2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

GPFIS-Control: A Fuzzy Genetic Model for Control Tasks

Adriano S. Koshiyama, Tatiana Escovedo, Marley M. B. R. Vellasco, and Ricardo Tanscheit

Abstract— **This work presents a Genetic Fuzzy Controller (GFC), called Genetic Programming Fuzzy Inference System for Control tasks (GPFIS-Control). It is based on Multi-Gene Genetic Programming, a variant of canonical Genetic Programming. The main characteristics and concepts of this approach are described, as well as its distinctions from other GFCs. Two benchmarks application of GPFIS-Control are considered: the Cart-Centering Problem and the Inverted Pendulum. In both cases results demonstrate the superiority and potentialities of GPFIS-Control in relation to other GFCs found in the literature.**

Keywords— Fuzzy Logic Control, Genetic Fuzzy Controller, Muti-Gene Genetic Programming.

I. Introduction

FUZZY Logic Controllers (FLCs) [1] have been extensively used as an alternative to manipulate and describe complex systems when traditional control methods do not provide viable solutions. FLCs have the capacity of modeling systems by using fuzzy "if-then" rules, normally provided by an expert. Classical approaches employ either a Mamdani-type Fuzzy Inference System (FIS) [2-3] or a Takagi-Sugeno (TSK) FIS [4-5]. FLC parameters (rule base, membership function parameters, etc.) can be tuned by an expert or by employing a learning approach. In this respect, the current work considers Genetic Fuzzy Systems [3,6], or, to more specific, Genetic Fuzzy Controllers.

In Genetic Fuzzy Controllers (GFC) the automatic learning and tuning is based on a Genetic-based Meta-Heuristic (GBMH). Some works consider FLCs embedded with a Genetic Algorithm (GA) to tune membership function parameters [7-8] or to search for concise fuzzy rule bases [9-10]. More recent works explore Genetic Programming (GP) to build an FLC by using methodologies and concepts similar to those employed on a GA based FLC [11-12].

In general, it is advantageous to use a GBMH exclusively to search for the FLC best configuration. In this perspective, the meta-heuristic is seen as a tool to build an FLC and not as a mechanism that may change reasoning. Still, in frameworks with a high level of hybridization, in which a genetic-based meta-heuristic has a higher participation, it may be possible to obtain better accuracy. Examples are Neuro-Fuzzy models [13,14], where Neural Networks play an important role in the hybrid architecture, enabling high accuracy and fast convergence.

This work deals with a GFC called Genetic Programming Fuzzy Inference System for Control tasks (GPFIS-Control). It makes use of Multi-Gene Genetic Programming [15-16] for extracting knowledge from the plant. The resulting architecture should: (1) automatically tune the FLC parameters, (2) make the plant output reach the setpoint as fast as possible, (3) provide linguistic comprehension for each FLC action and (4) be easy to implement.

This paper is organized as follows: the next section describes related works on GFC and considers some applications involving GP. Section III describes Multi-Gene Genetic Programming and GPFIS-Control in detail. Case studies are considered in Section IV and section V concludes the work.

II. Related Works

The first attempt to build an FLC by using GBMH algorithms was presented in [7], where a GA was used to tune membership functions parameters of input and output variables. Subsequently, many other works have employed evolutionary algorithms, mostly GA, to tune FLC parameters and search for concise rule bases [17-19].

Several works can be found in the GFC area, such as [9], which presents an evolutionary procedure to modify rules, initially set by an expert, for a Mamdani type FLC. In [20] membership functions, rule sets and consequent types (TSK or Mamdani types) are tuned by a GA based on Symbiotic Evolution. Two other approaches are: [8], which employs linguistic hedge operators, selected by a GA, to tune membership functions, and [10] where a hierarchical self-organized GA-based scheme is proposed.

Recently, most works that make use of GA to tune FLCs focus on real applications [19,21,22]. Type-2 FLC have also been tuned through GA [18]. Lastly, some non-GBMH works for tuning an FLC consider Particle Swarm Optimization [23] and other bio-inspired algorithms [24].

Few attempts have been made to build an FLC by using GP, despite its dynamic structure that benefits rule base codification [6]. The first works in this sense were [25] and [26], which used a type-constrained GP to build a fuzzy rule based system. In [27] an FLC based on GP for mobile robot path tracking is presented. More recently, [12] proposes the use of Memetic GP to build a TSK FLC. All those approaches adapt the GP structure to formulate an FLC in a canonical way, similarly to a GA common procedure. Small advantages of GP are effectively used by these authors, but many possibilities arise, such as the use of combinations of different t-norms and t-conorms, of linguistic hedges and of different

A.S. Koshiyama, M.M.B.R. Vellasco and R. Tanscheit are with the with the Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, Marquês de São Vicente street, 225, Gávea - Rio de Janeiro, RJ - Brasil – CEP: 22451-900 Cx.P.: 38097 - Tel: (55 21) 3527-1001 (e-mail: [adriano,tatiana,marley,ricardo]@ele.puc-rio.br).

