13

Genetic Reinforcement Learning Algorithms for On-line Fuzzy Inference System Tuning "Application to Mobile Robotic"

Abdelkrim Nemra and Hacene Rezine
Unit of Control, Robotic and Productic Laboratory
Polytechnical Military School
Algeria

1. Introduction

In the last decade, fuzzy logic has supplanted conventional technologies in some scientific applications and engineering systems especially in control systems, particularly the control of the mobile robots evolving (moving) in completely unknown environments. Fuzzy logic has the ability to express the ambiguity of human thinking and translate expert knowledge into computable numerical data. Also, for real-time applications, its relatively low computational complexity makes it a good candidate. A fuzzy system consists of a set of fuzzy if-then rules. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observation to express the knowledge of proper strategies.
Recently, many authors proved that it is possible to reproduce the operation of any standard continuous controller using fuzzy controller L. Jouffe, C. Watkins, P. Dayan Dongbing Gu, Huosheng Hu, Libor Spacek . However it is difficult for human experts to examine complex systems, then it isn't easy to design an optimized fuzzy controller.
Generally the performances of Fuzzy inference system (FIS) depend on the formulation of the rules, but also the numerical specification of all the linguistic terms used and an important number of choices is *given a priori,* also it is not always easy or possible to extract these data using human expert. These choices are carried with empirical methods, and then the design of the FIS can prove to be long and delicate vis-à-vis the important number of parameters to determine, and can lead then to a solution with poor performance. To cope with this difficulty, many researchers have been working to find learning algorithms for fuzzy system design. These automatic methods enable to extract information when the experts' *priori* knowledge is not available.
The most popular approach to design Fuzzy Logic Controller (FLC) may be a kind of supervised learning where the training data is available. However, in real applications, extraction of training data is not always easy and become impossible when the cost to obtain training data is expensive. For these problems, reinforcement learning is more suitable than supervised learning. In reinforcement learning, an agent receives from its environment a critic, called reinforcement, which can be thought of as a reward or a punishment. The objective then is to generate a policy maximizing on average the sum of the rewards in the course of time, starting from experiments (state, action, reward).

Source: Robotics, Automation and Control, Book edited by: Pavla Pecherková, Miroslav Flídr and Jindřich Duník,
ISBN 978-953-7619-18-3, pp. 494, October 2008, I-Tech, Vienna, Austria

In this chapter, we used the algorithm of reinforcement learning, Fuzzy Q-Learning (FQL) L. Jouffe, A. Souissi which allows the adaptation of apprentices of the type FIS (continuous states and actions), fuzzy Q-learning is applied to select the consequent action values of a fuzzy inference system. For these methods, the consequent value is selected from a predefined values set which is kept unchanged during learning, and if an improper value set is assigned, then the algorithm may fail. Also, the approach suggested called Fuzzy-Q-Learning Genetic Algorithm (FQLGA), is a hybrid method of Reinforcement Genetic combining FQL and genetic algorithms for on line optimization of the parametric characteristics of a FIS. In FQLGA we will tune free parameters (precondition and consequent part) by genetic algorithms (GAs) which is able to explore the space of solutions effectively. However, many times the priory knowledge about the FIS structure is not available, as a solution, the suggested approach called Dynamic Fuzzy Q-Learning Genetic Algorithm (DFQLGA), which is a hybrid learning method, this method combines the Dynamic Fuzzy Q-Learning algorithm (DFQL) Meng Joo Er, Chang Deng and the genetic algorithms to optimize the structural and parametric characteristics of the FIS, with out any priori knowledge, the interest of the latter (GA) is to explore the space of solutions effectively and permits the optimization of conclusions starting from a random initialization of the parameters.
This chapter is organized as follows. In Section II, overviews of Reinforcement learning, implementation and the limits of the Fuzzy-Q-Learning algorithm is described. The implementation and the limits of the Fuzzy-Q-Learning algorithm are introduced in Section III. Section IV describes the combination of Reinforcement Learning (RL) and genetic algorithm (GA) and the architecture of the proposed algorithm called Fuzzy-Q-Learning Genetic Algorithm (FQLGA). In section V we present the DFQL algorithm, followed by the DFQLGA algorithm in section VI. Section VII shows simulation and experimentation results of the proposed algorithms, on line learning of two elementary behaviours of mobile robot reactive navigation, "Go to Goal" and "Obstacles Avoidance" is presented with discussion. Finally, conclusions and prospects are drawn in Section VIII.

