Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Hybrid Intelligent Systems Applied to The Pursuit-Evasion Game

Sameh F. Desouky
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
sameh@sce.carleton.ca

Howard M. Schwartz
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
schwartz@sce.carleton.ca

***Abstract*—This paper presents a new method of using hybrid intelligent systems to solve the problem of tuning the parameters of a fuzzy logic controller. Two different hybrid intelligent systems are introduced in this paper. Each system proposes learning in a two-stage iterative process. The first system combines a fuzzy logic controller with genetic algorithms to form the iterative genetic based fuzzy logic controller technique (IGBFLC). The second system combines a fuzzy logic controller with an adaptive network to form the iterative adaptive network fuzzy inference system (IANFIS). The proposed systems are applied to a model of pursuit-evasion game. In this model, we are seeking for the optimal strategy of the pursuer given that the evader plays its optimal strategy. The proposed systems are compared with the PD controller, the Genetic-based fuzzy logic controller and the ANFIS technique. Computer simulations and results show that when compared to the optimal strategy, the proposed systems outperform the other techniques.**

***Index Terms*—ANFIS, fuzzy logic controller, genetic algorithms, hybrid intelligent system, pursuit-evasion game.**

I. Introduction

Hybrid intelligent system is a combination of two or more learning algorithms in order to combine their strengths and overcome their weaknesses. For example, fuzzy logic controllers (FLCs) are currently being used to a great extend in engineering applications [1] specially for plants that are complex and ill-defined [2], [3] and plants with high uncertainty in the knowledge about its environment such as autonomous mobile robotic systems [4]. However, FLC has a drawback of finding its knowledge base which is based on a tedious and unreliable trial and error process [5]. One way to overcome this difficulty is to use supervised learning to tune the parameters of FLC. The meaning of the supervised learning is that the desired "target" output values are given, from an expert, for each input pattern to guide the process of learning [6].

In [7]–[9], fuzzy rules were generated from given desired input/output data pairs. Genetic algorithms (GAs) are used so as to make the FLC behave as closely as possible to the expert but not better. In [10], GAs are used to obtain the best membership functions (MFs) for a fuzzy control system that controls a semi-autonomous mobile robot avoiding obstacles. In our previous work [11], iterative genetic based fuzzy logic controller technique (IGBFLC) is used for a wall-following mobile robot application and the results showed that the proposed technique outperforms the expert.

In this paper, we will examine the use of the IGBFLC technique for the more complex pursuit-evasion game. In addition, we will use this method to modify the adaptive network fuzzy inference system (ANFIS) to form the iterative ANFIS technique (IANFIS). ANFIS is used to a great extend in different applications such as electrochemical field as an optimization method [12], [13], chemistry [14], electronics as an estimation method [15], and aerospace [16].

The ANFIS and the standard GAs are broadly part of the class of supervised learning systems. In other words, one must know the correct answer a priori and then the system can automatically be trained to the correct answer. However, we are considering the case where the a-priori correct answer is unknown. As such we use a two-stage learning process. In stage 1, a PD controller is used as a coarse estimate of the correct actions. The results of the PD controller are then used to train the FLC using any of the proposed systems (IGBFLC or IANFIS). In stage 2, the PD controller and the FLC tuned in stage 1 operate in parallel and then we get new input/output data pairs which are used again by any of the proposed systems to tune the parameters of the FLC. Stage 2 is repeated until the FLC has achieved the desired performance. The proposed systems are applied to a model of pursuit-evasion game. One reason for choosing this model is the fact that the optimal strategy is known [17] and hence the minimum time of capture can be calculated to be a reference for our results.

This paper is organized as follows: in Section II the pursuit-evasion model is described, also some basic terminologies for the FLC, the GAs, and the ANFIS are reviewed. The proposed systems are described in Section III. Section IV represents computer simulation and results. Finally, conclusions are pointed out in Section V.