This work was supported in part by Brazil Reseach Council (CNPq and CAPES) and Pontifical Catholic University.

aggregation operators.

GPFIS-Control is a novel GFC based on Multi-Gene Genetic Programming. This model builds a Pittsburgh-type Fuzzy Rule Based System, making use of a different reasoning method to learn fuzzy rules.

III. GPFIS-CONTROL

A. Multi-Gene Genetic Programming

Genetic Programming (GP) [28-29] belongs to the Evolutionary Computation field. Typically, it employs a population of individuals, each of them denoted by a tree structure that codifies a mathematical equation, which describes the relationship between the output Y and a set of input terminals X_j (j=1,...,J) (features, in the current work).

Multi-Gene Genetic Programming (MGGP) [15-16] denotes an individual as a structure of trees, also called genes, that receives X_j and tries to predict Y (Figure 1). Each individual is composed of D functions f_d (d=1,...D) that map X_j variables to Y through user-defined mathematical operations. In GP terminology, the X_j input variables are included in the Terminal set, while the mathematical operations (plus, minus, etc.) are inserted in the Function Set (or Mathematical Operations Set).

With respect to genetic operators, mutation in MGGP is similar to that in GP. As for crossover, the level at which the operation is performed must be specified: it is possible to apply crossover at high and low levels. Figure 2 presents a multi-gene individual with five equations (D=5) accomplishing a low level crossover, while Figure 3 shows the mutation operation.

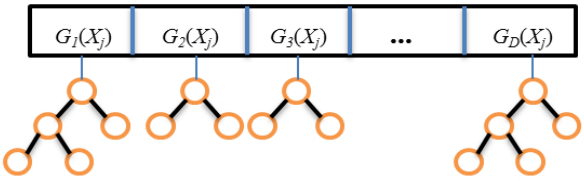

Fig. 1. Example of a multi-gene individual.

The low level is the space where it is possible to manipulate the structures (terminals and functions) of equations present in an individual. In this case, both operations are similar to those performed in GP.

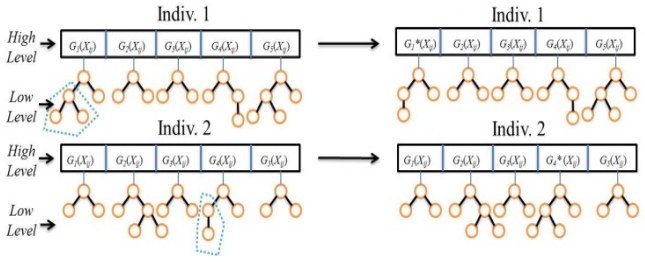

Fig. 2. Low-level crossover in a multi-gene individual.

The high level, on the other hand, is the space where expressions can be manipulated in a macro way. An example of high level crossover is displayed in Figure 4. By observing the dashed lines, it can be seen that the equations were switched from an individual to the other.

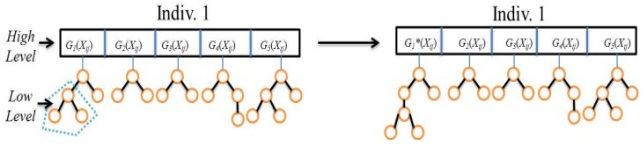

Fig. 3. Mutation operation in a multi-gene individual.

The cutting point can be symmetric – the same number of equations is exchanged between individuals –, or asymmetric. Intuitively, high level crossover has a deeper of effect on the output than low level crossover or mutation has. In case of GPFIS-Control, the high level crossover used is symmetric.

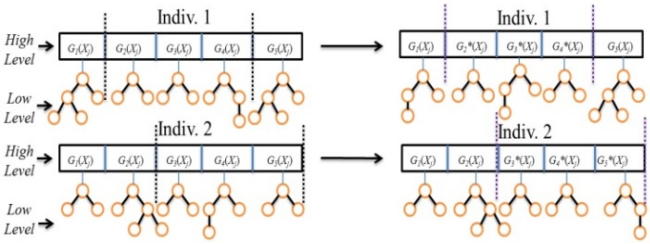

Fig. 4. High level crossover in a multi-gene individual.

B. GPFIS-Control

The GPFIS-Control model is shown in Figure 5. The control signal y_t is sent to the plant at time t (t=0,1, ..., T). The plant outputs z_{tk} (k=1,...K) are fed back to the input, so that the result of the difference between each feedback and its respective setpoint is the error signal $x_{tk} = z_{tk} - \text{Ref}_k$. By using x_{tk} it is possible to build a control signal y_t in order to satisfy performance criteria.

In general guidelines, the GPFIS-Control model is comprised of four sections: fuzzification, inference, defuzzification and evaluation. The inference process begins when each feedback error x_{tk} is mapped on fuzzy sets. Then, functions that map each linguistic state of x_{tk} to a state of y_t are synthesized based on MGGP principles. The crisp control signal is obtained through defuzzification. This solution is evaluated and then selection and recombination operators are applied. These steps are repeated until a stopping criterion is met.

