Automatic Extraction of the Fuzzy Control System for Industrial Processes

Jérôme Mendes, Ricardo Seco, and Rui Araújo
Institute of Systems and Robotics (ISR-UC), and
Department of Electrical and Computer Engineering (DEEC-UC),
University of Coimbra, Pólo II, PT-3030-290 Coimbra
jermendes@isr.uc.pt, rui@isr.uc.pt

Abstract

The paper proposes a new method to automatically extract all fuzzy parameters of a Fuzzy Logic Controller (FLC) in order to control nonlinear industrial processes. The learning of the FLC is performed from controller input/output data and by a hierarchical genetic algorithm (HGA). The algorithm is composed by a five level structure, where the first level is responsible for the selection of an adequate set of input variables. The second level considers the encoding of the membership functions. The individual rules are defined on the third level. The set of rules are obtained on the fourth level, and finally, the fifth level, selects the elements of the previous levels, as well as, the $t$-norm operator, inference engine and defuzzifier methods which constitute the FLC. To demonstrate and validate the effectiveness of the proposed algorithm, it is applied to control a simulated water tank level process.

1 Introduction

Several studies have shown fuzzy control to be an appropriate method for the control of complex nonlinear processes, many of which cannot be easily modeled in a mathematical way. This is because, unlike a conventional process controller, often no rigorous mathematical model is required to design a good fuzzy controller and, in many cases, they can also be implemented more easily [8]. Recently, fuzzy control has been used for a wide variety of industrial systems and consumer products, attracting the attention of many researchers. Fuzzy logic controllers (FLCs) are rule-based systems which are useful in the context of complex ill-defined processes, especially those which can be controlled by a skilled human operator without mathematical knowledge of their underlying dynamics [6]. FLCs are based on a set of fuzzy control rules that make use of people’s common sense and experience. However, there still exist many difficulties in designing FLCs to solve certain complex nonlinear problems. In general, it is not easy to determine and select the most suitable fuzzy rules and membership functions to control the output of a plant, when the only available knowledge concerning the process is the empirical information transmitted by the human operator. Thus, a possible candidate to meet this challenge is the appliance of the Genetic Algorithm (GA) approach, to data extracted from a given process, while it is being manually controlled by a human operator.

GAs are inspired by the process of natural evolution, being a powerful tool to solve optimization problems. These algorithms are applied in many automatic control problems, such as the development and tuning of fuzzy controllers. For that matter, they have been previously employed to select and tune an adequate set of membership functions and fuzzy rules.

In [2], a hybrid architecture is presented, which combines Type-1 or Type-2 fuzzy logic system (FLS) and GAs for the optimization of the FLS membership function parameters, in order to solve the output regulation problem of a servomechanism with nonlinear backlash. In [3], a trajectory tracking genetic fuzzy logic controller was designed for research reactors. Membership functions and action weights of the fuzzy controller were optimally determined by GAs. In [7], a genetic fuzzy logic control methodology is used to develop two production control architectures: genetic distributed fuzzy (GDF) and genetic supervisory fuzzy (GSF) controllers. The GA is used to tune the input variable membership functions for the GSF and GDF controllers. However, in all these methods, the model of the process to be controlled is always needed by the GA to calculate the correct fitness function values. In some problems, this type of method can prove to be of great difficulty, since the knowledge of some complex industrial processes is very hard to obtain. Another drawback of the cited methods is that they only optimize membership function parameters (and action weights in [3]) considering the other components of the fuzzy system fixed, such as implication, aggregation and defuzzifier methods. Other common limitation is the selection of the correct set of input variables. This process is usually manual and not accompanied with the accurate selection of the right time delays, leading to low-accuracy results. A variable with the correct delay can contain more information about the output, than one which does not consider any delay [11].

