ELSEVIER

Journal of Systems Architecture 47 (2001) 649–662

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Learning feed-forward and recurrent fuzzy systems: A genetic approach

Hartmut Surmann [a,*], Michail Maniadakis [b,1]

[a] GMD, German Nation Research Center for Information Technology, Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

[b] Department of Electronic and Computer Engineering, Technical University of Crete, 73100 Chania, Crete, Greece

Abstract

In this paper, we present a new learning method for rule-based feed-forward and recurrent fuzzy systems. Recurrent fuzzy systems have hidden fuzzy variables and can approximate the temporal relation embedded in dynamic processes of unknown order. The learning method is universal i.e., it selects optimal width and position of Gaussian like membership functions and it selects a minimal set of fuzzy rules as well as the structure of the rules. A genetic algorithm (GA) is used to estimate the fuzzy systems which capture low complexity and minimal rule base. Optimization of the "entropy" of a fuzzy rule base leads to a minimal number of rules, of membership functions and of subpremises together with an optimal input/output (I/O) behavior. Most of the resulting fuzzy systems are comparable to systems designed by an expert but offers a better performance. The approach is compared to others by a standard benchmark (a system identification process). Different results for feed-forward and first-order recurrent fuzzy systems with symmetric and non-symmetric membership functions are presented. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy logic controller; Recurrent fuzzy systems; Genetic algorithm; Entropy of fuzzy rule; Machine learning; Dynamic processes

1. Introduction

The use of fuzzy systems for automation tasks is highly increasing within the past decade. Fuzzy logic controller or fuzzy rule-based systems (FRBS) provide a formal method for the representation and approximation of imprecisely known relationships by encoding them in the antecedent and consequent parts of rules. The FRBS models the human decision making process by means of the collection of rules.

In most of the automatically learned fuzzy systems, computational effort is spent in finding parameters e.g., fuzzy sets and linguistic terms that give a desired behavior to the system. On one hand these standard fuzzy systems are simple feed-forward fuzzy systems (F-FRBS) with a one shot input/output (I/O) structure and with no hidden fuzzy variables. Nevertheless, it was shown [1–3] that F-FRBS are universal approximators. A great number of different optimization approaches for F-FRBS are known. The book from Herrera and Verdegay [4] provides a good collection of optimization papers using genetic algorithms (GAs). On the other hand recurrent fuzzy system (R-FRBS) i.e., fuzzy systems with hidden variables are introduced by Bersini and Gorrini in [5]. They

* Corresponding author. Tel.: +49-2241-142518; fax: +49-2241-142342.
E-mail address: surmann@gmd.de (H. Surmann).

[1] Visiting researcher at GMD.

also present a gradient-based optimization method to tune the membership functions (MFs). R-FRBSs are used to model a process of order superior to one i.e.,

$$y(t) = f(x(t-1), \ldots, x(t-k)),$$

where x/y are input/output vectors.

In both types, F-FRBS and R-FRBS, the rules and the fuzzy sets used in these rules play a crucial role for the outcome of the system. So, choosing the right rules and fuzzy sets becomes an important issue. Furthermore, to choose the parameters the human decision making process has also to be considered. Therefore, additional attention has to be given to the structure of the system. Most of today's fuzzy system learning algorithms consider only one of the above aspects. Particularly, attention is payed only to I/O behavior and not to minimization of the fuzzy systems. That is, computational effort is spent in estimating parameter values that give a desired behavior to the system, while no attention is payed to its structure. However, sometimes the parametric adjustment of the selected structure may be too poor. It does not represent the problem good enough or the behavior is satisfactory but the structure is too complex and the dimension of the problem increases unreasonably. Besides the large tuning period and the significant amount of wasted resources, a structure of unreasonable high complexity could also lead to biasing effects such as the well-known over-fitting. It is known [6–8] that a more simple structure leads to a more robust system.

First approaches of structural tuning were based on classification methods applied to the I/O space [9]. Some deterministic [10] and neural-based approaches [11] have also been proposed. Castellano et al. [10] represent the rule base as a tree and propose a method for adding and deleting subtrees based on the number of training samples that activates each rule subpremise. Gorrini et al. [12] use a gradient descent technique for tuning the shape of the MFs and a number of local variables is used for measuring the systems' learning ability. New MFs are added in the areas with the greatest instability.

Except for the widely used gradient-based optimization methods which suffer from the local minimum problem, GAs have been proved to be a new powerful optimization method able to overcome the local minimum problem. In 1989, Karr et al. [13] introduced the use of GAs for fuzzy systems optimization. Since then, some genetic-based approaches for structural and parametrical tuning have been proposed with the aim to design and optimize fuzzy system e.g., [14–16] or to minimize the number of MFs per variable by e.g., [17–19]. A comparative study where genetic parameters effect the learning of FRBS presented in [20]. Glorennec [14] suggests a GA based method for fuzzy systems structural and parametrical tuning. The method produce fuzzy systems with minimal number of rules, but it does not focus on the reduction of membership functions per variable. Wong and Ling [17] try to eliminate MF's whose middle value exceeds the corresponding variable domain during genetic process. A GA based on virus theory of evolution is used in [21]. The rule structure is encoded in chromosomes, while validity bits are used to cancel the less significant parts of the rule base. All of the above methods of parameter optimization and system identification are single step methods. A repetitive method e.g., a method which cycles between parameter optimization and structure optimization can be more reliable for structure identification because further simplification on the resulting structure can be achieved.

A new approach to optimal fuzzy systems in which the influence of the entropy of fuzzy rules is introduced was proposed by Surmann et al. [22,23]. In contrast to neural approaches minimal entropy of the FRBS leads to rule bases with appropriated overlapping MFs. The automatic optimization of the I/O behavior of a fuzzy system leads to very width and much overlapping MFs (e.g., >0.9) so that nearly all fuzzy rules are activated similar to neurons in a neural network. On one hand, these fuzzy systems are relatively robust because one failing fuzzy rule is overlapped by the other rules. On the other hand these fuzzy systems tends to the well known over-fitting, which decreases the approximation quality. The consideration of the entropy during the optimization represents the counterpart which prevents that the

MFs get to width. A compromise between robustness and overfitting is achieved. The resulting fuzzy system is much simpler and can be better understood by the user. Less activated rules leads together with a fast rule processing algorithm to a significant saving of computational resources [23]. After the optimization, the MFs are labeled with linguistic terms according to the order of the middle points e.g., small, normal, big for three MFs so that the linguistic rules are achieved. This rules can be regarded, understand, reused and maintained by users.

In the following, the notion of minimal entropy for a FRBS is extended. Based upon this a repetitive step by step genetic method is proposed for both structural and parametrical tuning at the same time. A supervised learning method is presented. It is able to generate minimal FRBSs with an optimal I/O behavior by using appropriate parameter values. An extension to unsupervised learning method is simply possible.

The paper is organized as follows. In Section 2, a brief introduction to F-FRBSs, R-FRBSs and GAs is presented together with the definition of optimal FRBS. In Section 3, the genetic procedure for finding the FRBSs is given. To illustrate the good basic characteristics of the proposed method the Box and Jenkins data is used as a standard benchmark in Section 4. The paper finishes with a conclusion in Section 5.