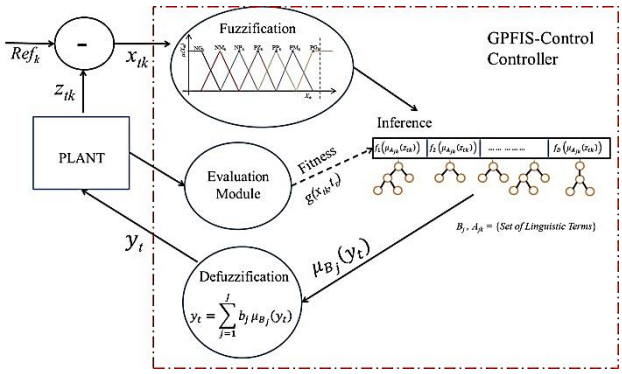

Fig. 5. Block diagram of GPFIS-Control model.

1) Fuzzification

Let x_{tk} and y_t admit J distinct linguistic terms, or fuzzy sets. These are defined by by normalized and uniformly distributed membership functions [30], as shown in Figure 6, for the k-th plant output.

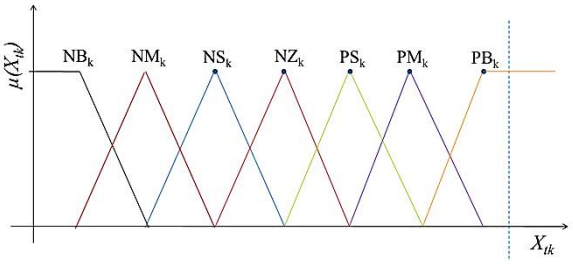

Fig. 6. Example of membership functions.

After fuzzification of each input x_{tk}, the GPFIS-Control inference process initiates.

2) Inference

The inference procedure consists of three stages: Formulation, Partitioning and Aggregation. In Formulation t-norm, t-conorm, linguistic hedges and negation operators are defined. In Partitioning the mechanism that connects each antecedent with a consequent is established. In Aggregation operators used to combine all rules are defined. Figure 7 shows a diagram of this process.

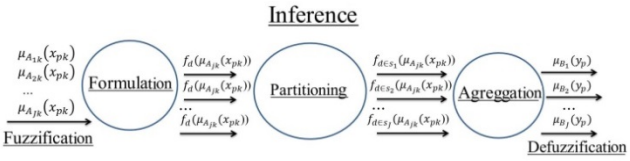

Fig. 7. Diagram of Inference procedure.

a) Formulation

Through each $\mu_{A_{jk}}(x_{tk})$ (membership degree of x_{tk} to a fuzzy set A_{jk}), GPFIS-Control evolves a controller whose ouput has several terms (B_1 = Negative Big, ..., B_7 = Positive Big, for example), with membership degrees given by:

$$\mu_{B_1}(y_t) = g\left[f_{d\in s_1}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)\right] \quad (1)$$

$$\mu_{B_2}(y_t) = g\left[f_{d\in s_2}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)\right] \quad (2)$$

...

$$\mu_{B_J}(y_t) = g\left[f_{d\in s_J}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)\right] \quad (3)$$

where $f_{d\in s_j}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)$ represents a set of functions, where each one combines all $\mu_{A_{jk}}(x_{tk})$, k=1,.., K, by using a set of user-defined mathematical operations; s_j (j=1,…, J) is an index set that describes which d-th function f_d is related to the j-th consequent term ($d \in s_j$). Methods to define s_j are best described in the Partitioning stage. In order to each function f_d associated to s_j behave as a fuzzy rule, it needs to employ t-norm, t-conorm, negation and linguistic hedges operators, with the aim to represent logic connectives for each linguistic term induced by $\mu_{A_{jk}}(x_{tk})$. Finally, g aggregates the activation degrees of each rule set (represented by $f_{d\in s_j}$) in a final value. Therefore, if a set A_{jk} is activated, GPFIS-Control builds a rule set (function set) that combines all membership degrees ($\mu_{A_{jk}}(x_{tk})$) and produces an action.

In Formulation some parameters of GPFIS-Control are defined. In MGGP, initial parameters are called Terminals (input variables) and Mathematical Operations or Function Set (plus, times, etc.). In GPFIS-Control, the terminology will be Input Fuzzy Sets and Fuzzy Operators Set, respectively. Table I presents the initial user-defined parameters.