2. Reinforcement learning

As previously mentioned, there are two ways to learn either you are told what to do in different situations or you get credit or blame for doing good respectively bad things. The former is called supervised learning and the latter is called learning with a critic, of which reinforcement learning (RL) is the most prominent representative. The basic idea of RL is that agents learn behaviour through trial-and-error, and receive rewards for behaving in such a way that a goal is fulfilled.
Reinforcement signal, measures the utility of the exits suggested relative with the task to be achieved, the received reinforcement is the sanction (positive, negative or neutral) of behaviour: this signal states that it should be done without saying how to do it. The goal of reinforcement learning, is to find the behaviour most effective, i.e. to know, in each possible situation, which action is achieved to maximize the cumulated future rewards. Unfortunately the sum of rewards could be infinite for any policy. To solve this problem a discount factor is introduced.

$$R=\sum_{k\neq 0}^{\infty}\gamma^{k}r_{k} \tag{1}$$

Where $0 \leq \gamma \leq 1$ is the *discount factor*.

The idea of RL can be generalized into a model, in which there are two components: an agent that makes decisions and an environment in which the agent acts. For every time step, the agent perceives information from the environment about the current state, ***s***. The information perceived could be, for example, the positions of a physical agent, to simplify say the ***x*** and ***y*** coordinates. In every state, the agent takes an action u_t, which transits the agent to a new state. As mentioned before, when taking that action the agent receives a reward.
Formally the model can be written as follows; for every time step ***t*** the agent is in a state $s_t \in S$ where *S* is the set of all possible states, and in that state the agent can take an action $a_t \in (At)$, where (At) is the set of all possible actions in the state s_t. As the agent transits to a new state s_{t+1} at time $t + 1$ it receives a numerical reward r_{t+1}. It up to date then its estimate of the function of evaluation of the action using the immediate reinforcement, r_{t+1}, and the estimated value of the following state, $Vt\ (S_{t+1})$, which is defined by:

$$V_t\left(s_{t+1}\right) = \max_{u_t \in U_{t+1}} Q\left(s_{t+1}, u_t\right) \tag{2}$$

the Q-value of each state/action pair is updated by

$$Q\left(s_{t+1t}, u_t\right) = Q\left(s_t, u_t\right) + \beta\left\{r_{t+1} + \gamma V_t\left(s_{t+1}\right) - Q\left(s_t, u_t\right)\right\} \tag{3}$$

Where $r_{t+1} + \gamma V_t\left(s_{t+1}\right) - Q\left(s_t, u_t\right)$ the TD error and β is is the learning rate.
This algorithm is called Q-Learning. It shows several interesting characteristics. The estimates of the function Q, also called the Q-values, are independent of the policy pursued by the agent. To calculate the function of evaluation of a state, it is not necessary to test all the possible actions in this state but only to take the maximum Q-value in the new state (eq.4). However, the too fast choice of the action having the greatest Q-value:

$$u_t = \arg \max_{u_t \in U_{t+1}} Q\left(s_{t+1}, u_t\right) \tag{4}$$

can lead to local minima. To obtain a useful estimate of Q, it is necessary to sweep and evaluate the whole of the possible actions for all the states: it is what one calls the phase of exploration L. Jouffe, A. Souissi. In the preceding algorithm, called TD (0), we use only the state which follows the robot evolution, moreover only the running state is concerned. Sutton A. Souissi extended the evaluation in all the states, according to their eligibility traces that memorise the previously visited state action pairs in our case. Eligibility traces can be defined in several ways L. Jouffe, C. Watkins, P. Dayan, A. Souissi. Accumulating eligibility is defined by:

$$e_t(s) = \begin{cases} 1 + \gamma\lambda e_{t-1}(s) & si \quad s = s_t \\ \gamma\lambda e_{t-1}(s) & \text{else} \end{cases} \tag{5}$$

The algorithm Q (λ) is a generalization of Q-Learning which uses the eligibilities traces A. Souissi: *Q-Learning* is then a particular case of Q (λ) when λ=0.

$$Q\left(s_{t+1}, u_t\right) = Q\left(s_t, u_t\right) + \beta\left\{r_{t+1} + \gamma V_t\left(s_{t+1}\right) - Q\left(s_t, u_t\right)\right\} . e_t(s) \tag{6}$$

3. Fuzzy Q-learning algorithm

In mobile robotics, input and output variables given by the sensors and effectors are seldom discrete, or, if they are, the number of state is very large. However reinforcement learning such as we described it uses a discrete space of states and actions which must be have reasonable size to enable the algorithms to converge in an acceptable time in practice.