In [5], a hierarchical evolutionary approach is introduced to optimize the parameters of Takagi–Sugeno fuzzy
systems, where the selection of the variables is performed completely from human knowledge, namely the knowledge of the real model. The problems addressed were function approximation and pattern classification. Moreover, as an improvement of [5], [4] was proposed, where the pre-selection of variables by auxiliary criteria is added. The proposed method was addressed for soft sensor applications, being its design carried out in two steps. First, the input variables of the fuzzy model are pre-selected from the secondary variables of a dynamical process by means of correlation coefficients, Kohonen maps and Lipschitz quotients. Second, a hierarchical GA proposed in [5] is used to identify the fuzzy model itself. However, the input variable selection approach proposed by [4] has some drawbacks. On one hand, the selection of the number of neurons in the Kohonen maps is not automatically performed. On the other hand, variable and delay selection are not jointly performed with the learning of the fuzzy model, which precludes the global optimization of the prediction setting.

The method proposed in this paper is based on the application of the HGA suggested by [4], although, applied to controller design. The main advances and differences contemplated in this work are the improvement of the whole hierarchical structure, automatically extracting the fuzzy rules, as well as, all other elements of the FLS. First, a new hierarchical level, responsible for the selection of an adequate set of input variables and respective time delays was added; then the T-S fuzzy model approach was replaced by a standard fuzzy control system, i.e., the consequent part of the rules is represented by a membership function, rather than the T-S model proposed by [4]; finally, the introduction of many types of antecedent aggregation methods, inference engines and defuzzifier methods were implemented with the aim of providing the GA with a larger search space, thus obtaining the best possible controller for a given dataset. The hierarchical genetic fuzzy system is constituted by five levels. The first level represents the population of the set of input variables and respective time delays. The population of the second level is given by all fuzzy system membership functions, i.e. the antecedent and consequent membership functions that constitute the fuzzy rules. The individual rules population is defined on the third level and the rules set population is obtained on the fourth level. Finally, the fifth level represents the population with the indexes of the selected elements of the previous levels, as well as, the antecedent aggregation method, inference engine and defuzzifier method that constitute the fuzzy controller. To optimize the proposed method, the initial populations of the HGA are obtained by an initialization algorithm proposed by [1]. The initialization method has the main goal of providing a good initial solution for membership functions and rule base populations, facilitating the GA’s tuning and reducing the total computation time.

Moreover, in order to validate and demonstrate the performance and effectiveness of the proposed algorithm, the level of a water tank will be controlled by simulation. To do so, a standard fuzzy controller is developed and applied to a water tank plant, obtaining the HGA learning dataset. Then, the proposed HGA is applied off-line with the aim of achieving a fuzzy controller with a similar response to the projected one. Once the fuzzy controller parameters are determined, the controller is applied to the same water tank plant mentioned before, considering different references, and comparing the results with the ones previously obtained using the standard FLC.

The paper is organized as follows. Section 2 introduces the fuzzy logic system. Section 3 presents an initialization method used to generate the HGA initial populations. The proposed HGA is described in Section 4. The HGA application and respective results are presented and analyzed in Section 5. Finally, remarks and conclusions are made in Section 6.

2 Fuzzy System

This section briefly overviews the main concepts of fuzzy control systems [12]. A fuzzy system is a knowledge-based system defined by a group of **IF-THEN** rules, which can be used to implement fuzzy controllers. The following example illustrates such a rule:

\[ \text{IF the speed of a car is high,} \]
\[ \text{THEN apply less force to the accelerator,} \]  
where **speed** and **force** are input and output variables, respectively. These variables are characterized by the fuzzy sets A, through a mapping defined by \( \mu_A(x) = U \rightarrow [0,1] \), for which, **high** and **less**, are referred as semantic terms.

Fuzzy systems are constituted by a group of four main elements: knowledge base, fuzzifier, fuzzy inference engine and defuzzifier, as shown in Fig. 1.

The knowledge base is composed by a set of \( N \) fuzzy IF-THEN rules \( R_j \) of the form

\[ R_j : \text{IF } x_1 \text{ is } A_{1j}, \text{ and } \ldots \text{ and } x_n \text{ is } A_{nj} \text{ THEN } u \text{ is } B_j, \]  
where \( j = 1, 2, \ldots, N; \) \( x_i \) \( (i = 1, 2, \ldots, n) \) are input variables of the fuzzy system and \( u \) is the output.