TABLE I
INPUT FUZZY SETS AND FUZZY OPERATORS

Input Fuzzy Sets	Fuzzy Operators Set
$\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})$	t-norms, t-conorms, negation and linguistic hedges operators

Subsequently, by using the Fuzzy Operators Set, each $\mu_{A_{jk}}(x_{tk})$ is combined in order to best describe the actions $\mu_{B_j}(y_t)$ taken by the controller. It is possible to enter into the Input Fuzzy Sets stage with a negated or modified ("hedged") fuzzy set, instead of using negation and linguistic hedge operators in the Fuzzy Operators Set stage. This entails a larger search space, but can be of help in rules analysis.

b) Partitioning

Let $S=\{s_1,s_2,\dots,s_J\}$, where each s_j represents which f_d ($d=\{1,\dots,D\}$) is related to the j-th consequent B_j. The method that describes which d-th function is associated to s_j is called Uniform Division. This partitioning method makes use of a simple heuristic, given by:

1. Compute: $U = \lfloor \frac{D}{J} \rfloor$ (where $\lfloor . \rfloor$ is the floor operator).
2. Partition: $s_1 = \{1,\dots,U\}$, $s_2 = \{U+1,\dots, 2*U\},\dots,$ $s_J = \{U*(J-1)+1,\dots,U*J\}$.

As an example, consider D (nr. of functions) = 10 and J (nr. of consequent terms) = 5. Thus $U = 2$, $s_1 = \{1, 2\}$, $s_2 = \{3,4\}$, $s_3 = \{5,6\}$, $s_4 = \{7,8\}$, $s_5 = \{9,10\}$. Figure 8 illustrates this process.

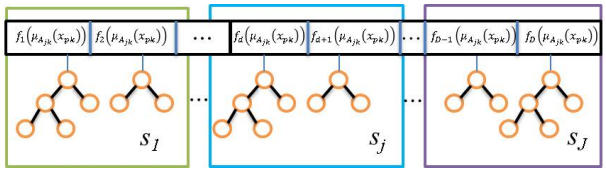

Fig. 8. Partitioning method: Uniform Division.

In summary, each f_d is uniformly divided for each s_j so that a consequent has at least one rule associated to it. This method is similar to others GFS based on GP, such that consequent and antecedent terms are both synthesized. Through the definition of the rule set associated to each consequent ($S=\{s_0,s_1,s_2,...,s_J\}$), the next step is to aggregate them, in order to generate a final degree of activation.

c) Agreggation

Many works on aggregation operators may be found in the literature [31-32].

Some examples of $g\left[f_{d\in s_1}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)\right]$ are:

- $g \rightarrow \max\left[f_{d\in s_1}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)\right]$: max aggregation operator are the most common used on Mamdani type FIS.
- $g \rightarrow \frac{1}{card(s_j)}\sum_{\mathrm{d\in s_j}}[f_{d\in s_1}\left(\mu_{A_{j1}}(x_{t1}),\dots,\mu_{A_{jK}}(x_{tK})\right)]$: arithmetic mean operator intends to provide same weights for each element of the rule set associated to the j-th consequent.

In [31] several aggregation operators are presented. It can be shown that t-norms and t-conorms are special cases of aggregation operators. After the user definition for the aggregation operators, it is possible to compute the membership degrees for different actions $\mu_{B_j}(y_t)$ taken by the controller. Then, the defuzzified control signal y_t is computed.

3) Defuzzification

Basically, a defuzzification method (center of gravity, mean of maximum, etc.) produces a crisp value that is an interpretation of the information contained in a fuzzy set. In GPFIS-Control the height method is used:

$$y_t = \frac{\sum_{j=1}^{J} b_j\, \mu_{B_j}(y_t)}{\sum_{j=1}^{J} \mu_{B_j}(y_t)} \tag{4}$$

where b_j represents the center (location) parameter of each B_j. The maximum height method may be employed when the control signal assumes values in some finite set:

$$y_t = \frac{\sum_{j=1}^{J} \phi_j b_j\, \mu_{Bj}(y_t)}{\sum_{j=1}^{J} \phi_j\, \mu_{Bj}(y_t)} \tag{5}$$

where ϕ_j is an indicator function, such that $\phi_j = 1$, when $\mu_{B_j}(Y_t) > \mu_{B_l}(Y_t)$, for all $l=1,...,J$, e $l \neq j$, e $\phi_j = 0$, otherwise.

4) Evaluation

The right definition of the fitness function is crucial for a good performance of GPFIS-Control. For optimal tracking of a trajectory, a possible fitness function is the Mean Squared Error (*MSE*):

$$MSE = \frac{1}{K}\sum_{k=1}^{K}(x_{tk})^2 \tag{6}$$

When *MSE* is minimized the GPFIS-Control model successfully obtains a trajectory close to the setpoint. In minimum time problems, the fitness function may be the time (t) the output takes to reach an $MSE < \varepsilon$, where ε is a tolerance.

GPFIS-Control tries to reduce the size and complexity of the rule base by employing a simple heuristic called Lexicographic Parsimony Pressure [33]. This technique is only used in the selection phase: given two individuals with the same fitness, the best one is that with fewer nodes. Fewer nodes indicate rules with fewer antecedents, hedge and negation operators, as well as few functions (f_d), and, therefore, a small rule set. After the evaluation process, selection and recombination operators are applied in order to generate a new population.

IV. Case Studies

A. Experimental Settings

Two benchmark have been considered as applications for the GPFIS-Control model: cart-centering problem [25,28] and inverted pendulum [9,12].