The principle of the Fuzzy Q-Learning algorithm consists in choosing for each rule R_i a conclusion among a whole of actions available for this rule. Then it implements a process of competition between actions. With this intention, the actions corresponding to each rule R_i have a quality q^i which then determines the probability to choose the action. The output action (*discrete* or *continues*) is then the result of the inference between various actions locally elected. The matrix q enables to implement not only the local policies of rules, but also to represent the function of evaluation of the overall t-optimal policy.

For every time step t the agent is in a state $st \in S$ where S is the set of all possible states, and in that state the agent can take an action $at \in (At)$, where (At) is the set of all possible actions in the state st. As the agent receives a numerical reward $rt+1 \in R$ at time $t + 1$, and it transits to a new state $st+1$. It then perceives this state by the means of its activated degree of its rules. The algorithm FQL uses a Policy of Exploration/Exploitation (PEE) L. Jouffe, A. Souissi, combining a random exploration part $\rho^i(U)$ and determinist guided part $\eta^i(U)$.

The steps are summarized as follows:

1. Calculate of the evaluation function :

$$Q_t^*(S_{t+1}) = \sum_{R_i \in A_t} (\underset{U \in U}{Max}(q_t^i(U)) \alpha_{R_i}(S_{t+1}),$$

2. Calculate the TD error

$$\tilde{\varepsilon}_{t+1} = r_{t+1} + \gamma Q_t^*(S_{t+1}) - Q_t(S_t, U_t)$$

3. Update the matrix of Q values

$$q_{t+1}^i = q_t^i + \beta . \tilde{\varepsilon}_{t+1} e_t^{iT}, \forall R_i$$

4. Election of the action to be applied

$$U_{t+1}(S_{t+1}) = \sum_{R_i \in A_{t+1}} Election_U(q_{t+1}^i) \alpha_{R_i}(S_{t+1}), \forall U \in U,$$

Where Election is defined by:

$$Election_U(q_{t+1}^i) = ArgMax_{U \in U}(q_{t+1}^i(U) + \eta^i(U) + \rho^i(U)),$$

5. Update of the eligibility traces

$$e_{t+1}^i(U^i) = \begin{cases} \gamma \lambda e_t^i(U^i) + \phi_{t+1}^i, (U^i = U_{t+1}^i) \\ \gamma \lambda e_t^i(U^i), \quad \text{sinon} \end{cases}$$

$$Q_{t+1}(S_{t+1},U_{t+1}) = \sum_{R_i \in A_{t+1}} q^i_{t+1}(U^i_{t+1})\alpha_{R_i}(S_{t+1}),$$

This value will be used to calculate the TD error in the next step time. However the performances of the controller are closely dependent on the correct choice of the discrete actions set, witch is determined using *a priori* knowledge about system, for complex systems like robots, *priori* knowledge are not available, then it becomes difficult to determine a set of correct actions in which figure the optimal action for each fuzzy rule. To solve this problem and to improve the performances of the reinforcement learning, the genetic algorithms will explore the broadest space of solutions to find the solution optimal Dongbing Gu, Huosheng Hu, Libor Spacek Chia-Feng Juang Min-Soeng Kim and Kim and Ju-Jang Lee Chia-Feng Juang and Chun-Feng Lu, and that without any priori knowledge.

4. Genetique reinforcement algorithmes

Genetic Algorithms (GA) are stochastic optimization algorithms, founded on species evolution mechanisms David E Goldberg. In GA, a candidate solution for a specific problem is called an individual or a chromosome and consists of a linear list of genes. Each individual represents a point in the search space and, hence, a possible solution to the problem. A population consists of a finite number of individuals. Each individual is decided by an evaluating mechanism to obtain its fitness value. Based on this fitness value and undergoing genetic operators, a new population is generated iteratively, with each successive population referred to as a generation.