The knowledge-base is one of the most important components of a fuzzy system, since all other components rely on it. The fuzzy rules are composed by two parts: the antecedent (IF part) and the consequent (THEN part), where \( A_{ij} \) and \( B_j \) are the linguistic terms characterized by the fuzzy membership functions \( \mu_{A_{ij}}(x) \) and \( \mu_{B_j}(u) \), respectively. The most commonly used types are the trapezoidal, triangular and gaussian membership functions.
The fuzzifier is the fuzzy system element responsible for mapping the real values of the input linguistic variables, \( x_i \), into corresponding fuzzy sets described by membership functions \( X \). In this paper, the only utilized fuzzifier is the singleton fuzzifier [12].

The next element, the fuzzy inference engine (FIE), uses the collection of fuzzy IF–THEN rules to map the input fuzzy set \( X \) into the rule consequent fuzzy sets \( B_j \). The collection of fuzzy outputs of the rules are then combined into an overall inferred fuzzy output \( U \). In the FIE, the first step is to process each rule individually. To process the antecedent part of the rule, using propositions connected by the fuzzy AND operator, \( T \)-norms are employed, while to process propositions connected by the fuzzy OR operator, \( S \)-norms are implemented. In this paper, only \( T \)-norms will be used.

After calculating the antecedent value, fuzzy propositions (antecedent and consequent) are interpreted as fuzzy relations using an implication operator. In fuzzy logic, the sentences IF \( A \) THEN \( B \) can be modeled/written as \( A \rightarrow B \), where \( A \) and \( B \) are the fuzzy propositions whose values are fuzzy sets, and \( \rightarrow \) is the fuzzy implication operator.

The next step in the inference process is to aggregate the outputs of all fuzzy rules with an aggregation method. In this process, the output fuzzy sets of the rules are combined into an overall output fuzzy set \( U \), which is the fuzzy implication operator. There are several choices for the defuzzifier to be associated with the fuzzy inference engine [12].

3 Initialization Method

In order to design a FLC, it is necessary to determine every parameter of the fuzzy system, i.e., membership functions, antecedents and consequents, universe limits and fuzzy rules (Sec. 2). These parameters can be derived directly from expert knowledge of the plant to be controlled, or by using numerical techniques with input/output data of controlled processes. Since expert knowledge is difficult to extract in complex industrial processes, one of the main goals of this work is to develop a controller that does not require any prior expert knowledge to determine these parameters. For that matter, a numerical initialization method, based on a set of input/output data of a process under control (which can be manually controlled) will be applied.

The following initialization algorithm was based on an initialization method proposed in [1], and will be used to generate part of the HGA initial populations, such as the set of rules and their respective membership functions. GAs are usually initialized with random population elements. This sort of approach increases the tuning/search difficulty of the GA, since a set of totally random populations can lead to a very exhausting optimality search, requiring more iterations to attain convergence. Therefore, the appliance of an initialization method is of great importance, in order to reduce the computational cost and increase the algorithm’s performance. However, it should be noted that the aim of this initialization method is not to obtain an optimal solution, but to achieve a satisfactory starting point from which the GA can subsequently commence its coevolutionary tuning. The method is described in Algorithm 1, where it should be noted that its aim is not to obtain an optimal solution, but to achieve a satisfactory starting point from which the GA can subsequently commence its coevolutionary tuning.

4 Hierarchical Genetic Fuzzy System

The approach presented in this paper has the main goal of coevolving the parameters of a fuzzy system, in order to control a plant, merely having/using information of a set of input/output data. The proposed coevolutionary model is shown in Fig. 2 and is constituted by a hierarchical structure, defined by five populations, where each population illustrates different species. The first level represents the population of the set of input variables and their respective time delays. The population of the second level is
given by all fuzzy system membership functions, i.e. the antecedent and consequent membership functions which constitute the fuzzy rules. The individual rule population is defined on the third level and the rules set population is obtained on the fourth level. Finally, the fifth level, represents the population with the indexes of the selected elements of the previous levels, as well as, the antecedent aggregation method, inference engine and defuzzifier that constitute the fuzzy controller.

4.1 Hierarchical Structure

Level I is formed by the population of the set of input variables, and delays, which will be used to design the fuzzy controller. Its chromosome is represented by binary encoding, where each allele (element of the chromosome located in a specific position) corresponds to each input variable or delay (see Fig. 2). The length of the chromosome is given by the total number of system variables and respective delays that are considered as possible candidates to be used as inputs for the fuzzy system. In the example presented in Fig. 2 the selected inputs are \( x_1 = e(t - 1) \), \( x_2 = d(t - 1) \) and \( x_3 = d(t - 2) \).