The cart-centering problem consists of a cart with mass m, moving on a frictionless rail; at some instant t its position is x_t (m), with velocity v_t (m/s). The cart must stop ($v_t = 0$) at a user-defined setpoint *ref.* Tolerance values ε may be considered, so that $|x_t - ref.| < \varepsilon$ and $|v_t - ref.| < \varepsilon$. Plant dynamics are shown in Equations (7) and (8):

$$v_{t+\tau} = v_t + \tau\frac{F_t}{m} \tag{7}$$

$$x_{t+\tau} = x_t + \tau v_t \tag{8}$$

where τ is the sampling period and F_t is the force (N) applied by the controller to the cart at time t. The objective is to reach the setpoint in minimum time. The performance of

TABLE II
CONFIGURATIONS SET FOR CART-CENTERING PROBLEM

Variable	Domain
F_t	[-2.5, 2.5] N
v_t	[-2.5, 2.5] m/s
x_t	[-2.5, 2.5] m
Parameter	**Value**
τ	0.02s
ε	0.5
m	2.0 kg
ref.	$x_t = v_t = 0$

GPFIS-Control has been compared to the GFC presented in [25]. Several configurations for GPFIS-Control (t-norms, aggregation operators, etc.) have been evaluated. To perform a fair comparison, configurations were the same as those in [25] for all variables and parameters. Table II display these values.

GPFIS-Control is required to move the cart until $|x_t - 0| < 0.5$ and $|v_t - 0| < 0.5$, given 16 initial values uniformly distributed on the x_t domain. The fitness function has been defined as:

$$Fitness = t_\varepsilon + \sum_t |x_t| \quad (9)$$

where t_ε is the the time needed to satisfy the stopping criteria ($|x_t - 0| < 0.5$ and $|v_t - 0| < 0.5$). An individual in the GPFIS-Control population is considered unfeasible if it cannot stop the cart in 10 seconds (500 sampling steps). Given the best solution, it is applied to 1000 initial random positions in order to evaluate the time taken by GPFIS-Control to stop the cart. The following procedure has been executed 10 times: (*i*) generate a GPFIS-Control model and, (*ii*) applying it on 1000 random position, in order to produce statistical relevant results. Finally, in each execution 25000 evaluations (population size = 50 and number of generations = 500, respectively) have been performed to generate a fair comparison with [25]. Table III displays the parameters.

TABLE III
REMAINING GPFIS-CONTROL CONFIGURATIONS

Parameter	Value
Tournament Size	5
Maximum Tree Depth	5
Elitism Rate	1%
Maximum rules per individual	50
Low level crossover rate	75%
High level crossover rate	50%
Mutation rate	*20%*
Direct reproduction rate	*5%*
Input Fuzzy Sets	*7 Fuzzy Sets + Classical Negation* of each Fuzzy Sets per variable*
Fuzzy Operators Set	*t-norm: product, others: described for each experiment*

* Used when first advised.

Based on the best configurations for the GPFIS-Control model in the cart-centering problem, the second experiment consists of a comparison with [12] regarding the inverted pendulum problem. In this, a cart of mass *M* with a pole of mass *m* and height *l* attached to its center moves on a frictionless rail. The controller must apply several F_t in order to increase or decrease v_t, and consequently change the angular velocity ω_t and the pendulum angle θ_t. The dynamic model can be found in [12,28].

In order to perform a fair comparison with [12], the feasible domain for each variable was set as: $\omega_t \in$ [-0.87, 0.87] *rad/s*, $\theta_t \in$ [-0.34, 0.34] *rad*, $F_t \in$ [-25, 25] *N*, while x_t and v_t are unconstrained, $M=1kg$, $m=0.1kg$, $l=0.5m$, $g=9.8m/s^2$ and $\tau=0,01s$. Two initial conditions were considered: $\theta_0 = \{-0,18, 0,18\}$rad, with $\omega_0 = \{0,0\}$rad/s and the setpoint is $ref=0$ *rad* with $\varepsilon=0.01$. The time allowed for the position $|\theta_t - 0| < 0.01$ to be reached is at most 1 second (100 sampling steps).

As in [12], 100,000 evaluations (population size = 100 and number of generations 1000) have been made. All this procedure was repeated 10 times, in order to generate statistical relevant results. Table III exhibit the remaining parameters used. The fitness function is [12]:

$$Fitness = \sum_{t=1}^{100} (\theta_t - ref)^2 \quad (10)$$

In both experiments seven fuzzy sets have been assigned to each variable (F_t, x_t, v_t, ω_t, θ_t), as shown in Figure 6. In some cases, the negation of a fuzzy set was entered in the Input Fuzzy Sets stage of the GPFIS-Control routine (as described in section 3.1a). All experiments were performed in MATLAB R2010a [34].