Genetic Reinforcement Learning enable to determine the best set of parameters of (antecedents / consequences) starting from a random initialization of these parameters and in each iteration, only one action is applied on the system to the difference of a traditional genetic algorithm GA use three basic operators (the selection, crossover, and mutation) to manipulate the genetic composition of a population:

- *Reproduction*: Individuals are copied according to their fitness values. The individuals with higher fitness values have more offspring than those with lower fitness values.
- *Crossover*: The crossover will happen for two parents that have high fitness values with the crossover probability *pc*. One point crossover is used to exchange the genes.
- *Mutation*: The real value mutation is done by adding a certain amount of noise (Gaussian in this paper) to new individuals to produce the offspring with the mutation probability ***pm***. For the ***ith*** variable in ***jth*** individual, it can be expressed as:

$$a_{t+1} = a_t + \beta(i).N(0,\sigma) \tag{7}$$

Where ***N*** denote a Gaussian noise, and $\beta(i) = \exp(-i)$ for the i^{th}generation.

A. FQLGA Algorithm

Because of its simplicity, a Takagi-Sugeno Fuzzy inference system is considered, with triangular membership function. The structure (partition of its input space, the determination of the number of *IF-THEN* rules) of the FIS is predetermined.

$$Rule\ R_i : if\ S_1\ is\ L^i_1\ \ and.....and\ \ S_{Ni}\ is\ L^i_{Ni}\ \ then\ \ Y\ \ is\ \ a^i_j\ \ with\ \ q^i_j$$

a_j^i is a vector representing the discrete set of *K* conclusions generated randomly for the rule R_i with which is associated a vector q_j^i representing the quality of each action (*i= 1 ~N and j= 1 ~ K).*

The principle of the approach is to use a population of *K* (FIS) and to treat the output of each one of them as a possible action to apply on the system. FQL algorithm exploits local quality function *q* witch is associated with each action of a fuzzy rule (équat.6) whereas FQLGA algorithm uses as function fitness the sum of local qualities *q* given by:

$$f(Ind_j) = Q(S_{t+1}, SIF_{t+1}) = \sum_{i=1}^{N} q^i_j \tag{8}$$

To reduce the training time, the quality matrix Q is not initialized after every iteration, but undergoes the same genetic operations as those applied to the set of the individuals (selection, crossing).

B. Optimization of the consequent part of a FIS

A population of *K* individuals of a predetermined structure is adopted. The size of an individual is equal to number *N* of the FIS's rules. The architecture of FQLGA algorithm proposed for the optimization of the conclusions is represented on the figure (1).

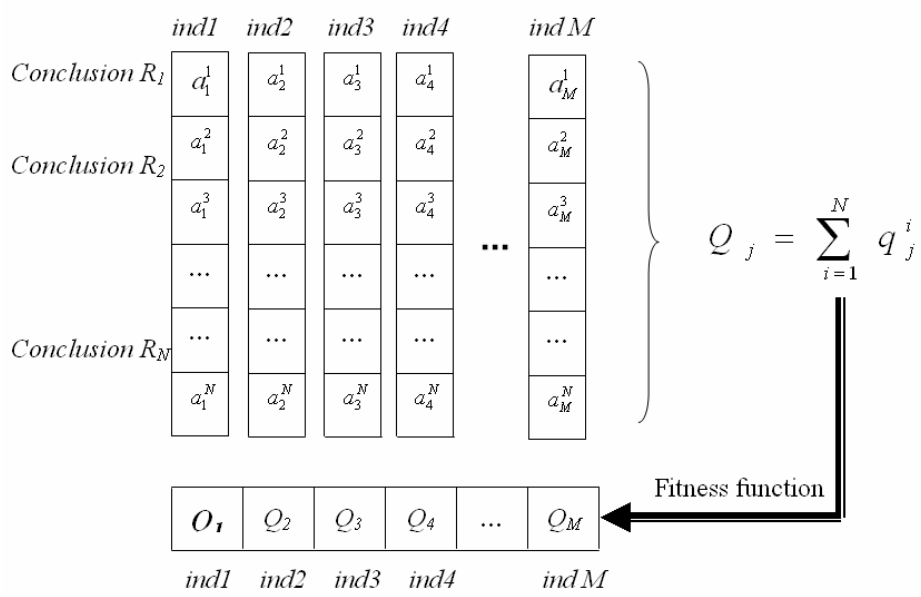

Fig. 1. Representation of the individuals and qualities of the actions in FQLGA algorithm

C. Optimization of the antecedent part of a FIS

To find the best set of premises generated by GA, a population made up of *M* FIS is created. Each individual (FIS) of the population encode the parameters of the antecedents i.e. the

modal points of the FIS and his performance is evaluated by the fitness function of *Q* (global quality).