2. Basic terms of genetic algorithms and fuzzy rule-based systems

2.1. Fuzzy rule-based systems

Fuzzy systems are based on Fuzzy Set Theory [24]. A number of rules is used to model the way that a system behaves. Fuzzy rules are constructed in the well-known IF–THEN form. When crisp values are given as input for an FRBS, they are converted to degrees of membership in the various linguistic values used by the system. This normalization process of different universe of discourses to $[0, 1]$ allows to compare different input values and to use linguistic concepts like "small", "near" or "approximately". The knowledge representation in terms of understandable linguistic rules differs significantly from the black-box approach used by other methods (e.g., neural networks).

Each set of crisp input values activates a number of rules to some degree. The degree to which rules are activated is calculated by combining the degrees of membership (subpremises) with the fuzzy AND operator. This leads to the total activation of the rule. Supposing that the kth rule of the system is:

$$k : \text{IF } x_1 \text{ is } I_{1,k} \text{ AND } x_2 \text{ is } I_{2,k} \text{ AND } \ldots x_n \text{ is } I_{n,k}, \\ \text{THEN } y \text{ is } O_k,$$

where $x_1, \ldots, x_n$ and y represent input and output variables and $I_{1,k}, \ldots, I_{n,k}, O_k$ their respective MFs. Then, the extent to which a rule is activated is calculated as:

$$\alpha_k = \mu_{I_{1,k}}(x_1) \text{ AND } \mu_{I_{2,k}}(x_2) \text{ AND} \cdots \text{AND} \mu_{I_{n,k}}(x_n),$$

where $\mu_{I_{i,k}}(x_i)$ is the membership value of x_i in the $I_{i,k}$ fuzzy set. The result of the AND combination is used as a measure for the truth of the rule where the AND operation is a T-Norm e.g., minimum. When a rule is activated with a truth value α_k, the inferencing process states that the MF of the output set is:

$$\mu^{\text{inf}}_{O_k}(y) = \alpha_k \text{ AND } \mu_{O_k}(y),$$

which is the fuzzy result of the rule. Here, the minimum operator min is used as an AND operator for both cases.

The total output fuzzy set is calculated by the compositional rule of inference as:

$$\mu^{\text{total}}_{O}(y) = \mu^{\text{inf}}_{O_1}(y) OR \mu^{\text{inf}}_{O_2}(y) OR \cdots OR \mu^{\text{inf}}_{O_K}(y),$$

where it is supposed that the system is described by K-rules. In order to get a crisp number as output, usually a defuzzification method is used. The calculation of the fuzzy result function $\mu^{\text{total}}_{O}(y)$ and the final crisp value is the bottleneck during computation. Therefore, a modified center of gravity algorithm is used to calculate the area A_i and the center of area M_i of each MF before runtime [25]: $A_i = \int O_i(y)\,\mathrm{d}y$, $M_i = \int y \times \mu_{O_i}(y)\,\mathrm{d}y$. The output value is computed as

$$y_{\text{crisp}} = \frac{\sum_{i=0}^{K} \alpha_i \times M_i}{\sum_{i=0}^{K} \alpha_i \times A_i}. \tag{1}$$

Definition 1 (*Fuzzy Rule-Based System*). A fuzzy rule-based system, $\text{FRBS} = (LV, R, T, I, AGR\text{–}DE)$. It is a 5-Tupel with a set LV of n linguistic input variables and m output variables, a set R of k fuzzy rules of T-Norm T, for the combination of rules subpremises an Implication strategy I verifying $I(a,0) = 0$ if $a \neq 0$ (e.g., an R-implication or t-norm), and an aggregation and a defuzzification strategy AGR–DE. This can be a T-CO-Norm T^*, for the combination of rules output sets, and defuzzyfication strategy DE (e.g., center of gravity, maximum or FCOG) or the addition/division of Eq. (1).

Theorem 1 (Universal approximator). *Let FRBS be the set of all FS and $f : U \subseteq IR^n \rightarrow IR$ be a continuous function defined on a compact U. For each $\epsilon > 0$ there exists an $FS_\epsilon \in FRBS$ such that*

$$\sup\{|f(\vec{x}) - FS_\epsilon(\vec{x})| \quad |\vec{x} \in U\} \leqslant \epsilon.$$

Wang [1] or Buckley [2] provide the proof. The universal approximator theorem is a very good theoretical results but unfortunately it is not constructive, i.e., it cannot be used to get an FRBS with a predefined error limit ϵ. Usually in real world, application ϵ is unknown and influence, e.g., by sensor noise so that it does not make sense to fix it. Later in this paper will give a hint how to find appropriate error limits.

2.2. Recurrent fuzzy rule-based systems

Starting from recurrent neural networks, Gorrini and Bersini introduced in 1994, R-FRBS together with a gradient-based learning algorithm for adapting the membership functions to model high order dynamic processes [5,26]. Independently from this approach, Surmann et al. [27] used R-FRBS to model behaviors and the activation of the behaviors of an autonomous mobile robot. Other recurrent structures are achieved with recurrent fuzzy neural networks [28,29]. This neural approaches cannot find a minimal rule base and/or did not generate understandable fuzzy rules.

A σ-order R-FRBS is characterized by rules in which one or more state variables s appear both in premise and consequent parts, like:

$$\begin{aligned} k &: \text{IF } s(t) \text{ is } I_{0,k} \text{ AND } x_1 \text{ is } I_{1,k} \text{ AND } \ldots x_n \text{ is } I_{n,k}, \\ &\text{THEN } s(t+1) \text{ is } O_{0,k} \text{ AND } y \text{ is } O_{1,k}, \end{aligned} \tag{2}$$

where $s(t)$ represents the state of the system at time t, $x_1, \ldots, x_n$ and y represent input and output variables and $I_{0,k}, \ldots, I_{n,k}$, $O_{0,k}, O_{1,k}$ their respective MFs. R-FRBSs are used to model a process of order superior to one i.e.,

$$\begin{aligned} y(t) = f(&y(t-1), y(t-2), \ldots, y(t-\sigma), x_1(t-1) \\ &, \ldots, x_n(t-1), x_1(t-2) \ldots, x_n(t-\sigma)). \end{aligned}$$

Dynamic processes of order superior to one i.e., map their output not only to the current input but also to previous inputs. The processes are more difficult to approximate and to control than first-order processes. One possibility is to know either the exact order or an over estimation of it, and then to settle the input of the fuzzy approximator as a temporal window containing all inputs at all required time steps. Since mostly the exact order is unknown the recurrent structure is used where some variables appear both in the input and output parts. This variables are used as internal variables and build a short-term-memory (Fig. 1). If the same input $\vec{x}$ is presented to a recurrent R-FRBS, the output y can be different depending on

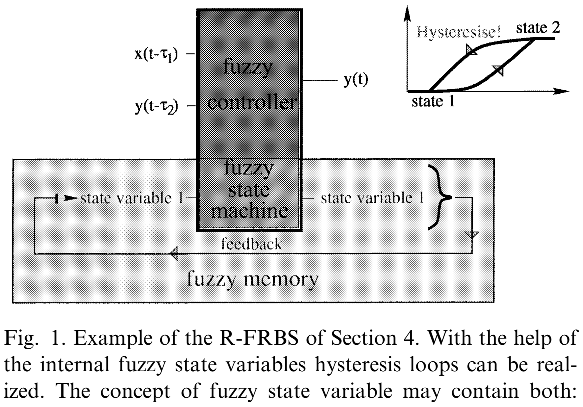


Fig. 1. Example of the R-FRBS of Section 4. With the help of the internal fuzzy state variables hysteresis loops can be realized. The concept of fuzzy state variable may contain both: crisp and fuzzy membership functions. A more complex example can be found in [30].

the current state s of the internal variables. Hysteresis loops can be modeled by the use of hidden fuzzy variables (Fig. 1). The different curves are selected according to the state of the hidden variables. For the controlling of behavior-based autonomous mobile robots [27], the hidden fuzzy variables are used to select the different behaviors and to model their temporal activation. The internal fuzzy variables have their own temporal dynamics given by the recurrent fuzzy rules. Note that the fuzzy approach is a generalization of the crisp one, so that crisp state variables can also be modeled by fuzzy state (hidden) variables.

