph.D. thesis (in polish), Czestochowa University of Technology (2015)
12. Łapa, K., Cpałka, K., Galushkin, A.I.: A new interpretability criteria for neuro-fuzzy systems for nonlinear classification. Artif. Intell. Soft Comput. Lect. Notes Comput. Sci. **9119**, 448–468 (2015)
13. Łapa, K., Cpałka, K.: On the application of a hybrid genetic-firework algorithm for controllers structure and parameters selection. Adv. Intell. Syst. Comput. **429**, 111–123 (2015)
14. Mendes, R.R.F., Voznika, F.B., Freitas, A.A., Nievola, J.C.: Discovering fuzzy classification rules with genetic programming and co-evolution. In: De Raedt, L., Siebes, A. (eds.) PKDD 2001, LNAI 2168, pp. 314–325 (2001)
15. Robinson, M.R.: Mersenne and Fermat numbers. Proc. Am. Math. Soc. **5**, 842–846 (1954)
16. Motulsky, H.J., Christopoulos, A.: Fitting models to biological data using linear and nonlinear regression. A practical guide to curve fitting. GraphPad Software Inc., San Diego, CA (2003)
17. Nallasamy, K., Ratnavelu, K.: Optimal control for stochastic linear quadratic singular Takagi-Sugeno fuzzy delay system using genetic programming. Appl. Soft Comput. **12**, 2085–2090 (2012)

18. Preen, R.J., Bull, L.: Fuzzy dynamical genetic programming in XCSF. In: GECCO'11, July 12–16, 2011, pp. 167–168
19. Quinlan, J.R.: Learning with continuous classes. In: Adams, A., Sterling, L. (eds.) Proceedings 5th Australian Joint Conference on AI, World Scientific, Singapore (1992)
20. Rutkowski, L.: Computational Intelligence. Springer (2008)
21. Stanimirovic, Z., Maric, M., Bozovic, S., Stanojevic, P.: An efficient evolutionary algorithm for locating long-term care facilities. Inf. Technol. Control **41**(1), 77–89 (2012)
22. Sugeno, M., Yasukawa, T.: A fuzzy-logic based approach to qualitative modelling. IEEE Trans. Fuzzy Syst. **1**, 7–31 (1993)
23. Yeh, I.C.: Modeling slump flow of concrete using second–order regressions and artificial neural networks. Cement Concr. Compos. **29**(6), 474–480 (2007)