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

An Investigation of Methods of Parameter Tuning For Q-Learning Fuzzy Inference System

Ahmad A. Al-Talabi*† and Howard M. Schwartz*
*Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada
Email: ahmadaltalabi@cmail.carleton.ca, Howard.Schwartz@sce.carleton.ca
† Mechatronics Engineering Department, Al-Khwarizmi College of Engineering, Baghdad University,
Baghdad, Iraq

***Abstract*—This paper investigates four methods of implementing a Q-Learning Fuzzy Inference System(QFIS) algorithm to autonomously tune the parameters of a fuzzy inference system. We use an actor-critique structure and we simulate mobile robots playing the differential form of the pursuit evasion game. Both the critique and the actor are fuzzy inference systems. The four methods come from the fact whether it is necessary to tune all the parameters (i.e. all the premise and the consequent parameters) of the critique and the actor or just tune their consequent parameters. The four methods are applied to three versions of the pursuit evasion games. In the first version just the pursuer is learning. In the second version, the evader uses its higher maneuverability and plays intelligently against a self-learning pursuer. In the final version, both the pursuer and the evader are learning. We evaluate which parameters are best to tune and which parameters have little impact on the performance.**

I. Introduction

Reinforcement learning (RL) represents one of the most widely used approaches to learn through interaction with the environment. In RL, the Q-learning algorithm [1], [2] is normally used to estimate the action-value function. The Q-learning algorithm is a tabular method that uses discrete state and action spaces. So, it can not be used directly with pursuit-evasion differential games. Hence, to solve this problem, it is possible to discretize the state and action spaces such that the resulting Q-table is not too large. However, for real-world problems the resulting Q-table will be large. Furthermore, the discretization of the state and the action space is not an easy task [3]. In order to avoid the discretization process, one could use an approximation method to deal directly with the state and the action spaces that are continuous. Therefore, one can use fuzzy logic to generalize the state and action spaces.

Fuzzy logic control (FLC) is a good choice for dealing with processes that are ill-defined and/or involve uncertainty or continuous change. Moreover, it is well known that fuzzy inference systems are widely used as function approximators [4], [5]. Reinforcement fuzzy learning methods have recently been proposed for the problem of learning in differential games [5]–[9] . In [5], only the consequent parameters of the FLC and fuzzy inference system (FIS) are tuned using a fuzzy actor-critic learning algorithm. A FIS is used as an approximation to the value function, V(s). Furthermore, it is supposed that only the pursuer learns its behaviour strategy while the evader plays an optimal control strategy [5]. In [6]–[8], a Q-learning fuzzy inference system (QFIS) is applied to the pursuit-evasion differential game. All the premise and consequent parameters of the FIS and the FLC are tuned. In addition the FIS is used as an approximation to the action-value function, Q(s,a). In [9], fuzzy actor-critic learning is applied to the guarding territory differential game. In this learning technique, the consequent parameters are tuned to allow the defender to learn its Nash equilibrium strategy.

In the previous work, the researchers did not investigate which parameters were best to tune. Hence, we need to know whether it is necessary to tune all the parameters of the FIS and the FLC or just tune their consequent parameters. As we know, it would be computationally efficient to just tune the consequent parameters. But, would there be a significant loss in performance measures if only a subset of the available parameters were tuned?

The organization of this paper is as follows: In Section II, we discuss the problem statement, Section III describes the pursuit-evasion game and its model. The structure of the FLC is presented in Section IV. RL is presented in Section V. The QFIS is described in Section VI. In Section VII, the simulation results are presented. Finally, conclusions and guidelines are given in Section VIII.

II. Problem Statement

In this paper, we will investigate methods of parameter tuning for the QFIS algorithm used in [8]. Four possible methods of parameter tuning are taken into consideration. The four methods come from the fact whether it is necessary to tune all the parameters (i.e. all the premise and the consequent parameters) of the FIS(i.e. the critique) and the FLC(i.e. the actor) or just tune their consequent parameters as explained in Table I. Three pursuit-evasion games are considered [8]: in the first game the pursuer self learns its control strategy while the evader uses a deterministic strategy which is to run away along the line of sight (i.e. we define this as the default control strategy). In the second game, the evader uses its higher maneuverability and plays intelligently against a self-learning pursuer, and in the final game, both players interact with each other in order to self-learn their control strategies.

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

TABLE I. METHODS OF PARAMETER TUNING

	The tuned parameters	
	for FLC	for FIS
Method(1)	The consequent parameters	The consequent parameters
Method(2)	The consequent parameters	All parameters
Method(3)	All parameters	The consequent parameters
Method(4)	All parameters	All parameters

III. PURSUIT-EVASION GAME

The application that we will use for this study is the pursuit-evasion differential game [10]. In the pursuit-evasion game there are one or several pursuers that attempt to capture one or several evaders in minimal time while the evaders try to escape or to maximize the capturing time [10]. Hence, this problem can be considered as an optimization problem with conflict objectives [11]. In the pursuit-evasion game each player should learn the best action to take at each instant of time to adapt to an uncertain or changing environment. Fig. 1 shows the pursuit-evasion model with its parameters.