2.3. Costs and minimal FRBS

The challenge is to find an automatic learning algorithm which learns and optimizes the structure of the fuzzy rule base, the input, output and hidden variables as well as the MFs of the R-FRBS. Therefore a definition of the complexity and with it of the costs of a R-FRBS is necessary:

Definition 2 (*Complexity Cost Function, E*). An FRBS with k fuzzy rules, n linguistic variables, $k * n$ subpremises and $m = m_1 + \cdots + m_n$ MFs is less complex than one with

(1) $k + 1$ fuzzy rules, $k * n$ subpremises and m MFs or

(2) k fuzzy rules, $(k * n) + 1$ subpremises and m MFs or

(3) k fuzzy rules, $k * n$ subpremises and $m + 1$ MFs.

The weighting of the above three terms depends on the application. A mathematical expression of the complexity function E is given later. On the basis of the cost function E a two-step learning procedure is described. First, an "opening step" is any operation on a FRBS that increases the cost function E. Second, a "closing step" is any operation on an FRBS that decreases the cost function E. Any opening step results in a new fuzzy system with more rules, MFs or subpremises. Whereas any closing step results in a new fuzzy system with less rules, MFs or subpremises. Finally, we are interested in minimal fuzzy systems and we define them by using the cost function E:

Definition 3 (*Minimal Fuzzy-System*). Let us approximate a real system $f(\vec{x})$, using a fuzzy system $X(\vec{x})$ and let $\epsilon > 0$ be a maximal accepted error limit. A fuzzy system X with $\sup\{|f(\vec{x}) - X(\vec{x})| \; |\vec{x} \in U\} \leqslant \epsilon$ is called minimal if there is no other fuzzy system Y with $\sup\{|f(\vec{x}) - Y(\vec{x})| \; |\vec{x} \in U\} \leqslant \epsilon$ and $E(Y) < E(X)$, where E is the complexity cost function.

Alternatively to the absolute metric, the Euclidian metric can be used: $(\sqrt{\sum (f(\vec{x}) - Y(\vec{x}))^2} \leqslant \epsilon)$. The real system $f(\vec{x})$ is represented by referential vectors $\vec{r}$.

2.4. Genetic algorithms

Traditional optimization methods are based on the fact that certain functions are differentiable. Unfortunately, in many real world problems such functions cannot be defined. But even if they can, gradient search methods may not find global optimal solutions. A possible way to overcome such problems is to use GAs. Generally, a GA consists of a problem, a number of encoded solutions for that problem, some genetic operators which produce new solutions and a fitness function which says how good a particular solution for the problem is seen in Fig. 2.

Usually the *fitness function* describes the aggregation of some desired properties for the solutions and is not necessarily differentiable. Each solution is *encoded* as a *chromosome* by binary or real values. A *population* consists of a number of

```
Genetic Algorithm
begin (1)
  t = 1
  Initialize Population(t)
  Evaluate fitness Population(t)
  While (Generations < Total Number) do
  begin (2)
    select Population(t + 1) out of Population(t)
    Apply Crossover on Population(t + 1)
    Apply Mutation on Population(t + 1)
    Evaluate fitness Population(t + 1)
    t = t + 1
  end (2)
end (1)
```

Fig. 2. The structure of standard GA.

individuals represented by chromosomes. A population at a certain time step is a *generation*. Genetic operators are applied to each generation to produce the next generation. Common genetic operators are *selection*, *crossover* and *mutation*.

During selection, individuals with high fitness values within the current population are selected to build the basis for the new generation. Crossover is a way of creating new solutions by randomly selecting two chromosomes of previous solutions from the gene pool and exchanging portions of their strings. Mutation is performed upon a selected chromosome by randomly changing a part of its coded value. Mutation is needed to ensure diversity in the population.

Definition 4 (*Genetic algorithm*). Let $B_0 = \{A_1, \ldots, A_M\} \in P(B)$ be a population from the set of all possible populations $P(B)$, created by the binary string $A_i = \{0,1\}^r$. M is the size of the population, $OF : P(B) \to R^+$ the Fitness function, $\Gamma : P(B) \to P(B)$ the Crossover function, $\Phi : P(B) \to P(B)$ the Mutation function, $\Theta : P(B) \to P(B)$ the Selection Strategy and $t : R^+ \to \{0,1\}$ a Termination function. Then, this 7-Tupel $\mathrm{GA} = (B_0, M, \Omega, \Gamma, \Phi, \Theta, t)$ is called Genetic Algorithm.

3. Optimizing FRBSs

Up to now a lot of work has been done in combining fuzzy systems and genetic algorithms [6]. Fuzzy–genetic combinations can be classified in two categories. On one hand fuzzy techniques are used to improve GA behavior [18] and to model GA components [31]. On the other GAs are used to optimize the structure of the fuzzy system and the I/O behavior [18]. In the following, attention will be paid to the second category and especially to achieving an optimal structure of a fuzzy system by means of GAs.

3.1. Membership function shape and rule base construction

Membership functions will be constructed from probability density functions by $\mu(x) = \lambda p(x)$. The constant λ is calculated using the constraint $\sup\mu = 1$. Here the Gaussian probability distribution $p(x) = \varphi(x; m, \sigma^2)$ is chosen, because this distribution fits a lot of real world problems. Therefore, the membership functions used in the system are defined by

$$\mu(x) = \sigma\sqrt{2\pi}\varphi(x; m, \sigma^2) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right). \tag{3}$$

The MFs are approximated by six straight lines and they are limited by $\mu(x) = 0$ for $x \notin [m - 2.9\sigma, m + 2.9\sigma]$. The line approximation speeds up the evaluation process of the FRBS by a factor two [25]. Start and end points of these six lines are the roots of the second derivative of the normalized Gaussian MF ($m = 0, \sigma^2 = 1$). Special margin MFs (smallest and biggest) are defined at the left and right border of a linguistic variable (Fig. 3(b)):

$$\mu_{\text{left}}(x) = \begin{cases} 1, & x < 0, \\ \mathrm{e}^{-(1/2)x^2}, & x \geqslant 0, \end{cases}$$
$$\mu_{\text{right}}(x) = \begin{cases} 1, & x \geqslant 0, \\ \mathrm{e}^{-(1/2)x^2}, & x < 0. \end{cases} \tag{4}$$

Non-symmetric Gaussian MFs consists of two parts (left and right) with a common middle point but different σ_{left} and σ_{right} for the left and right side (Fig. 3(a)). They are used to achieve more flexible systems.