Level II contains every membership function defined in the universe of the variables involved. The chromosome is represented by integer and real encoding and is formed by the aggregations of all partition sets associated with each input and output variable. An example of its structure can be seen in detail in Fig. 3, where each first allele uses integer encoding to represent the type of membership function. In this paper, three different types of possible membership function are permitted: trapezoidal \((S_k = 1)\), triangular \((S_k = 2)\) and Gaussian \((S_k = 3)\). Alleles 2-5 use real encoding to represent the parameters of the membership function. Considering the 5th membership function, for trapezoidal functions, alleles 2-5 are converted into absolute values, given by (see Fig. 3):

\[
\begin{align*}
  m_{1k} &= m_{2,k-1} + C_{1k}, \\
  m_{2k} &= m_{1k} + C_{2k}, \\
  b_{1k} &= m_{1k} - L_k, \\
  b_{2k} &= m_{2k} + R_k.
\end{align*}
\]

For triangular membership functions, the center is found by the average between \( m_{1k} \) and \( m_{2k} \) (see Fig. 3). For Gaussian functions, the central value is calculated the same way as in the triangular case, and the dispersion is given by \( \sigma_{kj} = \Delta_k / 3 \) where \( \Delta_k = (L_k + R_k) / 2 \). As can be seen in Fig. 2, Level II contains all membership functions parameters. In this example, the second level presents the 5th membership function of \( x_1 \), which is a
trapezoidal membership function \((S_5 = 1)\).

Level III is formed by a population of individual rules. The length of the chromosome is determined by the number of input variables selected by Level I, plus an additional allele that characterizes the output variable. The chromosome is represented by integer encoding, where each allele contains the index of the corresponding antecedent and consequent membership function. Null values indicate the absence of membership function and are only considered for antecedent indexes. In the example presented in Figure 2, the third level contains the composition of the 16th individual rule, where it can be seen that \(x_1\) is represented by the 5th membership function, \(x_2\) is given by the 1st membership function, \(x_3\) is not used and \(u\) is given by the 2nd membership function.

The population of Level IV is constituted by a set of fuzzy rules, where each allele contains the index of the corresponding individual rule that has been included in the set. The chromosome is represented by integer encoding, where once again, null values indicate that the matching allele does not contribute to the inclusion of any rule in the set of fuzzy rules. The length of the chromosome is determined by the maximum number of fuzzy rules. In Figure 2, Level IV represents the 27th set of fuzzy rules which contains the 9th, 16th, and 2nd individual rules specified in Level III.

Finally, each individual of Level V represents a fuzzy system, i.e., all the necessary information to develop the fuzzy controller is contemplated on this level. The chromosome is represented by integer encoding and is constituted by seven alleles. The first allele represents the \(t\)-norm operator. For this matter, the GA selects from between three types of \(t\)-norms: (1) product, (2) minimum and (3) bounded difference (other aggregation operators can be used, see more in [12]). The second allele indicates the index, \(k\), of the set of rules specified on Level IV. The third allele contains the \(q\)th partition set given by Level II. The fourth allele represents the index, \(l\), of the set of input variables selected from Level I. The fifth allele illustrates the type of implication operator used. For this study the implemented implication methods where the (1) Mamdani product and the (2) Mamdani minimum [12]. The sixth allele indicates the type of aggregation operator, where the following operators have been used as possibilities: (1) Maximum, (2) Bounded sum, and (3) Normalized sum. Finally, the seventh allele is responsible for determining the type of defuzzifier. The considered defuzzifiers are: (1) center of gravity (COG), (2) first of maximum (FOM), (3) last of maximum (LOM) and (4) mean of maximum (MeOM). All these operators \((t\)-norm, implication, aggregation and defuzzifier) can be consulted in [9]. In the example of Figure 2, the \(i\)th fuzzy system of Level V uses the product \(t\)-norm operator as the antecedent aggregation method, the 27th set of fuzzy rules of Level IV, the 3rd partition set of Level II, the 23rd set of selected input variables and delays, product implication, maximum aggregation and center of gravity defuzzification. The 27th set of fuzzy rules contains the 9th, 16th, and 2nd individual rules, where the 16th individual rule is composed by two input variables with membership functions 5 and 1, respectively, and one output variable corresponding to membership function 2, i.e.:

\[
R_{16} : \quad \text{IF } x_1(t) \text{ is } 5 \text{ and } x_2(t) \text{ is } 1 \quad \text{THEN } u(t) = 2.
\]