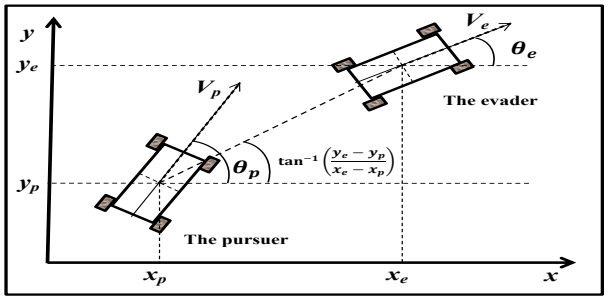

Fig. 1. The pursuit evader model

The dynamic equations that describe the motions of the pursuer and the evader robots in this game are [8], [12]

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

where i is "e" for the evader and is "p" for the pursuer. Also, (x_i, y_i), V_i, θ_i, L_i, and u_i refer to the position, the velocity, the orientation, the wheelbase, and the steering angle respectively. The steering angle is bounded and is given by $-u_{i_{max}} \le u_i \le u_{i_{max}}$, where $u_{i_{max}}$ is the maximum steering angle. The minimum turning radius of each robot is calculated from

$$Rd_{i_{min}} = \frac{L_i}{\tan(u_{i_{max}})} \tag{2}$$

We assume that the pursuer is faster than the evader ($V_p > V_e$) and at the same time the evader is more maneuverable than the pursuer ($u_{p_{max}} < u_{e_{max}}$). As in [8], we define the default control strategy for this game as

$$u_i = \begin{cases} -u_{i_{max}} & : \quad \delta_i < -u_{i_{max}} \\ \delta_i & : \quad -u_{i_{max}} \le \delta_i \le u_{i_{max}} \\ u_{i_{max}} & : \quad \delta_i > u_{i_{max}} \end{cases} \tag{3}$$

where δ_i represents the angle difference and is given by

$$\delta_i = \tan^{-1}\left(\frac{y_e - y_p}{x_e - x_p}\right) - \theta_i \tag{4}$$

This control strategy allows the player to run away along the line of sight. The distance between the pursuer and the evader at time t is given by

$$D(t) = \sqrt{(x_e(t) - x_p(t))^2 + (y_e(t) - y_p(t))^2} \tag{5}$$

and the capture occurs when this distance is less than a certain value , ℓ, which is called the capture radius.

IV. FUZZY LOGIC CONTROLLER STRUCTURE

We used a FLC with two inputs and one output [8]. The two inputs are the angle difference, δ_i, and its derivative, $\dot{\delta}_i$, and the output is the steering angle u_i. The fuzzy system is implemented using zero-order Takagi-Sugeno (TS) rules with constant consequents [13]. It consists of linguistic rules in the form

$$\begin{aligned} R_l : IF\ z_1\ is\ A_1^l\ and\ \ldots\ and\ z_N\ is\ A_N^l \\ THEN\ f_l = K_l \end{aligned} \tag{6}$$

where $z_1\ is\ A_1^l\ and\ \ldots\ and\ z_N\ is\ A_N^l$ is called the premise or the antecedent part, and $f_l = K_l$ is called the consequence or the conclusion part. Also, z_i is the i^{th} input variable, N is the number of input variables, A_i^l is the Fuzzy set of the input z_i in rule l, K_l is the consequent parameter of the fuzzy rule l, L is the number of the Fuzzy rules, and f_l is the output of rule l. In our problem we have two inputs z_1, and z_2 which represent δ_i and $\dot{\delta}_i$ respectively. Each one of them has three Gaussian membership functions (MFs). The three Gaussian MFs have the following linguistic values: P (positive), Z (zero), and N (negative). The Gaussian MF takes the following form

$$\mu(z) = \exp\left(-\left(\frac{z-m}{\sigma}\right)^2\right) \tag{7}$$

where m and σ are the mean and the standard deviation respectively and represent the tunable input parameters. So, we have $2 \times 6 = 12$ parameters in all MFs which are called the premise parameters. The mean and the standard deviation for the j^{th} MF of the input z_i can be denoted by m_{ij} and σ_{ij} respectively. Moreover, the number of rules depends on the number of inputs and their corresponding MFs. In our case, we have 2 inputs δ_i and $\dot{\delta}_i$ and each input has 3 MFs. Hence, we need to build $3 \times 3 = 9$ rules, and each rule contains one consequent parameter, "K_l". The fuzzy rules can be constructed using the fuzzy decision table as shown in Table II.

As a result, we have 12+9=21 parameters that may be tuned during the learning phase. The Fuzzy output u_i is defuzzified into a crisp output using the following weighted average defuzzification method

$$u_i = \frac{\sum_{l=1}^{L}((\prod_{i=1}^{N}\mu^{A_i^l}(z_i))K_l}{\sum_{l=1}^{L}(\prod_{i=1}^{N}\mu^{A_i^l}(z_i))} \tag{8}$$

Fig. 2 and Table III show MFs of the pursuer and the evader before learning and their fuzzy decision table. The reason for choosing these initial settings is to prevent the pursuer from

catching the evader at the beginning of the pursuit evasion game. It is possible to use different initial settings and this will certainly affect the period of the learning process.

TABLE II. FUZZY DECISION TABLE

		$\dot{\delta}_i$		
		N	**Z**	**P**
δ_i	**N**	K_1	K_2	K_3
	Z	K_4	K_5	K_6
	P	K_7	K_8	K_9

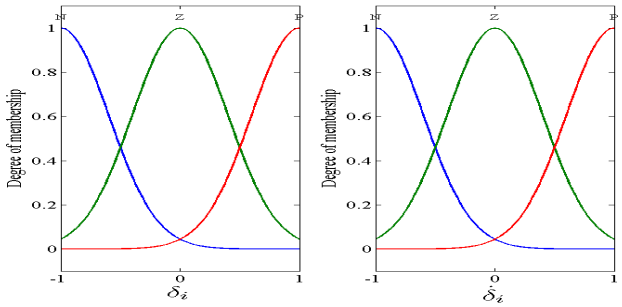

Fig. 2. Initial MFs of the pursuer and the evader

TABLE III. FUZZY DECISION TABLE OF THE PURSUER AND THE EVADER BEFORE LEARNING

		$\dot{\delta}_i$		
		N	**Z**	**P**
δ_i	**N**	-0.50	-0.25	0.00
	Z	-0.25	0.00	0.25
	P	0.00	0.25	0.50