II. Preliminaries

A. Pursuit-evasion Model

The pursuit-evasion game is one application of differential games in which we try to find the optimal strategy for a pursuer to catch an evader [18]. The pursuit evasion model is shown in Fig. 1. Equations of motion for the pursuer and the evader

978-1-4244-2794-9/09/$25.00

robots are [19], [20]

$$\begin{aligned} \dot{x}_i &= v_i \cos \theta_i \\ \dot{y}_i &= v_i \sin \theta_i \\ \dot{\theta}_i &= \frac{v_i}{R_i} \tan u_i \end{aligned} \tag{1}$$

where "i" is "p" for the pursuer and is "e" for the evader, (x_i, y_i) is the position of the robot, θ_i is the orientation, R_i is the turning radius , u_i is the steering angle and $u_i \in [-u_{i_{max}}, u_{i_{max}}]$ and v_i is the velocity which is governed by the steering angle, to avoid slips, such that

$$v_i = V_i \cos u_i \tag{2}$$

where V_i is the maximum velocity.

Our strategies are to make the pursuer faster than the evader ($V_p > V_e$) but at the same time to make it less maneuverable than the evader ($u_{p_{max}} < u_{e_{max}}$). The control strategy for the pursuer is to drive the angle difference between the pursuer and the evader, δ, to be zero where

$$\delta = \tan^{-1}\left(\frac{y_e - y_p}{x_e - x_p}\right) - \theta_p \tag{3}$$

The control strategy for the evader is to maximize the distance between them by two ways:

1) If the distance between the pursuer and the evader is greater than a certain amount, d, then the control strategy for the evader is

$$u_e = \begin{cases} -u_{e_{max}} & : \quad \delta_e < -u_{e_{max}} \\ \delta_e & : \quad -u_{e_{max}} \leq \delta_e \leq u_{e_{max}} \\ u_{e_{max}} & : \quad \delta_e > u_{e_{max}} \end{cases} \tag{4}$$

where

$$\delta_e = \tan^{-1}\left(\frac{y_e - y_p}{x_e - x_p}\right) - \theta_e \tag{5}$$

2) If the distance between the pursuer and the evader is smaller than d then the control strategy will be [20]

$$u_e = \theta_p + \pi - \theta_e \tag{6}$$

This strategy will increase the maneuverability of the evader and makes it more difficult for the pursuer to catch the evader. We choose this strategy to check the performance of the proposed systems. The capture occurs when the distance between the pursuer and the evader is less than a certain amount, ℓ. This amount is called the capture radius which is defined as

$$\sqrt{(x_e - x_p)^2 + (y_e - y_p)^2} < \ell \tag{7}$$

B. PD controller

We use a PD controller, as shown in Fig. 2, as the initial expert. The input to the PD controller is δ which acts as the error signal. The output of the PD controller is u_p which acts as the control action and is defined as

$$u_p = k_p \, \delta + k_d \, \dot{\delta} \tag{8}$$

where k_p and k_d are the proportional and the differential gain coefficients.

C. FLC

The FLC is shown in Fig. 3. It has two inputs δ and $\dot{\delta}$ and its output is u_p. For the inputs of the FLC, we use the gaussian MF described by

$$\mu(x) = e^{-\frac{1}{2}\left(\frac{x - c}{\sigma}\right)^2} \tag{9}$$

where σ is the standard deviation and c is the mean value. We use a zero order Takagi-Sugeno-Kang fuzzy inference system (TSK-FIS) in which the consequent part is a constant function and is described as follows

$$R_l : IF \; x_1 \; is \; A_1^l \; AND \ldots AND \; x_n \; is \; A_n^l \; THEN \; f_l = K_l \tag{10}$$

where R_l is the l^{th} rule, A_i^l is the i^{th} membership value for the input variable x_i, and K_l is the consequent parameter. The output is calculated using the weighted average method as follows

$$f(\bar{x}) = \frac{\sum_{l=1}^{M}((\prod_{i=1}^{n} \mu^{A_i^l}(x_i))K_l}{\sum_{l=1}^{M}(\prod_{i=1}^{n} \mu^{A_i^l}(x_i))} \tag{11}$$

D. GAs

Genetic algorithms (GAs) are search and optimization techniques that are based on a formalization of natural genetics [21], [22]. GAs have been used to overcome the difficulty and