The linguistic term “5” is defined in the 3rd chromosome of Level II, and the input variables \(x_1\), \(x_2\), and \(x_3\) are determined on Level I.

The main steps of the GA algorithm used to learn/improve the fuzzy controller parameters are presented in Algorithm 2. The fitness functions of each in-

<table>
<thead>
<tr>
<th>Algorithm 2 Proposed algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set Generation = 1;</td>
</tr>
<tr>
<td>2. Initialize populations of all levels - Initialize populations of Levels II, III and IV with the initialization method described in Sec. 3; Initialize Levels I and V randomly except the first individual. The first individual of Level I is given by the input variables chosen in Algorithm 1 and the first individual of Level V is initialized with the product (t)-norm, product implication, maximum aggregation and center of gravity defuzzification;</td>
</tr>
<tr>
<td>3. Compute the fitness of each individual, from Level V to Level I:</td>
</tr>
<tr>
<td>(a) Level V - Fuzzy system: the fitness function is (f_{FS}(i) = 1/\text{MSE}(i)), where (\text{MSE}(i)) is the mean square error of the (i)th fuzzy system. The MSE is given by: (\text{MSE}(i) = \frac{1}{T} \sum_{t=1}^{T} (u_t - \hat{u}_t)^2) where (\hat{u}_t) the control output and (u_t) is the target output;</td>
</tr>
<tr>
<td>(b) Level IV - Rule base: the fitness function is (f_{RB}(k) = \max(f_{FS}(b), \ldots, f_{FS}(d))), where (b, \ldots, d) are the fuzzy systems at Level V that contain rule-base (k) (set of fuzzy rules);</td>
</tr>
<tr>
<td>(c) Level III - Individual rule: the fitness function is (f_{IR}(j) = \text{mean}(f_{RB}(m), \ldots, f_{RB}(p))), where (m, \ldots, p) are the rule-bases at Level IV that contain individual rule (j);</td>
</tr>
<tr>
<td>(d) Level II - Partition set: the fitness function is (f_{PS}(q) = \max(f_{FS}(x), \ldots, f_{FS}(z))), where (x, \ldots, z) are the fuzzy systems at Level V that contain partition set (q);</td>
</tr>
<tr>
<td>(e) Level I - Inputs and delays selection: the fitness function is (f_{IS}(l) = \max(f_{PS}(e), \ldots, f_{PS}(h))), where (e, \ldots, h) are the fuzzy systems at Level V that contain the selection number (l) of inputs and delays;</td>
</tr>
<tr>
<td>4. Each level is evolved, considering it as a separate genetic algorithm. If the stop condition does not hold, do:</td>
</tr>
<tr>
<td>(a) Generation = Generation + 1;</td>
</tr>
<tr>
<td>(b) For each level, apply the following evolutionary operators to form a new population: (1) selection, (2) crossover and (3) mutation;</td>
</tr>
<tr>
<td>(c) For each level, replace the current population with the new evolved population;</td>
</tr>
<tr>
<td>(d) Return to step 3.</td>
</tr>
</tbody>
</table>

Each level of the genetic hierarchy is evolved separately as an independent genetic algorithm using its own population and its own kind of fitness function. However, since the values of the fitness functions of each level depend on all the other populations, the evolution of each level is also influenced by the evolution of every other level. It should also be noted that, before any evolutionary tuning
can start, it is necessary to initialize the fuzzy rule base and membership functions populations with Algorithm 1, in order to avoid the issues described in Sec. 3.

The genetic operators used by the HGA during its co-evolutionary tuning were: Roulette Wheel selection, Single Point crossover, Uniform mutation and Weakest Individuals replacement. For a better understanding of these genetic operators, [10] is recommended.