V. Reinforcement Learning

The main concept of RL is for the agent to learn how to achieve a specific goal by interacting with its environment. The agent is the learner and the decision-maker and everything outside the agent and that interacts with it, is called the environment. The interaction between the agent and the environment is done in a simplified manner and is shown in Fig. 3 [1]. The agent selects actions and the environment responds to those actions and produces new situations to the agent. Furthermore, the environment gives rise to numerical values which are called rewards, that should be maximized by the agent over time. According to

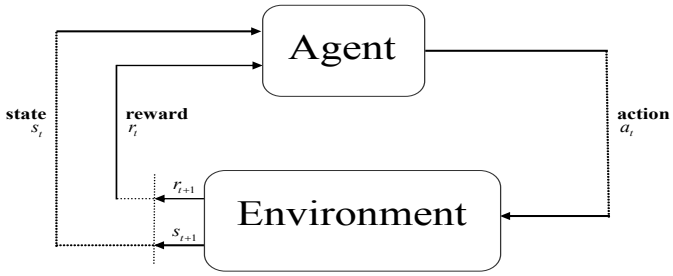

Fig. 3. Agent-environment interaction in reinforcement learning

the concept of discounting, the agent tries to select actions that maximize the sum of the received discounted rewards into the future. The return reward, R_t, is given by:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots = \sum_{k=0}^{\tau} \gamma^k r_{t+k+1} \tag{9}$$

where γ is a parameter, $(0 < \gamma \leq 1)$. The term γ is the discount factor and τ is the terminal time. One of the most popular types of RL is Q-learning. In Q-learning, the action value function under policy π , $Q^{\pi}(s,a)$, is evaluated to get the best expected return reward and is given by:

$$Q^{\pi}(s,a) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{k+t+1} | s_t = s, a_t = a\right) \tag{10}$$

The main task of RL is to find the appropriate policy that maximizes the agents reward over the long run. In RL, the reward function selection process is a task dependent problem. The right choice of this function enables the agent to update its value function accurately. For the pursuit-evasion game, the main task is to enable the pursuer to catch the evader in minimum time. So, the right choice for this function is defined by Desouky et al. [8] and is given by

$$r_{t+1} = \Delta D(t)/\Delta D_{max} \tag{11}$$

where

$$\Delta D(t) = D(t) - D(t+1) \tag{12}$$

and

$$\Delta D_{max} = (V_p + V_e)T \tag{13}$$

where T represents the sampling time.

VI. Q-Learning Fuzzy Inference System QFIS

Desouky et al. [8] proposed an actor-critique learning technique called the Q(λ)-learning fuzzy inference system (QLFIS) and applied it to the problem of the pursuit evasion game.[1] The QLFIS tunes all the parameters of the FIS and the FLC. The FIS is used to approximate the action-value function, $Q(s,a)$. The learning system proposed by Desouky et al. [8] is shown in Fig. 4. The FLC determines the control signal u. For exploration, a white Gaussian noise with zero mean and standard deviation σ_n is added to the signal u to generate the control signal u_c.

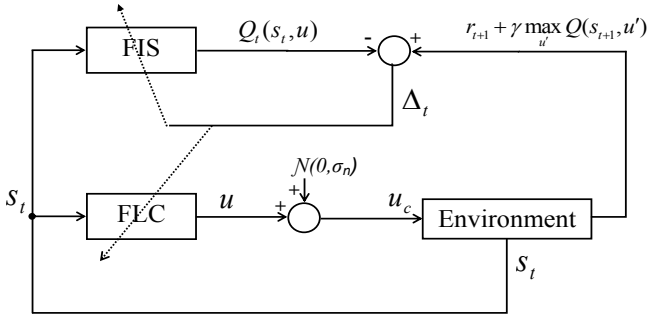

Fig. 4. Q-learning fuzzy inference system(QFIS)

[1]Desouky et al. [8] used eligibility traces defined by λ, as their RL algorithm. It has been discovered that the eligibility trace had little advantage in this application and as such we are not using it in our subsequent work.

It is well known that the update rule for Q-learning is

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha \triangle_t \tag{14}$$

where $(0 < \alpha \leq 1)$ is the learning rate parameter, and $\triangle_t$ is the temporal difference error. The term $\triangle_t$ is defined by

$$\triangle_t = r_{t+1} + \gamma \max_{\acute{a} \in \mathcal{A}(s_{t+1})} Q_t(s_{t+1},\acute{a}) - Q_t(s_t,a_t) \tag{15}$$

where $\mathcal{A}(s_{t+1})$ refers to the set of actions available in state s_{t+1} and $\acute{a}$ is the action that gives us the maximum action-value function. We assumed $Q_t(s_{t+1},a_{t+1}) \approx \max_{\acute{a} \in \mathcal{A}(s_{t+1})} Q_t(s_{t+1},\acute{a})$ [3]. Let ϕ represent the parameter vector of the FLC and the FIS. Where ϕ is given by

$$\phi = \begin{pmatrix} \sigma & m & K \end{pmatrix}^T \tag{16}$$

The parameter vector ϕ of the FIS and the FLC are updated according to the following gradient based formulas [8]

$$\phi_Q(t+1) = \phi_Q(t) + \eta \Delta_t \frac{\partial Q_t(s_t,u_t)}{\partial \phi_Q} \tag{17}$$

$$\phi_u(t+1) = \phi_u(t) + \xi \Delta_t (\frac{u_c - u}{\sigma_n}) \frac{\partial u}{\partial \phi_u} \tag{18}$$

where η and ξ are the learning rate of the FIS and the FLC respectively and they are defined by [8]

$$\eta = 0.1 - 0.09 \left(\frac{i}{\text{Max. Episodes}} \right) \tag{19}$$

$$\xi = 0.1\eta \tag{20}$$

where i is the current episode. The terms $\frac{\partial Q_t(s_t,u_t)}{\partial \phi_Q}$ and $\frac{\partial u}{\partial \phi_u}$ are given by