The conclusion part of each individual FIS remains fixed and corresponds to the values determined previously. The coding of the membership functions of the antecedent part of a FIS (individual) is done according to the figure (2). To keep the legibility of the FIS, we impose constraints during the evolution of the FIS to ensure the interpretability of the FIS.

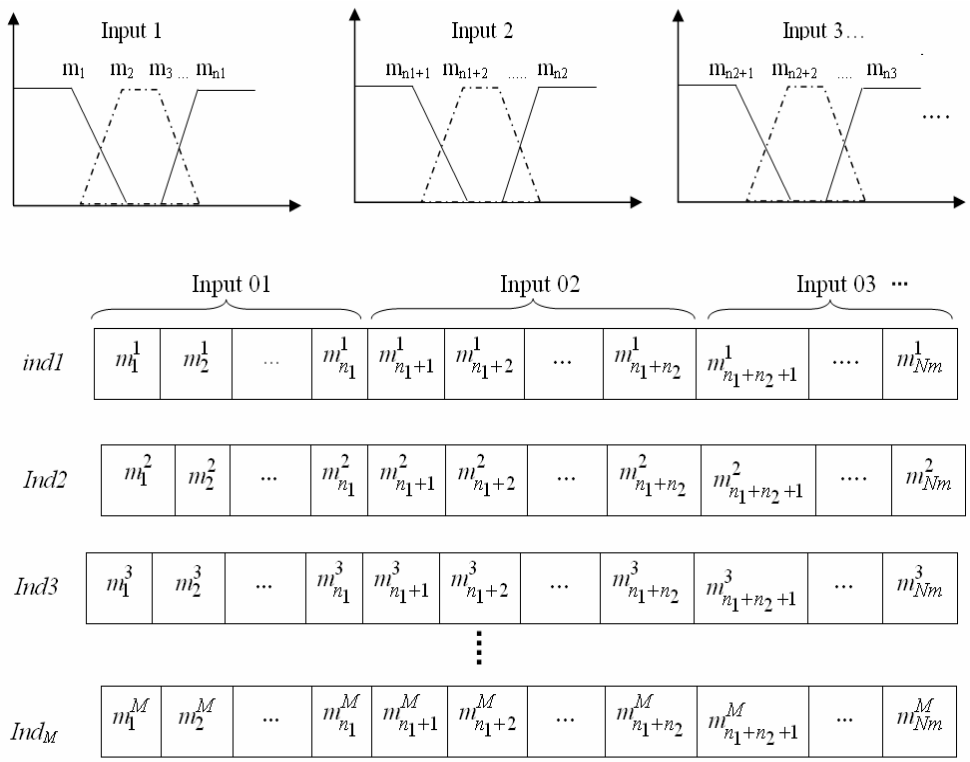

Fig. 2. Coding of the parameters of the antecedent part

Input N: $m^1_{N_m - n_N + 1} < \ldots < m^1_{Nm\text{-}2} < m^1_{Nm}$

The fitness function used in by genetic algorithm for the optimization of the antecedent part is the global quality of the FIS which uses the degree of activation of the fuzzy rules; this fitness function is given by the following equation:

$$f(ind_i) = Q(S(t), SIF_i) = \frac{\sum_{R_i \in R_A} \alpha_{Ri}(S(t)).q_i^i(t)}{\sum_{R_i \in R_A} \alpha_{R_i}(S(t))} \tag{9}$$