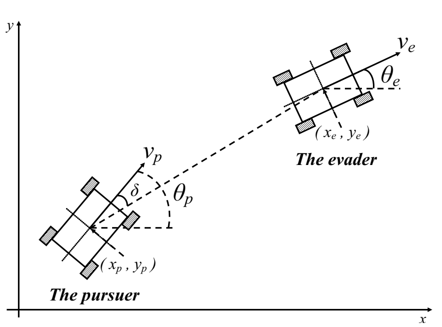


Fig. 1. The pursuer evader dynamics

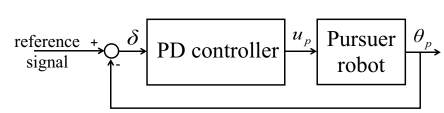


Fig. 2. Block diagram of a PD controller system

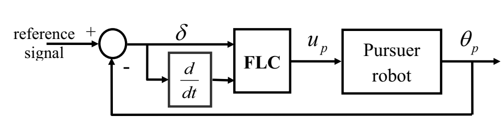


Fig. 3. Block diagram of a FLC system

complexity in the tuning of the FLC parameters such as MFs, scaling factors and control rules [8], [23], [24].

GAs search a multidimensional parameter space to find an optimal solution. A given set of parameters is referred to as a chromosome. The parameters can be either decimal or binary numbers. The GA is initialized with a number of randomly selected parameter vectors or chromosomes. This set of chromosomes is the initial population. Each chromosome is tested and evaluated based on a fitness function, in control engineering we would refer to this as a cost function. The chromosomes are sorted based on the lowest cost function or the ranking of the fitness functions. One then selects a number of the best, according to the fitness function, chromosomes to be parents of the next generation of chromosomes. A new set of chromosomes is selected based on reproduction.

In the reproduction process, we generate new chromosomes, which are called children. We use two GA operations. The first operation is a crossover in which we choose a pair of parents and select a random point in all of their chromosomes and make a cross replacement from one parent to another. The second operation is a mutation in which a parent is selected and we change one or more of its parameters to get a new child. Now, we have a new population to test again with the fitness function. The genetic process is repeated until the last iteration is reached.

E. ANFIS

ANFIS, which was originally introduced by Jang [25], is a combination of an adaptive network and fuzzy systems to learn fuzzy rules from examples. Fig. 4 shows the structure of ANFIS with 2 inputs, x_1 and x_2, and one output, f. Each input has 2 MFs. The structure has two types of nodes. The first type is an adaptive node (a squared shape) whose output need to be adapted (tuned) and the second type is a fixed node (a circled shape) whose output is a known function of its inputs. ANFIS structure has 5 layers. In layer 1, all nodes are adaptive. This layer has 4 outputs denoted by O_i^1, $i = 1,2,3,4$. The output of each node in layer 1 is the membership value of its input.

$$O_i^1 = \mu_{A_i} \tag{12}$$

In layer 2, all nodes are fixed. The AND operation between the inputs of each rule is calculated in this layer. This layer has 4 outputs denoted by O_i^2, $i = 1,2,3,4$. The output of each node in layer 2 is the multiplication (AND operation) of its inputs (the firing strength of each rule) as follows

$$O_i^2 = \omega_i = \prod_{i=1}^{4} \mu_{A_i} \tag{13}$$

In layer 3, all nodes are fixed. This layer normalizes the outputs of layer 2. This layer has 4 outputs denoted by O_i^3, $i = 1,2,3,4$. The output of each node in layer 3 is the normalized firing strength which is described as follows

$$O_i^3 = \overline{\omega}_i = \frac{O_i^2}{\sum_{i=1}^{4} O_i^2} = \frac{\omega_i}{\sum_{i=1}^{4} \omega_i} \tag{14}$$