$$\frac{\partial Q_t(s_t,u_t)}{\partial \phi_Q} = \begin{pmatrix} \frac{\partial Q_t(s_t,u_t)}{\partial \sigma_{ij}} \\ \frac{\partial Q_t(s_t,u_t)}{\partial m_{ij}} \\ \frac{\partial Q_t(s_t,u_t)}{\partial K_l} \end{pmatrix} = \begin{pmatrix} \frac{(\acute{K} - Q_t(s_t,u_t))}{\sum_{l=1}^{9} \omega_l} \acute{\omega}^T \frac{2(z_i - m_{ij})^2}{(\sigma_{ij})^3} \\ \frac{(\acute{K} - Q_t(s_t,u_t))}{\sum_{l=1}^{9} \omega_l} \acute{\omega}^T \frac{2(z_i - m_{ij})}{(\sigma_{ij})^2} \\ \overline{\omega}_l \end{pmatrix} \tag{21}$$

$$\frac{\partial u}{\partial \phi_u} = \begin{pmatrix} \frac{\partial u}{\partial \sigma_{ij}} \\ \frac{\partial u}{\partial m_{ij}} \\ \frac{\partial u}{\partial K_l} \end{pmatrix} = \begin{pmatrix} \frac{(\acute{K} - u)}{\sum_{l=1}^{9} \omega_l} \acute{\omega}^T \frac{2(z_i - m_{ij})^2}{(\sigma_{ij})^3} \\ \frac{(\acute{K} - u)}{\sum_{l=1}^{9} \omega_l} \acute{\omega}^T \frac{2(z_i - m_{ij})}{(\sigma_{ij})^2} \\ \overline{\omega}_l \end{pmatrix} \tag{22}$$

where ω_l and $\overline{\omega}_l$ represent the firing strength and the normalized firing strength of the rule l which are calculated from [8]

$$\omega_l = \prod_{i=1}^{2} \exp\left(-(\frac{x_i - m_{ij}^l}{\sigma_{ij}^l})^2 \right) \tag{23}$$

$$\overline{\omega}_l = \frac{\omega_l}{\sum_{l=1}^{9} \omega_l} \tag{24}$$

The two terms, $\acute{K}$ and $\acute{\omega}$ are two vectors containing the consequence and the strength of certain rules respectively. As an example, the parameter σ_{23} represents the standard deviation for the third MF of the second input z_2 and it appears in the rules R_3, R_6, and R_9. So, we can calculate $\frac{\partial Q_t(s_t,u_t)}{\partial \sigma_{23}}$ from (21) with $\acute{K} = [K_3 \quad K_6 \quad K_9]$ and $\acute{\omega} = [\omega_3 \quad \omega_6 \quad \omega_9]$. The QFIS learning algorithm is given in Algorithm 1.

Algorithm 1 Learning in the QFIS.

1) Initialize the premise and the consequent parameters of the FLC as shown in Fig. 2 and Table III, respectively.
2) Initialize the premise parameters of the FIS with the same values as those of the FLC and initialize the consequent parameters to zeros.
3) Set $\gamma \leftarrow 0.95$, and $\sigma_n \leftarrow 0.08$.
4) **For** each episode (game)
   a) Calculate η from (19) and calculate ξ from (20).
   b) Initialize the position of the pursuer, (x_p, y_p) to $(0,0)$.
   c) Initialize the position of the evader, (x_e, y_e), randomly.
   d) Calculate the initial state, $s = (\delta_i, \dot{\delta}_i)$, from (4).
   e) Calculate the output of the FLC, u, from (8).
   f) **For** each step (play) **Do**
      i) Calculate the output $u_c = u + \mathcal{N}(0, \sigma_n)$.
      ii) Calculate the output of the FIS, $Q(s,u)$, from (8).
      iii) Run the game for the current step and observe the next state s_{t+1}.
      iv) Get the reward, r, from (11).
      v) From (8), calculate $Q_(s_{t+1}, u')$.
      vi) Calculate the TD-error, Δ_t, from (15).
      vii) Calculate the gradient for the premise and the consequent parameters of the FIS and the FLC from (21),and (22) respectively.
      viii) Update the parameters of the FIS from (17).
      ix) Update the parameters of the FLC from (18).
      x) Set $s_t \leftarrow s_{t+1}$ and $u \leftarrow u'$.
   g) **end for**
5) **end for**

VII. Computer Simulation

For the purpose of simulation, we choose the same values used by Desouky et al. [8] unless stated otherwise. It is assumed that the pursuer is faster than the evader with $V_p = 2m/s$ and $V_e = 1m/s$ and the evader is more maneuverable than the pursuer with -1$\leq -u_{e_{max}} \leq$ 1 and -0.5$\leq -u_{p_{max}} \leq$ 0.5. The wheelbases of the pursuer and the evader are the same and equal to $0.3m$. In each episode, the pursuer's motion is started from the origin with an initial orientation $\theta_p = 0$ while the evader's motion is chosen randomly from a set of 64 different positions with $\theta_e = 0$. The selected capture radius is $\ell = 0.1m$ except for the second game where $\ell = 0.05m$ and the sample time is $T = 0.1sec$. The number of episodes/games is chosen to be 1000, and the number of plays in each game is 600. Hence, the game terminates when the time exceeds $60sec$ or when the pursuer captures the evader.