3.2. Using genetic algorithms

When the optimization process is initialized, it is desirable to produce a generation that has enough parameters to gain flexibility. For that reason, at the beginning of the optimization process the structure of the fuzzy system is opened. This means that for each multiply used MF in the rule base, copies of the function for each rule to which it participates are produced (i.e., $q_i = k$, $i = 1, \ldots, n+m$). Thus each rule has its own MFs and each variable has k MFs. All copies are equal at this point of the process, but they are allowed to vary independently in the next steps. Now, the GA has the possibility to eliminate the extra

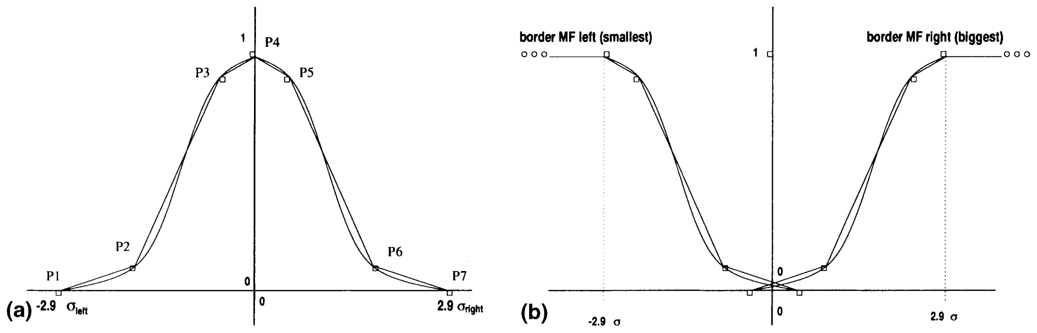


Fig. 3. Linearized model of Gaussian like MFs. Six lines are used to describe them (a). The smallest and biggest MF (ordered by the middle points) are linearized by three lines (b).

parameters while running. Therefore, we have a matrix structure with $n+m$ columns (input + output variables) and k rows (number of rules). Each row matches a fuzzy rule with constant length and can be simply coded in the chromosomes.

The chromosomes of the GA are built by two or three genes (parameter sets). Two genes are selected if the MF is symmetric, three if it is non-symmetric (middle gene, right sigma gene, left sigma gene). The first gene consists of the middle point of the MF, while the second one consists of the respective sigmas or σ_{right} and σ_{left}. Each middle point and sigma is converted from a real number to a fixed binary number of $l = 8$ or $l = 16$ bits accuracy using the *gray encoding* scheme. Gray-coding makes genetic operators more robust against single bit changes: The value of the gray-coded gene is only 1 higher or lower, if one bit of the encoding is changed. On the contrary several bits of a binary coded number may change if it increases or decreases by 1, e.g., 4 bits if 7 increases to 8. Let A3,A2,A1,A0 a four bit string. Then the decoding into an integer is [20]:

- For the binary coding:
  Result $= A3 \times 2^3 + A2\times 2^2 + A1 \times 2^1 + A0$.
- For the gray coding:
  $B3 = (A3 + A2 + A1 + A0) modulo 2$,
  $B2 = (A2 + A1 + A0) modulo 2$,
  $B1 = (A1 + A0) modulo 2$,
  $B0 = A0 modulo 2$,
  Result $= B3 \times 2^3 + B2 \times 2^2 + B1 \times 2^1 + B0$.

Every chromosome S contains all fuzzy set parameters m_{ij} and σ_{ij}, where $i = 1, \ldots, n+m$ is the number of variables and $j = 1, \ldots, q_n$ is the number of respective MFs per variable.

During fitness evaluation the reverse process (decoding) is done to get back the middle points and sigmas as reals. With this real values the MFs of the fuzzy variables are constructed and the fitness function is evaluated. The length of a string is: $L = 2 \times l \times (n+m) \times k$ for the symmetric and $L = 3 \times l \times (n+m) \times k$ for the non-symmetric case. An additional index field for each variable is English: to hold the similarity information of the MFs.

3.2.1. Fitness function

The definition of a suitable fitness function is basic for the genetic process. It expresses in a formal way the desired properties of solutions. Here we define a fitness function with two main items. The first part considers the I/O behavior and the second part the structure and complexity of the resulting fuzzy system. The structural part of the fitness function has three items.

$$OF = OF^{\mathrm{I/O}} \times (OF^{E} \times OF^{S} \times OF^{UZ}), \tag{5}$$

where *OF* is the fitness function that the GA has to maximize.

The only property for the I/O behavior of the FRBS is the maximization of the reciprocal of the mean square error (MSE).

$$OF^{\mathrm{I/O}}_{pv} = pv \Bigg/ \sum_{i=1}^{pv} \sum_{j=1}^{n_{\text{out}}} (r_{i,j} - o_{i,j})^2, \tag{6}$$

where pv is the number of referential data pairs, n_{out} is the dimension of the output vector, $o_{i,j}$ is current computed output value of the FRBS and $r_{i,j}$ is the respective referential value ($i = 1,\dots,pv$, $j = 1,\dots,n_{\text{out}}$). For other application fields this part of the fitness function should be adopted to the application class.

For control processes the system response parameter e.g., rise time RT, over-shoot OV, steady-state error SSE, settling time ST, etc. can be used. For the well-known pole balancing problem [32] a suitable I/O part of the fitness function is:

$$\begin{aligned} OF^{\text{I/O}} &= (1 + k_1 * f_{\text{RT}}) \times (1 + k_2 * f_{\text{OV}}) \\ &\quad \times (1 + k_3 * f_{\text{SSE}}) \times (1 + k_4 * f_{\text{ST}}), \end{aligned} \tag{7}$$

where

$$f_{\text{RT}} = \mathrm{e}^{-(\text{RT}/\text{RT}_{\text{expected}})},\ f_{\text{OV}} = \mathrm{e}^{-(\text{OV}/\text{OV}_{\text{expected}})},$$
$$f_{\text{SSE}} = \mathrm{e}^{-(\text{SSE}/\text{SSE}_{\text{expected}})},\ f_{\text{ST}} = \mathrm{e}^{-(\text{ST}/\text{ST}_{\text{expected}})}.$$

The constants $k_1,\dots,k_4$ are used to weight the corresponding system parameter.

Three criteria for the structure of the FRBS are also considered. The first criterion (OF^E) for the structure of a FRBS is the *degree of fuzziness* or the *entropy* of a FRBS and was introduced in [23]. It is defined by the average number of activated rules:

$$R_\phi = \frac{1}{pv} * \sum_{i=1}^{pv} R_{\text{cur},i}, \tag{8}$$

where $R_{\text{cur},i}$ is the current number of activated rules for the ith I/O pair and pv the number of those pairs. If the number of activated rules is minimal i.e., one, then an FRBS is maximally understandable by humans. FRBS with a high number of activated rules behave like a neural network i.e., a lot of rules (neurons) determine the output values. To decrease the entropy of an FRBS and with it the overlap of the MFs, a maximal number $R_{\max}$ of activated fuzzy rules has to be defined and the fitness function is extended as:

$$OF^{\text{E}} = \begin{cases} \frac{1}{((R_{\text{act}}/R_{\max})-1)a+\left((R_\phi/R_{\max})-1\right)b+1}, & R_{\max} < R_{\text{act}}, \\ \frac{1}{((R_{\text{act}}/R_{\max})-1)b+1}, & \text{else}, \end{cases} \tag{9}$$

where a and b are predefined weighting factors. The influence of the entropy part of the fitness function for the benchmark of Section 4 is shown in Figs. 4 and 5.