In layer 4, all nodes are adaptive. The defuzzification process using the weighted average method, described by (11), is performed in this layer and the next layer. This layer has 4 outputs denoted by O_i^4, $i = 1,2,3,4$. The output of each node in layer 4 is

$$O_i^4 = O_i^3 f_i = \overline{\omega}_i f_i \tag{15}$$

Layer 5 is the output layer and has only one fixed node whose output, f, is the sum of all its inputs as follows

$$f = \sum_{i=1}^{4} O_i^4 = \sum_{i=1}^{4} \overline{\omega}_i f_i \tag{16}$$

which is the same as (11). Now, we can extract (16) and substitute in from (15) to be

$$\begin{aligned} f &= \overline{\omega}_1 f_1 + \overline{\omega}_2 f_2 + \overline{\omega}_3 f_3 + \overline{\omega}_4 f_4 \\ &= \overline{\omega}_1 K_1 + \overline{\omega}_2 K_2 + \overline{\omega}_3 K_3 + \overline{\omega}_4 K_4 = a\ \Theta \end{aligned}$$

where $a = (\overline{\omega}_1 \quad \overline{\omega}_2 \quad \overline{\omega}_3 \quad \overline{\omega}_4)$, and $\Theta^T = (K_1 \quad K_2 \quad K_3 \quad K_4)$. For P input/output data we have a linear equation of the form

$$f = A\Theta \tag{17}$$

where A is a $P \times M$ matrix (called the design matrix), M is the number of unknown parameters (here, $M = 4$), and Θ is an $M \times 1$ vector of the unknown parameters.

In ANFIS, the back-propagation (BP) or gradient descent (GD) technique is widely adopted technique for learning but it suffers from low convergence rate [26]–[29] and local minimum [26], [27]. So, by exploiting the linearity in the consequent part as derived in (17) we can use the hybrid learning rule. The hybrid learning rule is a combination of the BP technique and the least square estimate (LSE) method. In the LSE method, we tune the parameters of the consequent part using (17) in which the best solution for the least square estimator, Θ, to minimize $\|A\Theta - f\|$, is defined as

$$\hat{\Theta} = (A^T A)^{-1} A^T f \tag{18}$$

where $(A^T A)^{-1} A^T$ is the pseudo-inverse of A assuming that $(A^T A)$ is a nonsingular matrix and to be sure that $(A^T A)$ is nonsingular, P should be greater enough than M, such as [30]

$$P \geq (1.2 \sim 1.5)M \tag{19}$$

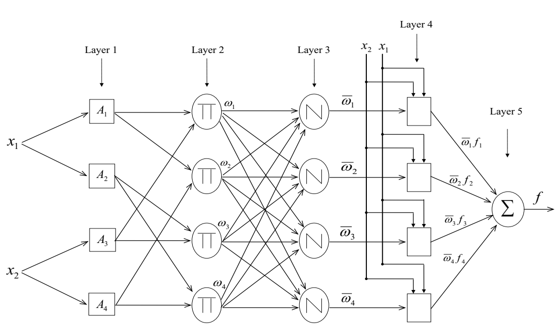


Fig. 4. Structure of ANFIS model using 4 rules

With the BP technique, we tune the parameters of the premise part to minimize the mean square error (MSE) of the output

$$\varepsilon = \frac{1}{2}(f_d - f)^2 \quad (20)$$

where f_d is the desired output and f is the actual output calculated from (16). Now we use the GD approach and according to the steepest descent algorithm, we make a change in σ_i and c_i along the $-ve$ gradient to minimize the error so,

$$\begin{aligned} \sigma_i(n+1) &= \sigma_i(n) - \eta \frac{\partial \varepsilon}{\partial \sigma_i} \\ c_i(n+1) &= c_i(n) - \eta \frac{\partial \varepsilon}{\partial c_i} \end{aligned} \quad (21)$$

where η is the learning rate.

III. The proposed IGBFLC/IANFIS systems

The proposed IGBFLC and the proposed IANFIS are shown in Fig. 5. In stage 1, GAs/ANFIS is used to tune the FLC parameters using input/output data pairs obtained from a PD controller which is used as an initial expert. In stage 2, the PD controller and the FLC tuned in stage 1 operate in parallel and then we get new input/output data pairs which are used by GAs/ANFIS again to tune the parameters of the FLC. Stage 2 is repeated until the FLC has achieved the desired performance. In the proposed systems we use the PD controller iteratively as described in Algorithm 1 for the proposed IGBFLC and in Algorithm 2 for the proposed IANFIS. The proposed IANFIS is described briefly as it has the same idea of the proposed IGBFLC.