A. *Evader follows a default control strategy*

In this game, we assume that the evader plays its default control strategy as defined by (3) and (4). Also, we assumed that the pursuer does not have any information about its default strategy or the evader's strategy. The goal is to make the pursuer self learn its control strategy by interacting with the evader. Furthermore, to find the best methods of parameters tuning for this game, the four methods that have been discussed in Section II are implemented. Their results are compared with the default control strategy. The capture times for different initial evader positions using the default control strategy and these methods are given in Table IV. The pursuer-evader paths using these methods compared with the default control strategy are shown in Fig. 5, Fig. 6, Fig. 7 and Fig. 8 (i.e. for the evader position (-6,7)). From Table IV and Fig. 5 and Fig. 6, it is clear that the performance of the first two methods are the same. Also, their capture times differ slightly from the default control strategy. On the other hand, from Table IV and Fig. 7 and Fig. 8 we can say that the performance of the last two methods are the same and approaches the performance of the default control strategy both on the capture times and on the pursuer-evader paths. Moreover, Fig. 8 shows that the pursuer-evader paths using Method(4) and the default control strategy are very close, so it is very difficult to distinguish between them. So, we can conclude that the performance of the learning algorithm is slightly effected by changing the method of tuning for the FIS but it is significantly effected by changing the method of tuning for the FLC. Furthermore, the results show that the pursuer is able to learn its control strategy in all methods but the last two methods outperform the first two methods. As an example, Fig. 9 and Table V show the MFs and the fuzzy decision table of the FLC for the pursuer after learning using Method(3).

TABLE IV. CAPTURE TIME(SEC) FOR DIFFERENT EVADER INITIAL POSITIONS

	Evader initial position				
	(-6,7)	(-7,-7)	(2,4)	(3,-8)	(-4,5)
Default control strategy	9.6	10.4	4.5	8.5	6.8
QFIS(Method(1))	10.1	10.7	4.7	8.8	7.2
QFIS(Method(2))	10.0	10.6	4.7	8.7	7.1
QFIS(Method(3))	9.7	10.4	4.5	8.6	6.8
QFIS(Method(4))	9.6	10.4	4.5	8.5	6.8

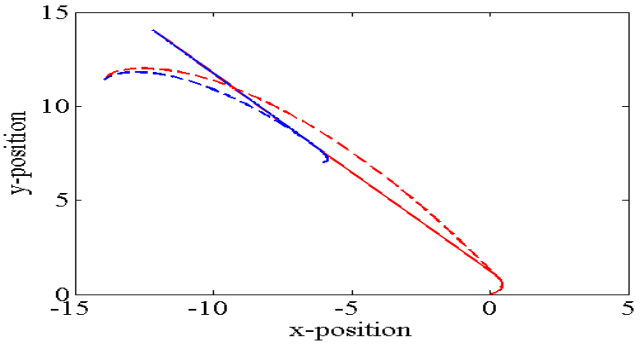

Fig. 5. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(1)) (dotted line)

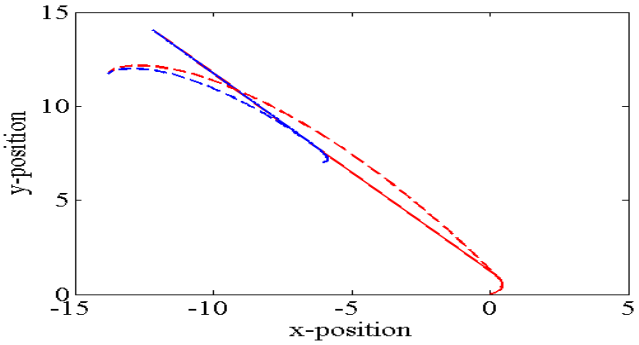

Fig. 6. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(2)) (dotted line)

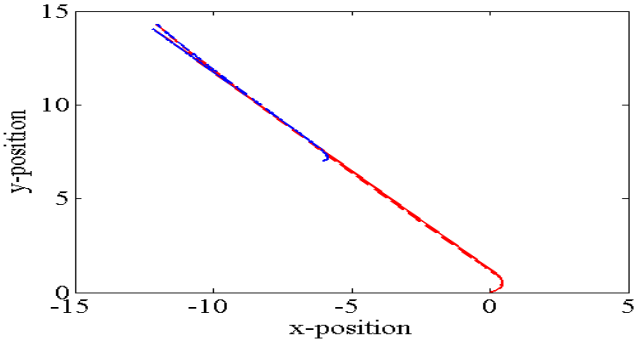

Fig. 7. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(3)) (dotted line)

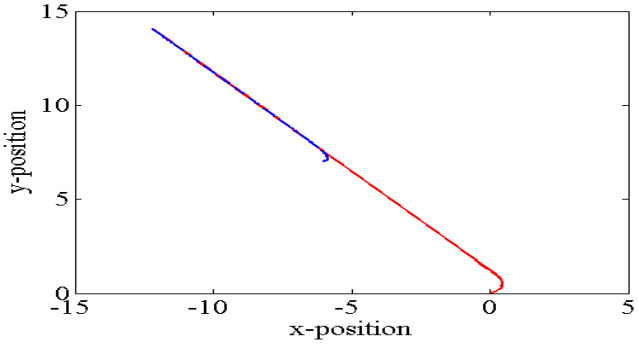

Fig. 8. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(4)) (dotted line)

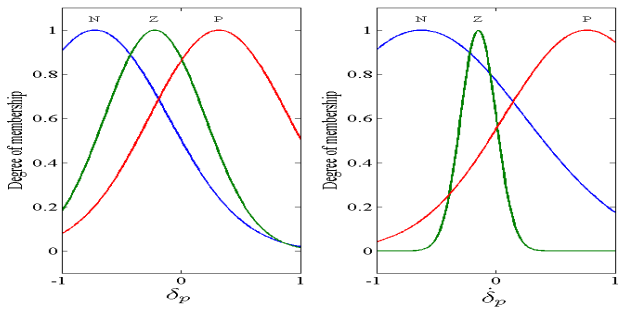

Fig. 9. Method(3): MFs of the FLC for the pursuer after learning

TABLE V. METHOD(3): FUZZY DECISION TABLE OF THE PURSUER AFTER LEARNING

		$\dot{\delta}_p$		
		N	**Z**	**P**
δ_p	**N**	-1.6967	-0.8306	-1.2808
	Z	-0.7253	-0.4021	-0.6387
	P	1.4108	0.8850	1.2287