B. Results and Discussions

1) Cart-Centering Problem

The main results are presented in Table IV, considering the linguistic hedge square root, the classic negation operator and different aggregation operators (max and average) and defuzzification methods (height and maximum height). It can be seen that for almost all configurations, the use of the average aggregation operator reduces by about 39% the mean time taken by the controller to position the cart at $|x_t - 0| < 0.5$ and $|v_t - 0| < 0.5$. It may also be noted that the maximum height defuzzification reduces that time in 14% in average. However, the use of the negation operator does not incur in any substantial time decrease, although fewer rules are generated. In fact, the negation operator has a summarizing power, due to the enlargement of a fuzzy set support in the universe of discourse.

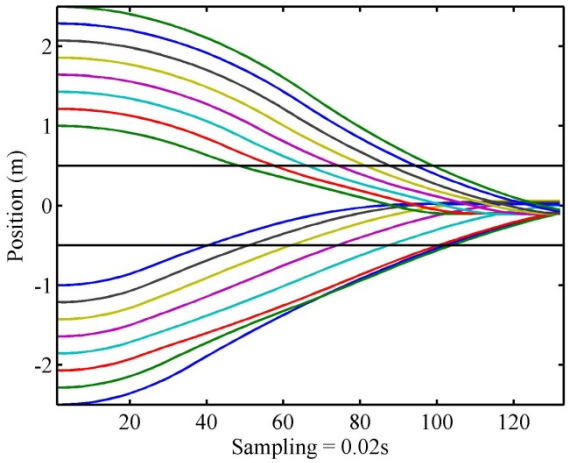

Fig. 9. Initial and final position for the best individual in an execution of GPFIS-Control, using Product+Root-Sq+MaxHeight and average aggregation operator configurations.

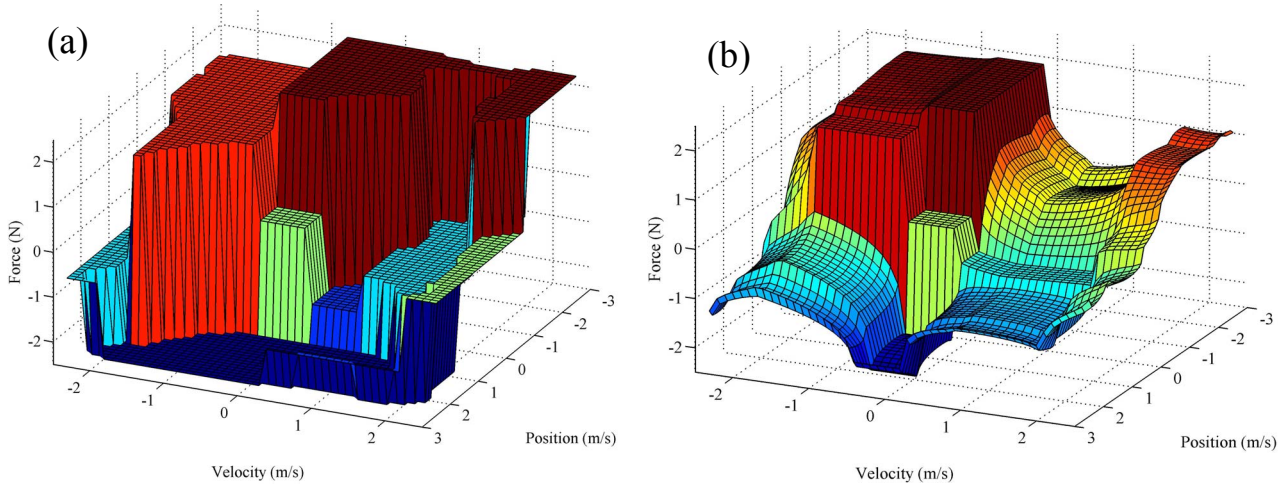

Fig 10. Response surface for the best individual in cart-centering for different defuzzification methods: (a): maximum height; (b) height.

The best configurations were: maximum height method for defuzzification and average as the aggregation operator. Figure 9 present the 16 initial positions and final positions when $|x_t - 0| < 0.5$ and $|v_t - 0| < 0.5$. Figure 10 exhibits the response surface for GPFIS-Control best configuration for (a): maximum height defuzzification method and (b): height defuzzification method. It can be seen that the surface for (b) is smoother than that for (a), due to broader set of values that F_t can assume when the height method is chosen.

The average best result for GPFIS-Control (135.8 steps) compares favorably with those of [25] (158 steps) and [35] (149 steps). The optimal solution is 129 steps.

2) *Inverted Pendulum*

Based on the best configuration previously established, GPFIS-Control has been applied to the inverted pendulum problem. Figure 11 shows the controller's behavior, generated by the best individual in 100,000 evaluations, given two initial conditions: θ_0 = {-0,18, 0,18}rad, with ω_0 ={0,0}rad/s. The average best result found for GPFIS-Control was 0.27 seconds to reach and stay at $|\theta_t - 0|<0.01$ during 1.00 second, generating in 14 rules in average. In [12] the GFC took 0.61 seconds to perform the same task, however producing few rules (7 rules).