The second criterion (OF^{S}) for the structure of the FRBS is the number of MFs. For that MFs with high degrees of overlap are counted. Such nearly equal MF pairs are desirable, because they can be easily unified to only one MF which reduced the cost function (Definition 2(3)) The criterion for the highest possible number of similar MFs is:

$$OF^{\text{S}} = \left(\frac{s_{\text{no}}}{(n+m)R_{\text{total}}}\right)\gamma + 1, \tag{10}$$

where s_{no} is the number of similar MFs (see below), $n+m$ is the total number of input and output variables, R_{total} is the total number of rules and $\gamma \in \mathbf{R}$ is a predefined weighting factor for the number of similar MFs. Fig. 6 shows an example of a similar and non-similar MFs.

Let us suppose we have two MFs denoted by A and B which are described by seven points $(P_i(x_{Ai}, y_{Ai}), P_i(x_{Bi}, y_{Bi}))$ because a linearized model of Gaussian MFs is used (Fig. 3(a)). The average width is defined by

$$w = \frac{|x_{A2} - x_{A6}| + |x_{B2} - x_{B6}|}{2}. \tag{11}$$

Two MFs are similar if the distance of their second, third, fifth and sixth points is small enough compared to the average width of the MFs. This similarity criterion is used for symmetric and non-symmetric cases:

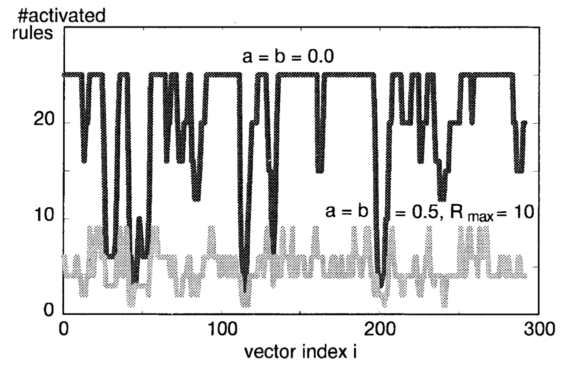


Fig. 4. Activated rules with and without entropy consideration.

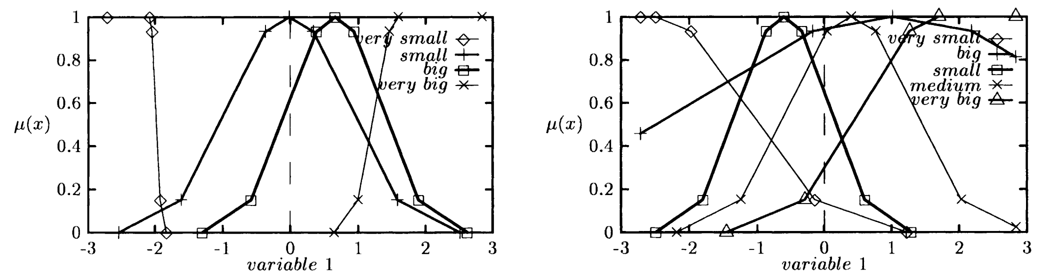


Fig. 5. Membership functions with and without entropy consideration. The entropy reduction deletes one MF and limits the width distinctly.

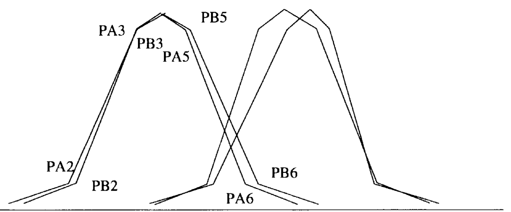


Fig. 6. Example of similar (left) and non-similar (right) MFs. The right MFs have non-symmetric widths.

$$s_{\text{no}} = \sum_{i}^{n+m} \sum_{j}^{q_i} \sum_{j<k}^{q_i} S_{i,j,k}, \text{ with}$$

$$s_{i,j,k} = \begin{cases} 1, & \frac{w}{c} > |x_{ij_2} - x_{ik_2}| \text{ and} \\ & \frac{w}{d} > |x_{ij_3} - x_{ik_3}| \text{ and} \\ & \frac{w}{c} > |x_{ij_6} - x_{ik_6}| \text{ and} \\ & \frac{w}{d} > |x_{ij_5} - x_{ik_5}|, \\ 0, & \text{else,} \end{cases} \tag{12}$$

where $n + m$ is the number of variables and q_i the number of membership functions of the variable i. The constant factors $c = 10$, and $d = 16$ have been selected by simulation experiences. The use of linear approximated Gaussian like MFs is a little more complicated and increases the number of parameters but it speeds up the evaluation of the fuzzy controller [25] (each controller is one individual) which is more important to speed up the total genetic process.

The third criterion (OF^{UZ}) for the structure of the FRBS deals with *never activated* ($\mu(x) = 0\ \forall x \in U$) and *always completely activated* ($\mu(x) = 1\ \forall x \in U$) MFs. On one hand, a subpremise of a rule can be eliminated if the belonging MF is always completely activated for all of the training pairs so that the cost function E decrease (Definition 2(2)). On the other hand, MFs which are never activated can be eliminated together with all the rules that they participated to so that the cost function E also decrease (Definition 2(1)). The criterion for the highest possible number of always zero and always one MFs with the predefined weighting factors $\zeta, \eta \in \mathbf{R}$ is:

$$OF^{UZ} = \left(\frac{u_{\text{no}}}{(n+m)R_{\text{total}}}\right)\zeta + \left(\frac{z_{\text{no}}}{(n+m)R_{\text{total}}}\right)\eta + 1, \tag{13}$$

where u_{no} is the number of always one MFs, z_{no} is the number of always zero MFs. Always one MFs are replaced by trapezes covering the total variable domain, while zero MFs are replaced by impulses. A collection of MFs with zero, one and similar MFs is shown in Fig. 7.

3.2.2. Producing new generations

After estimating the fitness value of each chromosome in the current population, genetic opera-

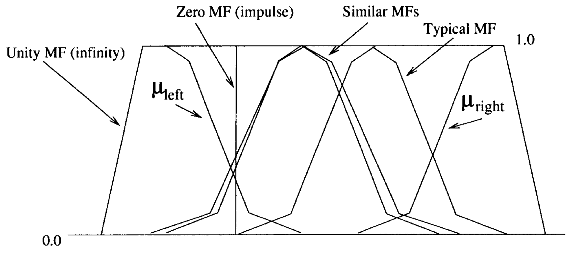


Fig. 7. Different kinds of membership functions.

tors are applied to produce the new generation. Two new operators are introduced: *Set zero/one* is used to produce MFs which are always one or always zero. It randomly selects an MF for each I/O variable and changes its sigma value to infinity ($+\infty$) in case of a always one MF or to zero in case of a always zero MF. The other genetic operator called *set similar*, selects randomly for each I/O variable a membership function and makes it equal to the MF that is most similar to it (Eq. (12)).