B. *Evader using its higher maneuverability advantageously*

The second version of the pursuit-evasion game allows the evader to use its advantage of higher maneuverability. The dynamic equations that describe the motions of the pursuer and the evader robots are given by (1). In this game, the robot velocity, V_i, slows down in the turn and is defined by

$$V_i = v_i \cos(u_i) \tag{25}$$

where v_i represents the robot maximum velocity. Desouky et al. [8] modified the evader default strategy to allow the evader to use its higher maneuverability advantageously as follows:

1) If the evader is far enough from the pursuer (i.e $D(t)$ is greater than a certain value, d), then the evader will run away along the line of sight. So, the evader control strategy is

$$u_e = \tan^{-1}\left(\frac{y_e - y_p}{x_e - x_p}\right) - \theta_e \tag{26}$$

2) Otherwise the evader uses its advantage of higher maneuverability and takes the opposite direction of the pursuer. So, the evader control strategy is

$$u_e = (\theta_p + \pi) - \theta_e \tag{27}$$

where d is the minimum turning radius of the pursuer, $Rd_{p_{min}}$.

In order to find the best methods of parameter tuning for this game, the QFIS learning process is implemented for each method. The results are compared with the default control strategy of this game. The capture times for different initial evader positions using the default control strategy and the four tuning methods are given in Table VI. The pursuer-evader paths for these methods compared with the default control strategy are shown in Fig. 10, Fig. 11, Fig. 12 and Fig. 13 (i.e. for the evader position (-4,5)). From Table VI and Fig. 10 and Fig. 11, it is clear that the first two methods of parameter tuning are not good enough to get a good performance. There are significant differences in the capture times and the pursuer-evader paths are different from the default control strategy. It is clear that the pursuer does not learn well. On the other hand, from Table VI and Fig. 12 and Fig. 13 we can say that the performance of the third method is quite similar to the performance of the fourth method and outperforms the first two methods both on the capture times and on the pursuer-evader paths. So, we can conclude that the performance of the QFIS is slightly effected by changing the method of tuning of the FIS. But, the performance will be much better by changing the method of tuning for the FLC. The fuzzy decision table and the tuned MFs of the pursuer after learning using Method(3) are shown in Table VII and Fig. 14 respectively.

TABLE VI. CAPTURE TIME(SEC) FOR DIFFERENT EVADER INITIAL POSITIONS

	Evader initial position				
	(-6,7)	(-7,-7)	(2,4)	(3,-8)	(-4,5)
Default control strategy	18.6	11.9	4.3	10.2	10.2
QFIS(Method(1))	15.8	16.5	10.7	14.8	8.4
QFIS(Method(2))	15.3	16.2	10.2	14.5	8.4
QFIS(Method(3))	18.6	11.9	4.3	8.4	10.3
QFIS(Method(4))	18.6	11.9	4.3	8.4	10.2

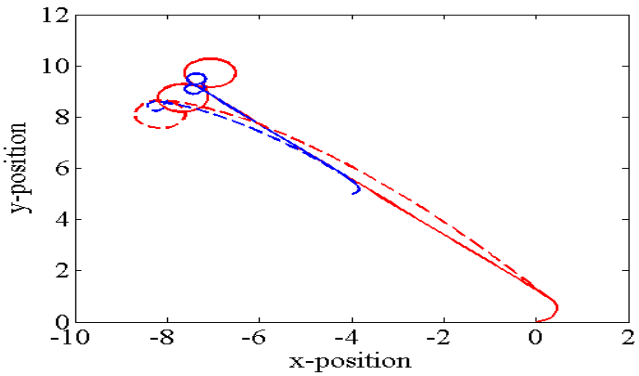

Fig. 10. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(1)) (dotted line)

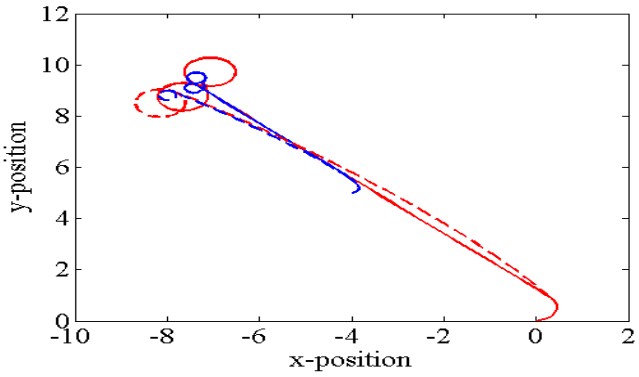

Fig. 11. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(2)) (dotted line)

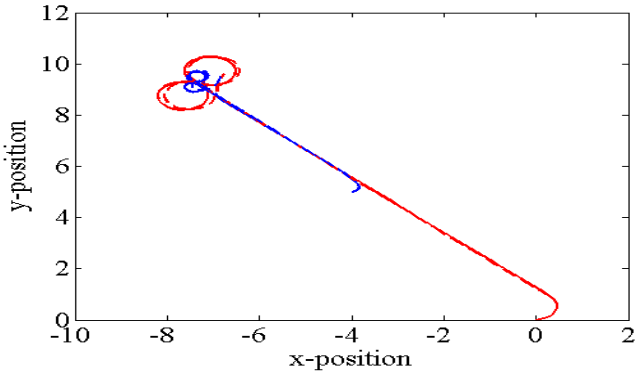

Fig. 12. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(3)) (dotted line)