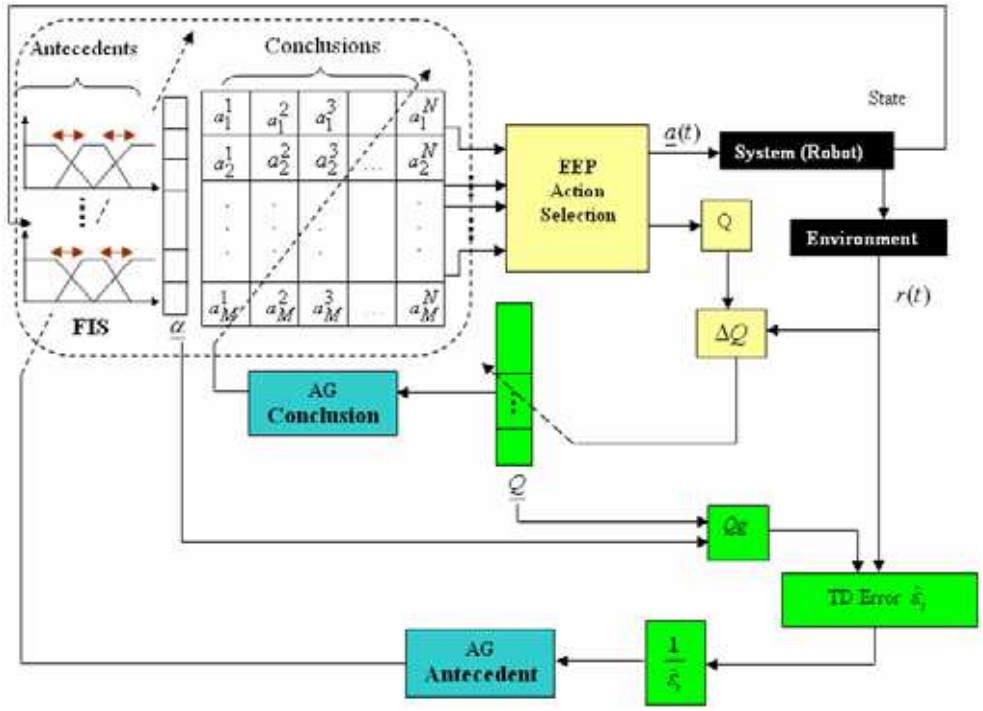

Fig. 3. General Architecture of FQLGA algorithm

D. Optimization of the Consequent and antecedent part of FIS:

A FIS consists of a set of fuzzy rules each one of it is made of an antecedent and a consequent part. Optimize the antecedent and the consequent part of the fuzzy controller at the same time is a complex problem which can admit several solutions which are not acceptable in reality, the great number of possible solutions makes the algorithm very heavy and can lead to risks of instability.

FQLGA algorithm proposed in (fig.3) for the optimization of the premises and the conclusions allows the total parametric optimization of the FIS in three stages represented in the flow chart (fig.4).

- At the beginning of the learning process, the quality matrix is initialized at zero, and then traditional algorithm FQL evaluates each action using an exploration policy. This step finishes when a number of negative reinforcements is received.
- After the evaluation of the individuals, the genetic algorithm for the optimization of the consequent part of the fuzzy rules creates a new better adapted generation. This stage is repeated until obtaining convergence of the conclusions or after having reached a certain number of generations. The algorithm passes then to the third stage:
- Once the conclusions of the FIS are optimized, the second genetic algorithm for the optimization of the antecedent part is carried out to adjust the positions of the input membership functions of the controller which are initially equidistant on their universe of discourse.