A mate pool is built by selecting a predefined percentage of the best individuals from the current population (truncation selection). Each chromosome of the new population is produced by randomly selecting its two parents from the mate pool. The crossover operator is applied to them to produce two children. Crossover is done for each gene of the two chromosomes and not for the total chromosome (Fig. 8). Mutation, set similar and set zero/one operators are randomly applied to the selected children as described above. The last two operators are used only in case of FRBSs with MSE less than the predefined upper limit.

3.2.3. *Fuzzy rule based system minimization*

To achieve a faster convergence only the reciprocal of the MSE is used as an optimization criterion in the first steps of the GA: $OF = OF^{\mathrm{I/O}}$ (Eq. (5)). When the GA produces FRBSs with MSE less than a predefined threshold ϵ, the optimization criterion is changed to the global fitness function OF of Eq. (5). The above is according to the definition of minimal fuzzy systems of a predefined error limit (ϵ of Definition 3). To find an appropriated threshold $\epsilon_{\mathrm{thres}}$ a first run of the GA is done with $\epsilon = 0$ that means the GA only tries to optimize the I/O behavior without optimizing the structure. In the next run of the GA an error limit $\epsilon_{\mathrm{thres}} = 1.1, \ldots, 1.5 \times OF_{\epsilon=0}$ can be set.

After *opening* the structure of the rule base at the beginning of the GA it now has to be *closed* back. Rules with a zero MF ($\sigma = 0$) are deleted, one MFs ($\sigma = +\infty$) are eliminated from the rules and similar MFs ($\sigma_i \approx \sigma_j$ and $m_i \approx m_j, i \neq j$) are unified by the genetic operators. The unification of similar MFs results in a new MF with the average of the given middle points as the new middle point and the maximum of the given sigmas as the new sigma. Simulations showed that 50–100 generations are enough for the GA to stabilize the FRBS structure. After the FRBS structure is stable a closing step is done. Then the whole process repeats with this new structure. Fig. 9 shows the affect of a closing step to the structure of a rule base.

If no more closure steps can be applied, the most compact structure of the FRBS has been found. Nearly optimal middle points and sigmas for the MF have been estimated and the GA terminates. It is well known that the estimation of an exact global optimum is very time consuming with GAs. For that reason an easy local optimization method is used after the GA to "climb the"

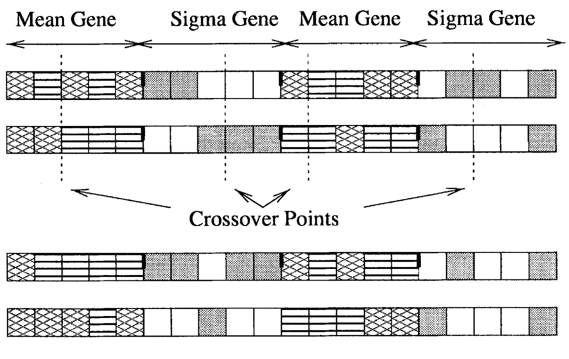
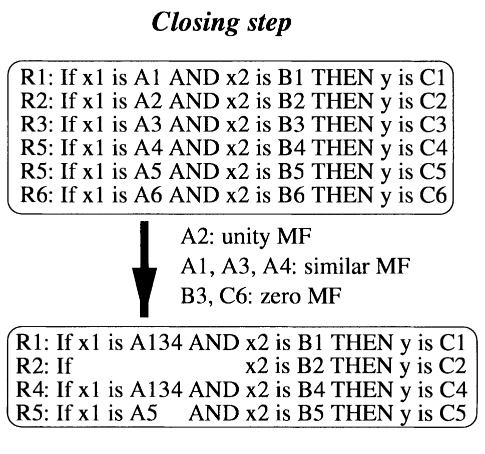


Fig. 8. Crossover is applied between genes of chromosomes.


Closing step

R1: If x1 is A1 AND x2 is B1 THEN y is C1
R2: If x1 is A2 AND x2 is B2 THEN y is C2
R3: If x1 is A3 AND x2 is B3 THEN y is C3
R5: If x1 is A4 AND x2 is B4 THEN y is C4
R5: If x1 is A5 AND x2 is B5 THEN y is C5
R6: If x1 is A6 AND x2 is B6 THEN y is C6

A2: unity MF
A1, A3, A4: similar MF
B3, C6: zero MF

R1: If x1 is A134 AND x2 is B1 THEN y is C1
R2: If x2 is B2 THEN y is C2
R4: If x1 is A134 AND x2 is B4 THEN y is C4
R5: If x1 is A5 AND x2 is B5 THEN y is C5

Fig. 9. Example of a closing step. After the closing step two rules and one subpremise are deleted. Three MFs are combined.

remaining of the hill. It starts by selecting the first encoded parameter value and a small increment or decrement is applied to it. The process is controlled by the fact that only small variations of the parameters are allowed and that after a change the total fitness of the fuzzy system is evaluated to insure increasing fitness. If that change results in an FRBS with a better performance, a new small change is done in the same direction until those changes do not increase the fitness function any more. This procedure is done for all coded parameters. The process terminates if no better performance can be gained. The idea behind the after optimization is that the GA put the population in the correct area where local hill climbing put it on the top. After the optimization, the MFs of the best fuzzy system are automatically labeled according to the numerical order of the middle points and the number of MFs e.g., small, normal, big for three MFs. Therefore, the resulting rule base is readable be the user.

4. Application – gas furnace data

The design algorithm will be illustrated by means of a system identification example. A number of simulations is performed on the data set of Box and Jenkins' *gas furnace data* which is a common benchmark. A collection of representative results is given. The task is to build a rule base model from the referential data set which identifies the process. The data set consists of 296 pairs of I/O observations. The input is the gas flow rate into the furnace and the output is the concentration of CO_2 in the exhausted gas.

As well as in literature, two input variables $n_{in} = 2 : x(t - \tau_1), y(t - \tau_2)$ and output variable $y(t)$ are chosen. Here, $x(t - \tau_1)$ denotes the input at time $t - \tau_1$ and $y(t - \tau_2)$ the output of the process at $t - \tau_2$. With $\tau_1 = 4$ and $\tau_2 = 1$ the referential data set contains $p = 296 - \tau_1 = 292$ elements. The above data set is used for training and as validation set for testing.

For the initialization of the FRBS four MFs for each variable are used together with a rule base of 16 rules. So, after the FRBS is opened a system of 16 rules and 16 MFs per variable is selected. Different strategies are applied with symmetric and non-symmetric MFs for feed-forward and recurrent fuzzy systems. Thousand time steps are used for the GA. A closure step takes place 60 generations after the error limit has been reached. After a closure step a new genetic process is started to continue the optimization. Fig. 10 shows the referential curve and an optimization result.

Representative results are shown in the Tables 1–4. The number of always one MFs is equal to the number of eliminated subpremises, the number of impulses is equal to the number of eliminated rules and the final number of MFs is reduced by number of similar MFs. On one hand symmetric and non-symmetric MFs show a similar I/O behavior (feed-forward case), but non-symmetric MFs reduce the overlap of the MFs which result to a lower number of activated rules. On the other hand R-FRBS with one hidden variable improved the I/O behavior of fuzzy systems. Here, the non-symmetric MFs show the best results. It is worth to mention that none of the previous approaches in literature focused on the elimination of rules or MFs. Compared to the results from literature in Table 5 with our proposed method it is possible to achieve much simpler FRBSs and also the best I/O behavior of all approaches. Considering the fuzzy rule base structure in the fitness function minimize the number of rules, MFs and subpremises. The interesting point is that the approximation quality increases while using structural information in the fitness functions. The reason is that over-fitting is avoided.