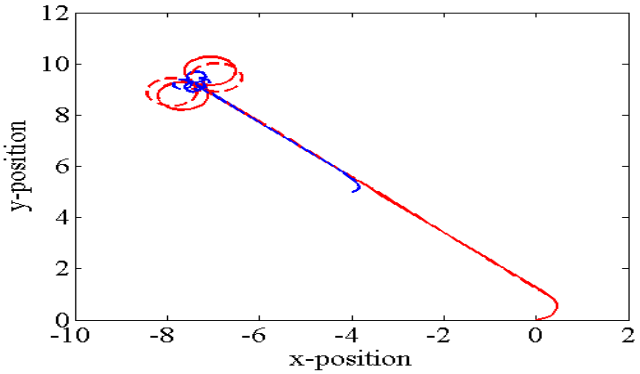

Fig. 13. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(4)) (dotted line)

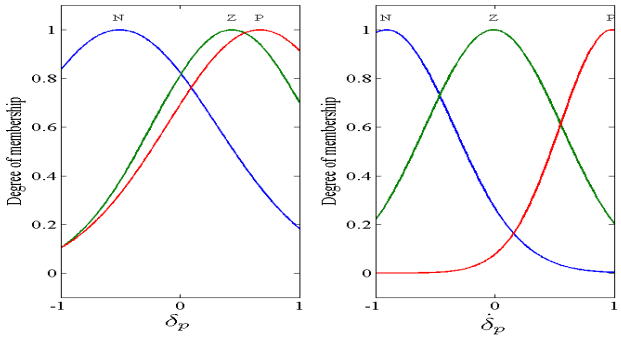

Fig. 14. Method(3): MFs of the FLC of the pursuer after learning

TABLE VII. METHOD(3): FUZZY DECISION TABLE OF THE PURSUER AFTER LEARNING

		$\dot{\delta}_p$		
		N	**Z**	**P**
δ_p	**N**	-0.9866	-0.6775	-1.0195
	Z	0.2777	0.1783	0.2399
	P	1.0779	0.7357	0.9164

C. Multi-robot learning

In this game, we assumed that each robot does not have any information about its default strategy or the other robot strategy. The goal is to make both players interact with each other in order to self-learn their control strategies simultaneously without using the advantage of higher maneuverability.

The capture times for different initial evader positions using the default control strategy and the four methods of parameter tuning are given in Table VIII. The pursuer-evader paths using these methods compared with the default control strategy are shown in Fig. 15, Fig. 16, Fig. 17, Fig. 18 (i.e. for the evader position (-6,7)). From Table VIII and Fig. 15 we can see that the first method of parameter tuning is not good enough for multi-robot learning to get the desired performance compared with the default control strategy. It is clear that the evader does not learn well and it gets captured too soon. As we see there are differences in the capture times and the pursuer-evader paths are completely different. Also, from Table VIII we can see that the capture times of the last three methods of parameter tuning are slightly different from that of the default control strategy. From Fig. 16 we can see that the pursuit-evader path is different from the path of the default control strategy. Furthermore, from Fig. 17 and Fig. 18 we can see that the pursuer-evader paths of the last two methods differ slightly from that of the default control strategy. As a result, we can say that tuning all the parameters of the FLC gives us the best performance regarding the capture time and the pursuer-evader path. The fuzzy decision tables of the pursuer and the evader after learning using Method(3) are given in Table IX and Table X. Fig. 19 shows the tuned MFs for both players after learning.

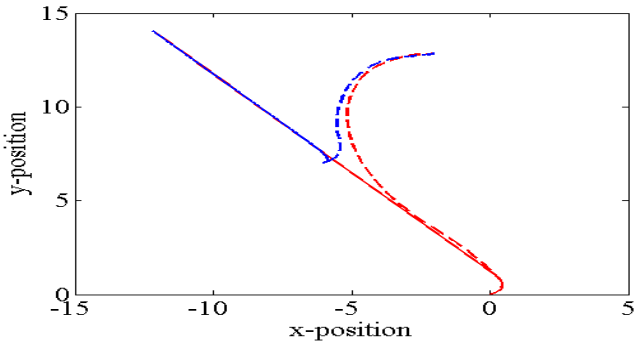

Fig. 15. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(1)) (dotted line)

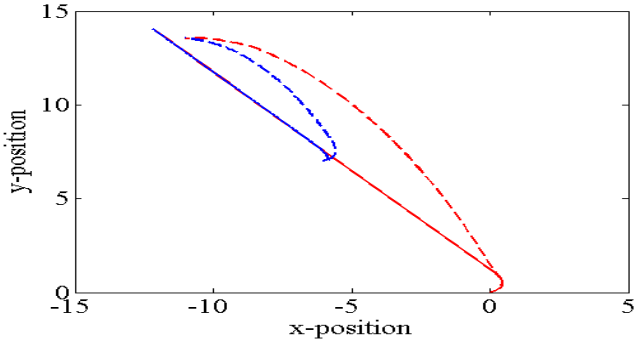

Fig. 16. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(2)) (dotted line)

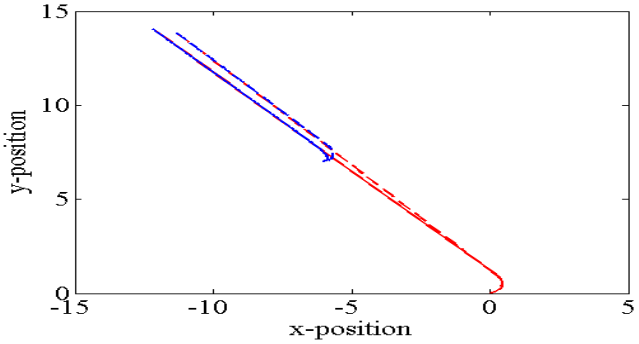

Fig. 17. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(3)) (dotted line)