IV. Computer Simulation and Results

The pursuer starts motion from position $(x_p, y_p) = (0,0)$ with initial orientation $\theta_p = 0$ rad and turning radius $R_p = 1$ m. The maximum velocity of the pursuer $V_p = 1$ m/s. The steering angle $u_p \in [-\frac{\pi}{6}, \frac{\pi}{6}]$ rad. The evader starts motion from several positions $(x_e, y_e) = (3,3)$, $(-2,4)$, $(2,5)$, and $(-1,5)$, to cover most of situations, with initial orientation $\theta_e = 0$ rad and turning radius $R_e = 1$ m. The maximum velocity of the evader $V_e = 0.5$ m/s which is half that of the pursuer (i.e. slower). The control input $u_e \in [-\frac{\pi}{3}, \frac{\pi}{3}]$ rad which is twice that

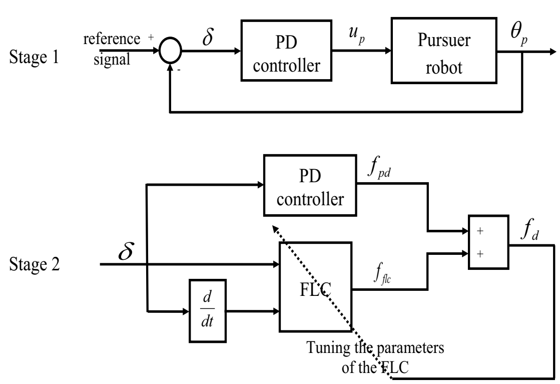


Fig. 5. Structure of the proposed IGBFLC/IANFIS systems

Algorithm 1 : The proposed IGBFLC

- Stage 1
  1) Run the PD controller and obtain L input/output data pairs.
  2) Initialize population with P chromosomes for the MFs parameters, (σ, c), and the consequent parameters, K.
  3) Repeat for each iteration i
     a) Repeat for each population p
        i) Construct a FLC from the p^{th} chromosome of the MFs parameters and the consequent parameters.
        ii) Repeat for each input/output data pair l
           – Evaluate the output, f^l_{flc}, of the constructed FLC using the inputs obtained in step 1.
        iii) Calculate the p^{th} fitness function. We use the mean square error function (MSE) as the fitness function which is defined as

$$MSE = \frac{1}{2L}\sum_{l=1}^{L}(f^l_d - f^l_{flc})^2 \quad (22)$$

     b) Sort the entire chromosomes (the MFs parameters, and the consequent parameters) of the population according to their fitness values.
     c) Select a portion of the sorted population as the new parents.
     d) Create a new generation for the remaining portion of population using crossover and mutation.
- Stage 2
  1) Put the tuned FLC in parallel with the PD controller.
  2) Run the system and obtain new L input/output data pairs.
  3) Run stage 1 from step 2 to 3.
  4) Does the tuned FLC achieve the desired performance?
     – If YES then end the algorithm.
     – If NO then go to the next step.
  5) Put the tuned FLC in parallel with the PD controller.
  6) Go to step 2.

Algorithm 2 : The proposed IANFIS

- **Stage 1**
  1) Run the PD controller and obtain input/output data pairs.
  2) Use ANFIS to tune the FLC parameters.
- **Stage 2**
  1) Put the tuned FLC in parallel with the PD controller.
  2) Run the system and obtain new input/output data pairs.
  3) Use ANFIS to retune the FLC parameters.
  4) Does the tuned FLC achieve the desired performance?
     – If YES then end the algorithm.
     – If NO then go to the next step.
  5) Put the tuned FLC in parallel with the PD controller.
  6) Go to step 2.

of the pursuer (i.e. more maneuverable). The capture radius is chosen to be $\ell = 0.10$ m.

We choose the proportional and the differential gain coefficients, k_p and k_d of the PD controller to be 0.99 and 0.02, respectively. We run the simulation of the PD controller 4 times for different initial positions of the evader and save the input/output data pairs which we use in the learning process.