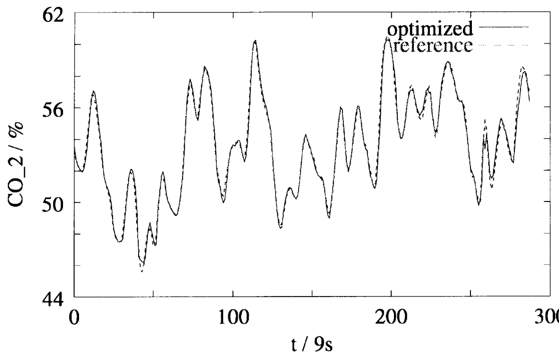


Fig. 10. The gas furnace data curve of our optimized approach and the reference from literature.

Table 1
Results for gas furnace data obtained from FRBS structures with *symmetric* MFs

MSE	ϵ_{tres}	q_i	u_{no}	k	R_ϕ	$a, b\gamma, \zeta, \eta$
Feed-forward symmetric						
0.116	0.15	4-7-7	2	8	5.19	0.10, 0.10, 0.40, 0.20, 0.30
0.121	0.16	4-5-6	1	7	4.73	0.10, 0.20, 0.50, 0.20, 0.30
0.120	0.15	4-6-6	2	8	4.49	0.20, 0.20, 0.15, 0.15, 0.15

The columns shows the MSE Eq. (6), the predefined error limit ϵ_{thres}, the number of MFs q_i per variable $i = 1, \ldots, n+m$ how many always one MFs exist u_{no} Eq. (13), the number of rules k, R_ϕ Eq. (8) the average number of activated rules and the weighting factors of the fitness function (a, b Eq. (9), γ Eq. (10), ζ, η Eq. (13)).

Table 2
Results for gas furnace data obtained from FRBS structures with *non-symmetric* MFs

MSE	ϵ_{tres}	q_i	u_{no}	k	R_ϕ	$a, b\gamma, \zeta, \eta$
Feed-forward non-symmetric						
0.117	0.15	8-8-6	0	8	3.34	0.1, 0.2, 0.2, 0,1, 0.2
0.119	0.16	6-7-5	0	8	3.31	0.2, 0.3, 0.3, 0.2, 0.1
0.121	0.16	7-7-7	2	8	3.62	0.1, 0.2, 0.3, 0.4, 0.2

The columns shows the MSE Eq. (6), the predefined error limit ϵ_{thres}, the number of MFs q_i per variable $i = 1, \ldots, n+m$ how many always one MFs exist u_{no} Eq. (13), the number of rules k, R_ϕ Eq. (8) the average number of activated rules and the weighting factors of the fitness function (a, b Eq. (9), γ Eq. (10), ζ, η Eq. (13)).

Table 3
Results for gas furnace data obtained from *recurrent* FRBS structures with *symmetric* MFs

MSE	ϵ_{tres}	q_i	u_{no}	k	R_ϕ	$a, b\gamma, \zeta, \eta$
Recurrent symmetric						
0.100	0.15	7-6-6-8-8	2	10	3.82	0.10, 0.20, 0.01, 0.06, 0.06
0.115	0.16	7-7-4-6-6	5	10	4.23	0.15, 0.10, 0.05, 0.08, 0.09
0.118	0.16	4-6-3-6-5	3	7	4.40	0.10, 0.10, 0.10, 0.10, 0.20

The columns shows the MSE Eq. (6), the predefined error limit ϵ_{thres}, the number of MFs q_i per variable $i = 1, \ldots, n+m$ how many always one MFs exist u_{no} Eq. (13), the number of rules k, R_ϕ Eq. (8) the average number of activated rules and the weighting factors of the fitness function (a, b Eq. (9), γ Eq. (10), ζ, η Eq. (13)).

Table 4
Results for gas furnace data obtained from *recurrent* FRBS structures with *non-symmetric* MFs

MSE	ϵ_{tres}	q_i	u_{no}	k	R_ϕ	$a, b\gamma, \zeta, \eta$
Recurrent non-symmetric						
0.078	0.15	7-6-5-8-8	4	11	5.70	0.10, 0.15, 0.09, 0.09, 0.20
0.100	0.15	5-8-4-6-4	8	10	5.06	0.10, 0.20, 0.25, 0.10, 0.15
0.109	0.17	3-6-4-5-4	0	8	4.03	0.15, 0.15, 0.20, 0.10, 0.30

The columns shows the MSE Eq. (6), the predefined error limit ϵ_{thres}, the number of MFs q_i per variable $i = 1, \ldots, n+m$ how many always one MFs exist u_{no} Eq. (13), the number of rules k, R_ϕ Eq. (8) the average number of activated rules and the weighting factors of the fitness function (a, b Eq. (9), γ Eq. (10), ζ, η Eq. (13)).

5. Conclusion

The paper presented a learning algorithm for feed-forward and recurrent fuzzy rule based knowledge representations. Both structural and parametrical tuning of fuzzy systems are based of: (1) a complexity cost function and (2) a minimal fuzzy system. In contrast to other known ap-

Table 5
Results for the same problem from literature

MSE	Init. rules	Rules	Author
Literature. Symmetric FRBS			
0.469	7×6	19	Tong [33]
0.355	5×5	6	Sugeno et al. [34]
0.320	9×9	81	Pedryz [35]
0.328	5×5	25	Xu-Lu [36]
0.172	7×7	38	Abreu et al. [9]
0.138	3×5	15	Surmann [3]

The columns show the MSE, the structure of the initial rule base, the final number of rules and the author.

proaches where only parametrical tuning takes place, here optimization of the entropy and the complexity of the fuzzy rule base leads to a minimal number of rules, of MFs and of subpremises together with an optimal I/O behavior. The MFs are labeled with linguistic terms according to the order of the middle points so that the linguistic rules are achieved. This rules can be regarded, understand, reused and maintained by users. The proposed method learns also recurrent fuzzy systems with hidden fuzzy variables. The recurrent fuzzy systems approximate the temporal relation embedded in dynamic processes of unknown order. In contrast to neural approaches the resulting fuzzy system is comparable to systems designed by an expert but with a better I/O behavior. The most characteristic features of the proposed method were illustrated by means of a numerical example.

In future more attention should be given to structural tuning because it highly impacts the capabilities of a system. This has several reasons. Simpler structures i.e., structures with less parameters but fulfilling the approximation requirements lead to systems that:

- can be tuned more easily (automatic or by hand);
- are more understandable and maintainable by users;
- can be better combined with other fuzzy rules, e.g., with rules designed by users while no data is available for an automatic design;
- are more robust;
- have a better approximation quality while avoiding the over-fitting.

Acknowledgements

Special thanks to Claudia Hirsch and the unknown reviewers for hints and corrections.

References

[1] L.-X. Wang, Fuzzy systems are universal approximators, in: Proceedings of the First IEEE International Conference on Fuzzy Systems, San Diego 8–12 March,1992, pp. 1163–1170.

[2] J. Buckley, Sugeno type controllers are universal fuzzy controller, Fuzzy Sets and Systems 53 (1993) 299–304.