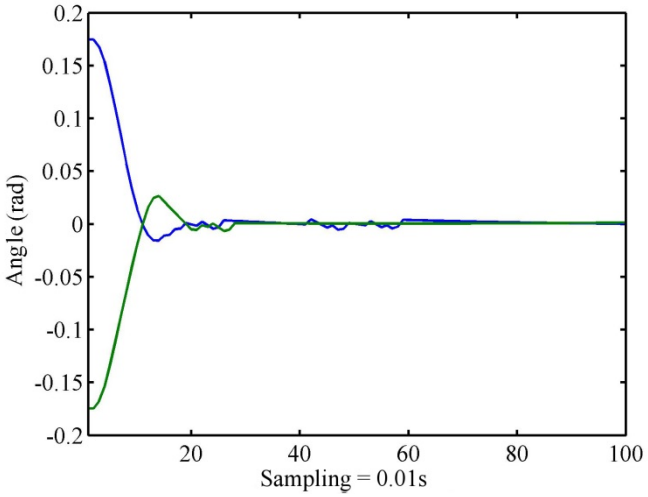

Fig. 11. Initial and final position for the best individual in an execution of GPFIS-Control, using Product+Root-Sq+MaxHeight and average aggregation operator configurations.

TABLE IV
RESULTS FOR GPFIS-CONTROL: CART-CENTERING PROBLEM

Attribute	Aggregation operator = Max			
	Product+Root-Sq+ Height	Product+Root-Sq+ MaxHeight	Product+Root-Sq+ Neg+Height	Product+Root-Sq+ Neg+MaxHeight
Average Steps (0.02s)	215.9	243.6	224.6	**203.5**
Std. Dev. Steps (0.02s)	25.73	94.09	37.89	60.78
Average Time (s)	4.318s	4.872	4.492	4.07
Average Rules	21	24	**14**	15
Attribute	Aggregation operator = Average			
	Product+Root-Sq+ Height	Product+Root-Sq+ MaxHeight	Product+Root-Sq+ Neg+Height	Product+Root-Sq+ Neg+MaxHeight
Average Steps (0.02s)	160.2	**135.8**	205.5	144.9
Std. Dev. Steps (0.02s)	18.92	18.94	38.99	11.43
Average Time (s)	3.204	2.796	4.11	2.89
Average Rules	*27*	*28*	*26*	***24***

V. CONCLUSIONS

A novel approach for solving control problems has been presented. It consists of a Genetic Programming Fuzzy Inference System for Control tasks (GPFIS-Control), based on Multi-Gene Genetic Programming. Its model considers the usual stages of a Genetic Fuzzy Inference System: fuzzification, inference, defuzzification and evaluation.

The performance of GPFIS-Control has been evaluated through two benchmarks problems: cart-centering and inverted pendulum. The use of aggregation, defuzzification and negation operators has been analyzed. It was shown that the right choice of defuzzification and aggregation operators improves results, while the use of negation may reduce the number of rules. When compared to other Genetic Fuzzy Controllers, GFPIS-Control has shown a better performance in average.

Future works shall consider other benchmark and actual problems, as well as new methods in formulation, partitioning and aggregation. For example, rules could be aggregated by using a weighted average, with adaptive weights for the rules during the controller operation. This could improve results with fewer rules. Build of others partitioning methods could help GPFIS-Control to select the most promising rules for each consequent. A sensitivity analysis of some parameters (tournament size, maximum tree depth, etc.) would help to evaluate their influence on the final result.

REFERENCES

[1] J. M. Mendel, Fuzzy logic systems for engineering: a tutorial. Proceedings of the IEEE, Vol.83, No.3, p.345-377, 1995.

[2] C. Elmas, C., O. Deperlioglu, and H. H. Sayan, Adaptive fuzzy logic controller for DC–DC converters. Expert Systems with Applications, Vol.36, No.2, pp.1540-1548, 2009.

[3] O. Cordón, A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems. International Journal of Approximate Reasoning, Vol.52, No.6, pp.894-913, 2011.

[4] J.S.R. Jang, C. T. Sun, and E. Mizutani, Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[5] R.E. Precup, and H. Hellendoorn, A survey on industrial applications of fuzzy control. Computers in Industry, Vol.62, No.3, pp.213-226, 2011.

[6] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy sets and systems, Vol.141, No.1, pp. 5-31, 2004.

[7] C. Karr, Genetic algorithms for fuzzy controllers. Ai Expert, Vol.6, No. 2, pp.26-33, 1991.

[8] B. D. Liu, C. Y. Chen, and J. Y. Tsao, Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol.31, No.1, pp.32-53, 2001.

[9] F. Herrera, M. Lozano, and J. L. Verdegay, A learning process for fuzzy control rules using genetic algorithms. Fuzzy sets and systems, Vol. 100, No. 1, pp.143-158, 1998.

[10] T. Pal, and N. R. Pal, SOGARG: A self-organized genetic algorithm-based rule generation scheme for fuzzy controllers. Evolutionary Computation, IEEE Transactions on, Vol.7, No.4, pp.397-415, 2003.