In our work, we have 5 gaussian MFs for each input with a total of 10 MFs for the premise part. Each MF has 2 parameters (σ, c) to be tuned with a total of $10 \times 2 = 20$ parameters for the inputs. We use a zero-order TSK-FIS described in (10) to be

$$R_l : IF \ \delta \ is \ A_1^l \ AND \ \dot{\delta} \ is \ A_2^l \ \ THEN \ f_l = K_l$$

where A_1, A_2 are the membership values of the two inputs, δ and $\dot{\delta}$, respectively, $l = 1, 2, \ldots, 25$, and K_l is the consequent parameter. The consequent part has 25 parameters to be tuned. So, the total number of parameters to be tuned for the FLC is $20 + 25 = 45$ parameters. Table I represents the fuzzy decision table for the input variables, δ and $\dot{\delta}$ before tuning. For the FLC, the output can be calculated using (11) as follows

$$u_p = \frac{\sum_{l=1}^{25} (\prod_{i=1}^{2} \mu^{A_i^l}) K_l}{\sum_{l=1}^{25} (\prod_{i=1}^{2} \mu^{A_i^l})}$$

We build our proposed IGBFLC according to Algorithm 1. In stage 1, we run the PD controller for different initial positions of the evader to cover the most situations and get the input/output data pairs. The maximum and minimum values of the input/output data pairs are set to be the upper and lower boundaries of the inputs and the output MFs. We then use these input/output data pairs to tune the 45 parameters of the FLC using GAs. Then we start stage 2 in which we put the FLC obtained from stage 1 in parallel with the PD controller and obtain new input/output data pairs that are used to retune the parameters of the FLC. Stage 2 is repeated once then we separate the PD controller and the tuned FLC is now ready to be used. Values of GA parameters are shown in Table II.

We build our proposed IANFIS according to Algorithm 2. In stage 1, we use ANFIS with the hybrid learning rule to tune the 45 parameters of the FLC. The LSE method is used in the forward pass to tune the 25 parameters of the consequent part using (18) and the BP technique in (21) is used in the backward pass to tune the 20 parameters of the premise part. Then we start stage 2 in which we put the FLC obtained from stage 1 in parallel with the PD controller and obtain new input/output data pairs. We use these new data pairs to retune the FLC. The procedures of stage 2 are repeated once then we separate the PD controller and the tuned FLC is now ready to be used.

Fig. 6 and Fig. 7 show the tuned MFs for the inputs δ and $\dot{\delta}$ using the proposed IGBFLC and the proposed IANFIS, respectively. Table III and Table IV show the fuzzy decision table after tuning using the proposed IGBFLC and the proposed IANFIS, respectively. Table V shows the capture time using the optimal strategy, the PD controller, the genetic-based FLC (GBFLC), the ANFIS, the proposed IGBFLC, and the proposed IANFIS. From Table V we can see that the proposed IGBFLC and the proposed IANFIS outperform the GBFLC, the ANFIS, and the PD controller. The results of the proposed systems are the same and approach the optimal results. To check again the validity of our proposed systems we select positions for the evader different from those used in the training process. The results are shown in Table VI. The results in Table VI agree with those in Table V.

V. Conclusion

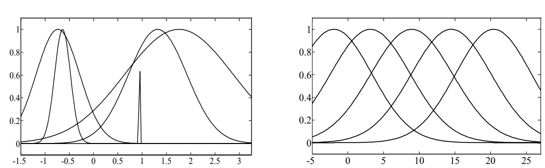
This paper introduces two different new hybrid systems. The proposed systems are used to tune the parameters of a FLC. The proposed systems are applied for a model of pursuit-evasion game. The performance of the pursuer is improved by using the proposed IGBFLC and the proposed IANFIS systems. We can see that the proposed IGBFLC and the proposed IANFIS outperform the GBFLC, the ANFIS, and the PD controller itself. By using the proposed systems the time of capture is reduced and it is extremely near to that of the optimal strategy. The proposed systems can be used when the optimal strategy is unknown or can not be determined.