[3] H. Surmann, Automatischer Entwurf von Fuzzy Systemen, Dissertation, Universität Dortmund, Fakultät für Elektrotechnik, VDI-Verlag Düsseldorf, 1995 (distinguished with the AKI-Dissertationspreis 1995).

[4] F. Herrera, J. Verdegay (Eds.), in: Studies in Fuzzines and Soft Computing: Genetic Algorithms and Soft Computing vol. 8, Physica-Verlag, Wurzburg, 1996.

[5] V. Gorrini, H. Bersini, Recurrent fuzzy systems, in: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, 1994, pp. 193–198.

[6] O. Cordon, F. Herrera, M. Lozano, On the bidirectional integration of genetic algorithms and fuzzy logic, http://www.bioele.nuee.nagoyau.ac. jp/wec2/papers/p003.html, 1996.

[7] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modeling and control, IEEE Transaction Systen Man and Cyberentic (1985) 116–132.

[8] S.-T. Song, L.-H. Sheen, Heuristic fuzzy-neuro network and its application to reactive navigation of a mobile robot, Fuzzy Sets and Systems 110 (3) (2000) 331–340.

[9] A. Abreu, L. Custodio, C. Pinto-Ferreira, A fuzzy modelling: a rule based approach, in: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 96, 1996, pp. 162–168.

[10] G. Castellano, G. Attolico, A. Distante, Automatic generation of fuzzy rules for reactive robot controllers, Robotics and Autonomous Systems 22 (1997) 133–149.

[11] M. Lee, S. Lee, C. Park, Neuro-fuzzy identifiers and controllers, Journal of Intelligent and Fuzzy Systems 2 (1994) 1–14.

[12] V. Gorrini, T. Salome, H. Bersini, Self-structuring fuzzy systems for function approximation, in: Proceedings of the FUZZ-IEEE / IFES'95, 1995.

[13] C.L. Karr, L.M. Freeman, D.L. Meredith, Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm, SPIE Intelligent Control and Adaptive Systems 1196 (1989) 274–288.

[14] P. Glorennec, Constrained Optimization of FIS using an Evolutionary method, vol. 8 of Studies in Fuzziness and Soft Computing, ch. Optimization of Fuzzy Controllers, Physica-Verlag, Wurzburg, Sep. 1996, pp. 349–368.

[15] A. Gonzlez, F. Herrera, Multi-stage genetic fuzzy systems based on the iterative rule learning approach, Mathware and Soft Computing (1997) 233–249.
[16] F. Hoffsmann, Soft computing techniques for the design of mobile robot behaviors, Information Sciences 122 (2–4) (2000) 241–258.
[17] C.-C. Wong, N.-S. Lin, Rule extraction for fuzzy modeling, Fuzzy Sets and Systems 88 (1997) 23–30.
[18] F. Herrera, M. Lozano, in: Adaptation of Genetic Algorithm parameters Based on Fuzzy Logic Controllers, in: Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, Wurzburg, September 1996, pp. 95–128 (Ch. Optimization of Fuzzy Controllers).
[19] H.B. Gürocak, A genetic-algorithm-based method for tuning fuzzy logic controllers, Fuzzy Sets and Systems 108 (1) (1999) 39–47.
[20] P. Siarry, F. Guely, A genetic algorithm for optimizing takagi-sugeno fuzzy rule bases, Fuzzy Sets and Systems 99 (1998) 37–47.
[21] K. Shimojima, N. Kubota, T. Fukuda, in: Virus Evolutionary Genetic Algorithm for Fuzzy Controller Optimization, in: Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, Wurzburg, September 1996, pp. 369–388, (Ch. Optimization of Fuzzy Controllers).
[22] H. Surmann, A. Kanstein, K. Goser, Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems, in: Proceedings of the First European Congress on Fuzzy and Intelligent Technologies, EUFIT'93, Aachen, 7–10 September 1993, pp. 1097–1104.
[23] H. Surmann, in: Genetic Optimizing of Fuzzy Rule-Based Systems, in: Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, Wurzburg, September 1996, pp. 389–402 (Ch. Optimization of Fuzzy Controllers).
[24] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[25] H. Surmann, A.P. Ungering, Fuzzy-rule-based systems on general purpose processors, IEEE MICRO, Fuzzy Systems (special issue), August 1995, pp. 40–48.
[26] H. Bersini, V. Gorrini, Mlp, rbf, flc: what's the difference, in: Proceedings EUFIT'94, 1994, pp. 19–26.
[27] H. Surmann, J. Huser, L. Peters, A fuzzy system for indoor mobile robot navigation, in: Proceedings of the Fourth IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 95, Yokohama, Japan, 20–24 March 1995, pp. 83–88 (distinguished with Robot Intelligence Award).
[28] C.-F. Juang, C.-T. Lin, Recurrent self-organizing neural fuzzy inference network, IEEE Transactions on Neural Networks (1999) 828–845.
[29] Y. Yoshidaa, An identification algorithm in fuzzy relational systems, Computers and Mathematics with Applications 37 (1999) 87–93.
[30] H. Surmann, L. Peters, in: MORIA – A Robot with Fuzzy Controlled Behaviour, in: Studies in Fuzziness and Soft Computing, vol. 61, Springer, Berlin, 2000, pp. 343–365 (Ch. Layer Integration).
[31] W. Pedryz, M. Reformat, in: Genetic Optimizing of Fuzzy Rule-Based Systems, in: Studies in Fuzziness and Soft Computing, vol. 8, Physica-Verlag, Wurzburg, September 1996, pp. 51–67 (Ch. Optimization of Fuzzy Controllers).
[32] A.G. Barto, R. Sutton, C. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man, and Cybernetics SMC-13 (1983) 834–846.
[33] R.M. Tong, The evaluation of fuzzy models derived from experimental data, Fuzzy Sets and Systems 4 (1980) 1–12.
[34] M. Sugeno, T. Yasukawa, Linguistic modeling based on numerical data, in: Proceedings of IFSA'91, Brüssel, 1991.
[35] W. Pedrycz, An identification algorithm in fuzzy relational systems, Fuzzy Sets and Systems 13 (1984) 153–167.
[36] C.-W. Xu, Y.-Z. Lu, Fuzzy model identification and self-learning for dynamic systems, IEEE Transactions on Systems, Man, and Cybernetics SMC-17 (1987) 683–689.

Hartmut Surmann received his diploma in computer science from the Department of Computer Science, University of Dortmund and his Ph.D. in electrical engineering from the Faculty of Electrical Engineering, University of Dortmund, Germany in 1989 and 1994, respectively. He is a senior researcher at GMD Autonomous intelligent System Institute in St. Augustin, Germany. His primary research interests are robotics and computational intelligence. He has published several papers in the area of design, application and hardware for robotics and fuzzy logic controllers. He is a member of the GI and VDE. Dr. Surmann received the FUZZ-IEEE/IFES'95 Robot Intelligence Award for a fuzzy system for indoor mobile robot navigation and the Ph.D. award for his thesis from the German AI institute in 1996.

Michail Maniadakis received his diploma in computer science from the Dept. of Electronic and Computer Engineering, Technical University of Crete, Greece, 1999. From 1997–1998 he was research student at GMD in St. Augustin, Germany funded by the Leonardo da Vinci program. His primary research interests are robotics and computational intelligence. Currently, he serves the military service.