[11] E. Tunstel, and M. Jamshidi, On genetic programming of fuzzy rule-based systems for intelligent control. Int. Journal of Intelligent Automation and Soft Computing, Vol.2, No.3, pp.271-284, 1996.

[12] A. Tsakonas, Local and global optimization for Takagi–Sugeno fuzzy system by memetic genetic programming. Expert Systems with Applications, Vol.40, No.8, pp.3282-3298, 2013.

[13] N. Kasabov, and Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems, Vol.10, N.2, pp.144-154, 2002.

[14] R. J. Contreras, M.M.B.R. Vellasco, and R. Tanscheit, Hierarchical type-2 neuro-fuzzy BSP model. Information Sciences, Vol.181, No.15, pp.3210-3224, 2011.

[15] M. P. Hinchliffe, M. J. Willis, H. Hiden, M.T. Tham, B. McKay, and G.W. Barton, Modeling chemical process systems using a multi-gene genetic programming algorithm. in Proc. of the First Annual Conference of Genetic Programming, Massachussets, pp. 56-65, 1996.

[16] D. P. Searson, M. J. Willis, and G.A. Montague, Co-evolution of non-linear PLS model components. Journal of Chemometrics, Vol. 2, pp. 592-603, 2007.

[17] F. Herrera, Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence, Vol.1, No.1, pp.27-46, 2008.

[18] O. Castillo, and P. Melin, A review on the design and optimization of interval type-2 fuzzy controllers. Applied Soft Computing, Vol.12, No.4, pp.1267-1278, 2012.

[19] M. Fazzolari, R. Alcalá, Y. Nojima, H. Ishibuchi, and F. Herrera, A Review of the Application of Multiobjective Evolutionary Fuzzy Systems: Current Status and Further Directions. IEEE Trans. on Fuzzy Sets, Vol.21, No.1, pp.45-65, 2013.

[20] C. F. Juang, J. Y. Lin, and C. T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, Vol.30, No.2, pp.290-302, 2000.

[21] E. De Santis, A. Rizzi, A. Sadeghiany, and F. M. F. Mascioli, Genetic optimization of a fuzzy control system for energy flow management in micro-grids. in Proc. IFSA World Congress and NAFIPS Annual Meeting, pp. 418-423., June 2013.

[22] L. H. Hassan, M. Moghavvemi, H. A. Almurib, O. Steinmayer, Application of genetic algorithm in optimization of unified power flow controller parameters and its location in the power system network. International Journal of Electrical Power & Energy Systems, Vol.46, pp.89-97, 2013.

[23] R. P. Prado, S. García-Galán, J. Exposito, and A. J. Yuste, Knowledge acquisition in fuzzy-rule-based systems with particle-swarm optimization. IEEE Trans. Fuzzy Systems, Vol.18, No.6, pp.1083-1097, 2010.

[24] O. Castillo, R. Martínez-Marroquín, P. Melin, F. Valdez, and J. Soria, Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. Information Sciences, Vol.192, pp.19-38, 2012.

[25] E. Alba, C. Cotta, and J. M. Troya, Type-constrained genetic programming for rule-base definition in fuzzy logic controllers. in Proc. of the First Annual Conference on Genetic Programming, pp. 255-260, July 1996.

[26] E. Tunstel, and M. Jamshidi, On genetic programming of fuzzy rule-based systems for intelligent control. International Journal of Intelligent Automation and Soft Computing, Vol.2, No.3, pp.271-284, 1996.

[27] A. Homaifar, D. Battle, E. Tunstel, and G. Dozier, Genetic Programming Design of Fuzzy Logic Controllers for Mobile Robot Path Tracking. International Journal of Knowledge Based Intelligent Engineering Systems, Vol.4, No.1, pp.33-52, 2000.

[28] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Massachusetts: MIT Press, 1992.

[29] W. B. Langdon, and R. Poli, Foundations of Genetic Programming. Heidelberg: Springer-Verlag, 2002.

[30] G. J. Klir, and B. Yuan, Fuzzy sets and fuzzy logic. New Jersey: Prentice Hall, 1995.

[31] T. Calvo, A. Kolesárová, M. Komorníková, and R. Mesiar, Aggregation operators: properties, classes and construction methods. In: Aggregation Operators, T. Calvo et al. (Eds.), Heidelberg: Physica-Verlag, 2002, pp.3-104.

[32] Yager, R. R., Kacprzyk, J., & Beliakov, G. Recent developments in the ordered weighted averaging operators: theory and practice. Springer, 2011.

[33] S. Luke and L. Panait, Lexicographic parsimony pressure. In: Proceedings of the Genetic and Evolutionary Computation Conference, W. B. Langdon et al. (Ed.). New York: Morgan Kaufmann Publishers, 2002, pp. 829-836.

[34] MATLAB (2010). MATLAB 7.10.0 (R2010a). Massachusetts: The MathWorks Inc.

[35] P. R. Thrift, Fuzzy Logic Synthesis with Genetic Algorithms. In: Proc. International Conference on Genetic Algorithms, pp. 509-513, July 1991.