TABLE I
Fuzzy decision table before tuning

δ \ $\dot{\delta}$	NB	NS	Z	PS	PB
NB	-0.8	-0.6	-0.4	-0.2	0.0
NS	-0.6	-0.4	-0.2	0.0	0.2
Z	-0.4	-0.2	0.0	0.2	0.4
PS	-0.2	0.0	0.2	0.4	0.6
PB	0.0	0.2	0.4	0.6	0.8

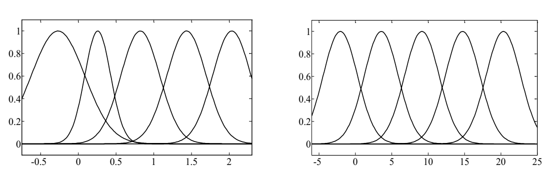
TABLE II
Values of GA parameters

Population size	Number of iterations	Crossover probability	Mutation probability
80	100	0.15	0.10



(a) The input δ (b) The input $\dot{\delta}$

Fig. 6. Tuned MFs for the inputs using the proposed IGBFLC



(a) The input δ (b) The input $\dot{\delta}$

Fig. 7. Tuned MFs for the inputs using the proposed IANFIS

TABLE III
FUZZY DECISION TABLE AFTER TUNING USING THE PROPOSED IGBFLC

δ \ $\dot{\delta}$	NB	NS	Z	PS	PB
NB	-0.9279	-0.8289	-0.6959	0.2914	0.4784
NS	-0.0816	-0.0867	-0.1114	-0.0681	0.2129
Z	-0.1767	0.0169	0.5664	-0.0474	0.6246
PS	0.6362	0.1725	0.7642	0.4957	0.8390
PB	0.6552	0.5013	0.8510	0.6737	0.8532

TABLE IV
FUZZY DECISION TABLE AFTER TUNING USING THE PROPOSED IANFIS

δ \ $\dot{\delta}$	NB	NS	Z	PS	PB
NP	-0.6442	0.5204	3.3461	0.6967	0.6198
NS	1.0458	-0.2798	-6.7812	0.3023	0.2690
Z	0.5954	0.3267	0.0487	0.0058	0.0051
PS	0.2834	1.2875	0.0017	0.0	0.0
PB	0.8922	-0.6527	-0.0021	0.0	0.0

TABLE V
CAPTURE TIME, IN SECONDS, USING THE DIFFERENT TECHNIQUES

Evader position	**(3,3)**	**(-2,4)**	**(2,5)**	**(-1,5)**
Optimal strategy	7.0	8.7	9.2	9.3
PD controller	18.8	21.5	21.3	22.0
GBFLC	19.2	23.0	21.7	22.2
ANFIS	19.1	21.7	21.6	22.0
Proposed IGBFLC	7.5	10.1	10.0	10.5
Proposed IANFIS	7.5	10.1	10.0	10.5

TABLE VI
CAPTURE TIME, IN SECONDS, USING DIFFERENT INITIAL POSITIONS FOR THE EVADER

Evader position	(1,6)	(-3,5)	(4,-2)	(7,1)
Optimal strategy	10.7	11.5	7.6	13.0
PD controller	23.1	24.3	19.2	24.4
Proposed IGBFLC	11.6	12.9	7.9	13.1
Proposed IANFIS	11.6	12.9	7.9	13.1

REFERENCES

[1] T. L. Seng, M. B. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernatics*, vol. 29, pp. 226–236, Apr. 1999.

[2] S. Labiod and T. M. Guerra, "Adaptive fuzzy control of a class of SISO nonaffine nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, no. 10, pp. 1126–1137, 2007.

[3] H. K. Lam and F. H. F. Leung, "Fuzzy controller with stability and performance rules for nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, pp. 147–163, 2007.

[4] M. Mucientes, D. L. Moreno, A. Bugarin, and S. Barro, "Design of a fuzzy controller in mobile robotics using genetic algorithms," *Applied Soft Computing*, vol. 7, no. 2, pp. 540–546, 2007.

[5] F. Hoffmann, "Evolutionary algorithms for fuzzy control system design," in *Proceedings of the IEEE*, vol. 89, pp. 1318–1333, Sep. 2001.

[6] C. J. Lin and Y. J. Xu, "Efficient reinforcement learning through dynamic symbiotic evolution for TSK-type fuzzy controller design," *International Journal of General Systems*, vol. 34, pp. 559–578, Oct. 2005.

[7] L. X. Wang, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 1414–1427, Nov./Dec. 1992.

[8] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *International Journal of Approximate Reasoning*, vol. 12, pp. 299–315, 1995.