5 Simulation Results

This section describes the implemented procedures, and respective results, in order to demonstrate the performance and effectiveness of the proposed method, for an automatic extraction of all fuzzy parameters of a fuzzy controller for industrial processes.

First, a standard fuzzy controller is developed with the aim of providing the necessary input/output dataset, allowing the proposed HGA method to learn the FLC’s parameters. The dataset is obtained by applying the developed FLC to a water tank process simulation, where it is used to control the water level of the tank, thereby extracting the values of the input and output variables that constitute the dataset. If the aim was to learn the control actions of a human expert, the dataset would be obtained by extracting information of the manually controlled process by the expert.

In the second part of this section, the proposed algorithm is applied to the previously obtained dataset, determining a new fuzzy controller. Subsequently, the responses of both fuzzy controllers are compared by applying them to the same water tank process simulation, varying the water level reference along time.

The results of Section 5.2 were obtained by considering 5000 generations, and fixing the populations of each level as: 100 individuals for Level V, 80 individuals for Level IV, 200 individuals for Level III, 15 individuals for Level II and 200 individuals for Level I.

5.1 Standard Fuzzy Controller Developed to Obtain the Input/Output Dataset

In the present subsection, a standard fuzzy controller is implemented to control the level of a water tank. The tank is illustrated in Fig. 4 and has the following parameters:

- \( a \): Cross-section of the Tank outlet hole (\( m^2 \));
- \( A \): Tank cross-section (\( m^2 \));
- \( h \): Tank water level (m);
- \( g \): Gravitational acceleration (\( m/s^2 \));
- \( v \): Voltage applied to the pump;
- \( \gamma \): Constant related to the flow rate into the tank.

The mathematical model of the tank is represented by:

\[
\frac{dh}{dt} = -\frac{a}{A} \sqrt{2gh} + \frac{\gamma}{A} v,
\]

where the parameter values considered for simulation were: \( a = 0.04 m^2 , A = 1.5 m^2, h = 2 m, g = 9.8 m/s^2 \) and \( \gamma = 0.5 \).

Once the tank parameters are defined, the fuzzy controller’s implementation can begin. To commence this process, it is necessary to define the controller’s knowledge base, i.e., membership functions and fuzzy rules. To build the membership functions, we have to determine the fuzzy system linguist variables. In this paper, the following linguistic variables are considered: level error, \( E(t) \), and flow rate, \( R(t) \) [the derivative of \( E(t) \)], as input variables; Incremental valve actuation, \( \Delta u(t) \), as the output variable.

Figures 5a, 5b and 5c, illustrate the membership functions of \( E(t) \), \( R(t) \) and \( \Delta u(t) \), respectively. The next step of the FLC implementation is the definition of the fuzzy rule base. The rule base was defined employing the generalized rule table presented in Table 1, where \( NB, NS, ZE, PS \) and \( PB \) represent the linguistic terms: Negative Big, Negative Small, Zero, Positive Small and Positive Big, respectively. The knowledge base membership functions are then tuned by a trial and error procedure.

Finally, the last step of the FLC implementation consists in defining the fuzzifier, inference engine and defuzzifier used by the controller. For that matter, the following elements were considered: singleton fuzzification, algebraic product \( t \)-norm, Mamdani product implication, maximum rule aggregation and center of gravity defuzzification.

<table>
<thead>
<tr>
<th>( E(t)/R(t) )</th>
<th>( NB )</th>
<th>( NS )</th>
<th>( ZE )</th>
<th>( PS )</th>
<th>( PB )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( NB )</td>
<td>( NB )</td>
<td>( NB )</td>
<td>( NB )</td>
<td>( NS )</td>
<td>( ZE )</td>
</tr>
<tr>
<td>( NS )</td>
<td>( NB )</td>
<td>( NB )</td>
<td>( NS )</td>
<td>( ZE )</td>
<td>( PS )</td>
</tr>
<tr>
<td>( ZE )</td>
<td>( NB )</td>
<td>( NS )</td>
<td>( ZE )</td>
<td>( PS )</td>
<td>( PB )</td>
</tr>
<tr>
<td>( PS )</td>
<td>( NS )</td>
<td>( ZE )</td>
<td>( PS )</td>
<td>( PB )</td>
<td>( PB )</td>
</tr>
</tbody>
</table>

Table 1: Generalized fuzzy rule structure used by the standard FLC.