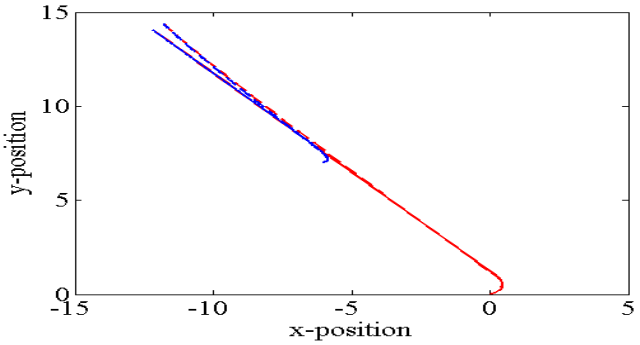

Fig. 18. The pursuit evader path using the default control strategies (solid line) versus QFIS(Method(4)) (dotted line)

TABLE VIII. CAPTURE TIME(SEC) FOR DIFFERENT EVADER INITIAL POSITIONS

	Evader initial position				
	(-6,7)	(-7,-7)	(2,4)	(3,-8)	(-4,5)
default control strategy	9.6	10.4	4.5	8.5	6.8
QLFIS(Method(1))	8.4	8.8	4.0	7.6	5.8
QLFIS(Method(2))	9.5	10.3	4.6	8.8	6.6
QLFIS(Method(3))	9.3	10.0	4.4	8.5	6.5
QLFIS(Method(4))	9.5	10.3	4.5	8.5	6.7

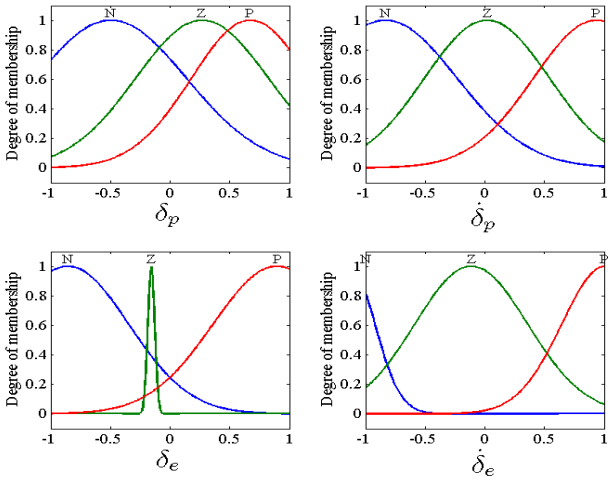

Fig. 19. Method(3): MFs of the FLC of the pursuer and the evader after learning

TABLE IX. METHOD(3):FUZZY DECISION TABLE OF THE PURSUER AFTER LEARNING

		$\dot{\delta}_p$		
		N	**Z**	**P**
δ_p	**N**	-0.8790	-0.7248	-0.9882
	Z	0.4463	0.4341	0.5159
	P	0.9057	0.3450	0.8912

TABLE X. METHOD(3):FUZZY DECISION TABLE OF THE EVADER AFTER LEARNING

		$\dot{\delta}_e$		
		N	**Z**	**P**
δ_e	**N**	-0.6914	-0.4082	-0.6211
	Z	-0.4277	-0.0994	0.4062
	P	0.4836	0.4710	0.6662

VIII. CONCLUSION

Four methods of parameter tuning for QFIS are applied to three pursuit-evasion games. The results show that the performance of the QFIS in each game depends on the parameter tuning method. In the first and second games, the results demonstrate that the performance of the learning algorithm is slightly effected by changing the method of tuning for the FIS but it is significantly effected by changing the method of tuning for the FLC. Furthermore, in the first game, it was found that the pursuer is able to learn its control strategy in all methods but the last two methods outperform the first two methods. This is because the first game is so simple. In the second game, it was found that the first two methods of parameter tuning are not good enough to get a good performance because the pursuer does not learn well. So, it was found that it is necessary to tune all the parameters of the FLC as in Method(3) and Method(4) to get best performance regarding the capture time and the pursuer-evader path. On the other hand, in the third game, the results show that the performance of the learning algorithm is effected by changing the method of tuning for both the FIS and the FLC. But, changing the method of tuning for the FLC as in Method(3) and Method(4) has significant impact on the performance. In order to reduce the computational complexity, it is better to use Method(1) in the first game and Method(3) in the second and the third game.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[2] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, 1989.

[3] X. Dai, C. K. Li, and A. B. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 285–293, Sep. 2005.

[4] T. Ross, *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, 2004.

[5] S. N. Givigi, H. M. Schwartz, and X. Lu, "A reinforcement learning adaptive fuzzy controller for differential games," *Journal of Intelligent and Robotic Systems*, vol. 59, pp. 3–30, 2010.

[6] S. F. Desouky and H. M. Schwartz, "Self-learning fuzzy logic controllers for pursuit-evasion differential games," *Robotics and Autonomous Systems*, vol. 59, pp. 22–33, 2011.

[7] B. A. Faiya and H. M. Schwartz, "Q(λ)-learning fuzzy controller for the homicidal chauffeur differential game," in *2012 20th Mediterranean Conference on Control and Automation (MED)*, 2012, pp. 247–252.

[8] S. F. Desouky and H. M. Schwartz, "Q(λ)-learning adaptive fuzzy logic controllers for pursuit-evasion differential games," *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 10, pp. 910–927, 2011.

[9] X. Lu, "Multi-agent reinforcement learning in games," Ph.D. dissertation, Carleton University, March 2012.

[10] R. Isaacs, *Differential Games*. John Wiley and Sons, 1965.

[11] V. Gesu, B. Lenzitt, G. Bosco, and D. Tegolo, "Comparison of different cooperation strategies in the prey-predator problem," in *The International Workshop on Computer Architecture for Machine Perception and Sensing*, 2006, pp. 108–112.

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

[13] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.