[9] A. Lekova, L. Mikhailov, D. Boyadjiev, and A. Nabout, "Redundant fuzzy rules exclusion by genetic algorithms," *Fuzzy Sets and Systems*, vol. 100, pp. 235–243, 1998.

[10] L. Ming, G. Zailin, and Y. Shuzi, "Mobile robot fuzzy control optimization using genetic algorithm," *Artificial Intelligence in Engineering*, vol. 10, no. 4, pp. 293–298, 1996.

[11] S. F. Desouky and H. M. Schwartz, "Genetic based fuzzy logic controller for a wall-following mobile robot," in *2009 American Control Conference*, to appear.

[12] E. Entchev and L. Yang, "Application of adaptive neuro-fuzzy inference system techniques and artificial neural networks to predict solid oxide fuel cell performance in residential microgeneration installation," *Journal of Power Sources*, vol. 170, pp. 122–129, 2007.

[13] X. J. Wu, X. J. Zhu, G. Y. Cao, and H. Y. Tu, "Nonlinear modeling of a SOFC stack based on ANFIS identification," *Simulation Modelling Practice and Theory*, vol. 16, pp. 399–409, Apr. 2008.

[14] E. R. Nucci, R. G. Silva, V. R. Souza, R. L. C. Giordano, R. C. Giordano, and A. J. G. Cruz, "Comparing the performance of multilayer perceptrons networks and neuro-fuzzy systems for on-line inference of bacillus megaterium cellular concentrations," *Bioprocess and biosystems engineering*, vol. 30, pp. 429–438, 2007.

[15] F. Daldaban, N. Ustkoyuncu, and K. Guney, "Phase inductance estimation for switched reluctance motor using adaptive neuro-fuzzy inference system," *Energy Conversion and Management*, vol. 47, pp. 485–493, 2006.

[16] I. T. K. Güney, "Computation of association probabilities for single target tracking with the use of adaptive neuro-fuzzy inference system," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 13, no. 1, pp. 105–118, 2005.

[17] J. Baltes and Y. J. Park, "Comparison of several machine learning techniques in pursuit-evasion games," in *RoboCup-01: Robot Soccer World Cup V*, (New York), Springer, 2002.

[18] J. Ge, L. Tang, J. Reimann, and G. Vachtsevanos, "Hierarchical decomposition approach for pursuit-evasion differential game with multiple players," in *2006 IEEE Aerospace Conference*, (Big Sky, MT), Mar. 2006.

[19] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[20] S. H. Lim, T. Furukawa, G. Dissanayake, and H. F. D. Whyte, "A time-optimal control strategy for pursuit-evasion games problems," in *International Conference on Robotics and Automation*, (New Orleans, LA), Apr. 2004.

[21] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[22] D. E. Goldberg, *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading: Addison-Wesley, 1989.

[23] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, no. 2, pp. 26–33, 1991.

[24] T. L. Seng, M. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithms with an application to a coupled-tank liquid-level control system," *International Journal of Engineering Applications on Artificial Intelligence*, vol. 11, pp. 517–529, Sep. 1998.

[25] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 665–685, May/Jun. 1993.

[26] L. Zhao and F. Y. Wang, "The design of self-organizing fuzzy neural networks based on GA-ECPSO and MBP," in *IEEE International Conference on Systems, Man, and Cybernetics, ISIC*, (Montreal, QC, Canada), pp. 1618–1623, Oct. 2007.

[27] L. Zhao and F. Y. Wang, "Design for self-organizing fuzzy neural networks using a novel hybrid learning algorithm," in *IEEE Congress on Evolutionary Computation, CEC*, (Singapore), pp. 2972–2979, Sep. 2007.

[28] Y. He, Y. Zhang, and L. Xiang, "Study of application model on BP neural network optimized by fuzzy clustering," in *MICAI*, pp. 712–720, 2005.

[29] M. Negnevitsky, "Multi-layer neural networks with improved learning algorithms," in *Digital Imaging Computing: Techniques and Applications, DICTA*, (Cairns, Australia), pp. 1–6, Dec. 2005.

[30] C. H. Cai, D. Du, and Z. Y. Liu, "Battery state-of-charge (SOC) estimation using adaptive neuro-fuzzy inference system (ANFIS)," in *12th IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1068–1073, May 2003.