5.2 HGA Application and Result Analysis

This subsection analyses the performance and effectiveness of the HGA proposed by this paper. First, an input/output dataset is obtained by simulating a controlled water tank process with the FLC described in Section 5.1. Once the dataset is obtained, the HGA is applied, therefore learning the desired FLC parameters and structure. After attaining the FLC parameters, both controllers are compared while controlling the same water tank process.
After obtaining the learning dataset, the HGA was applied, reproducing the following results. First, Fig. 6 shows, both target and learnt, command signals. Next, Fig. 7 compares the responses of the standard and learnt FLC’s, as well as the response obtained by the initialization method, while varying the water level reference along time. Finally, the learnt FLC’s membership functions and fuzzy rules are shown in Figs. 8a, 8b and 8c, and Table 2, respectively.

Table 2: Fuzzy rule structure obtained by the HGA.

<table>
<thead>
<tr>
<th>E(t)/R(t)</th>
<th>NB</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>-</td>
<td>NB</td>
<td>NB</td>
<td>NB</td>
<td></td>
</tr>
<tr>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZE</td>
<td>ZE</td>
<td>ZE</td>
<td>ZE</td>
<td>ZE</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>PS</td>
<td>-</td>
<td>PS</td>
<td>-</td>
<td>PS</td>
</tr>
<tr>
<td>PB</td>
<td>PS</td>
<td>ZE</td>
<td>PB</td>
<td>PB</td>
<td>PB</td>
</tr>
</tbody>
</table>

When analyzing the obtained results, it can be verified that the standard fuzzy controller was learnt by the HGA in a satisfactory manner, even though the learning process originated a controller with a minor controlling error. The steady state error obtained for each controller was: 0.01m, for the standard FLC, 0.11m, using the initialization method and, 0.03m, with the proposed HGA. The error of the proposed HGA may not be very significant in many industrial processes, since most of these processes do not require a high control precision. If an higher accuracy level was necessary, the proposed algorithm could easily be applied to initialize the required fuzzy knowledge base of adaptive controllers, such as [9]. In, Fig. 7 it may also be verified that the HGA optimized the initial FLC response given by the initialization method of Algorithm 1.

Furthermore, when viewing Table 2 and Figs. 8a, 8b and 8c, it can also be observed that HGA optimized the set of fuzzy rules and membership functions, therefore being necessary a much smaller set of rules to define the FLC when compared to the standard FLC rule set (Table 1). Besides choosing the adequate set of rules and membership functions, the HGA also identified the best set of input variables as being E(t) and R(t). The results presented in Fig. 7 were obtained by selecting the best individual evolved by the HGA, which is composed by the...
The proposed method does not require any prior knowledge concerning the fuzzy rule structure; membership functions location or shape; implication and aggregation operators; defuzzification methods, or selection of an adequate set of input variables and their respective time delays.

To validate and demonstrate the performance and effectiveness of the proposed methodology, the algorithm was applied to a dataset obtained by using a standard FLC to control the water level of a tank during simulation. Moreover, both controllers were compared, verifying their responses when subjected to the same water level references along time. As can be seen by the obtained results, the FLC learnt by the proposed HGA presents a similar response to the target standard FLC, therefore achieving the goal of this work. The fuzzy controller obtained by the HGA can then be a starting point for further improvements by adaptive methods [9].

In future work, other antecedent membership function optimization methods will be investigated, as well as the inclusion of self-adaptive strategies, in order to improve the fuzzy system learning and reduce the computational effort.

Acknowledgement

This work was supported by Mais Centro Operacional Program, financed by European Regional Development Fund (ERDF), and Agência de Inovação (AdI) under Project SInCACI/3120/2009.

Jérôme Mendes is supported by Fundação para a Ciência e a Tecnologia (FCT) under grant SFRH/BD/63383/2009.

References


Figure 8: Membership functions of: (a) Level Error, $E(t)$; (b) Flow Rate, $R(t)$; (c) Incremental Valve Actuation, $\Delta u(t)$.