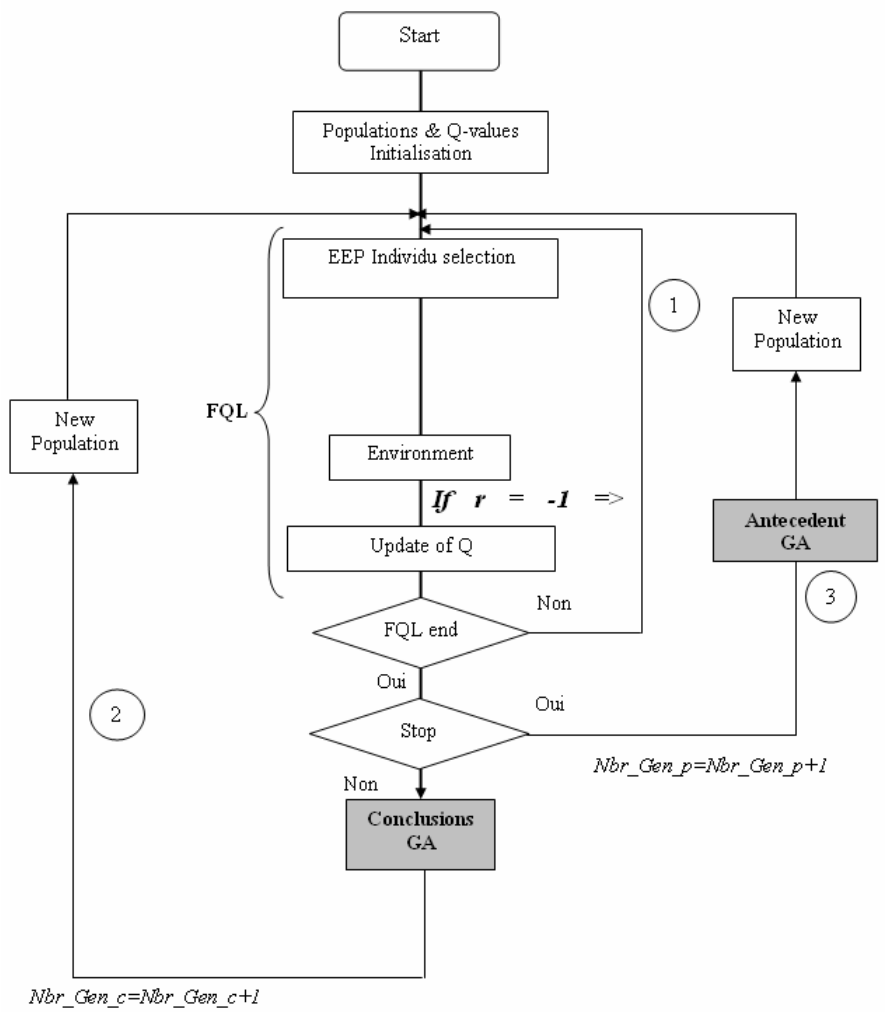

Fig. 4. Flow chart of FQLGA algorithm

5. FQLGA simulation & experimental results

To verify the performance of FQLGA two elementary behaviours of reactive navigation of a mobile robot: "Go to *Goal*" and "*Obstacles Avoidance*" are presented in this section; the algorithm was adopted in the experiments for both simulations (Saphira simulator) and real robots (Pioneer II).

A. «Go to Goal» behaviour:

The two input variables are: the angle"θ_{Rb}" between the robot velocity vector and the robot-goal vector, and the distance robot-goal "ρ_b".They are respectively defined by three (*Negative, Zeros, Positive*) and two (*Near, Loin*) fuzzy subsets (fig.5). The two output variables are the

rotation speed, *Vrot_CB* and the translation speed *Vtran_CB* each output is represented by nine actions initialised randomly.

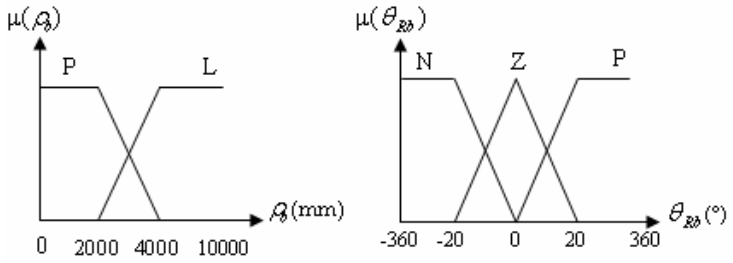

Fig. 5. Membership functions of the input space

The reinforcement functions adopted for the two outputs are respectively given by:

$$r_{Vrot\_CB}(t)=\begin{cases}1 & Si \quad (\theta\cdot\dot{\theta}<0 \;\; ou \;\; -1°<\theta<+1°)\\ 0 & Si \quad (\theta\cdot\dot{\theta}=0)\\ -1 & Else\end{cases} \tag{10}$$

$$r_{Vtran\_CB}(t)=\begin{cases}1 & Si \quad Vtrans\le 0.15\rho_0 \text{ et } \rho_0\ge 800\\ 1 & Si \quad Vtrans\ge 220 \text{ et } \rho_0\ge 800 \text{ et } abs(\theta)<5\\ -1 & Else\end{cases}$$

The parameters of FQL algorithm and the genetic algorithms are as follows:

γ	λ	L_p	L_c	N_p	N_c	p_m
0.9	0.7	5	6	10	09	0.2

L_P and Lc respectively indicate the sizes of the chromosomes for the antecedents and the conclusions part, Np, Nc respectively represent the size of the population of the parameters of antecedent and the conclusions and Pm the probability of mutation.

The simulation results of the first behaviour "Go to Goal" are presented in the figure (6), 28 generations were sufficient to find the good actions.

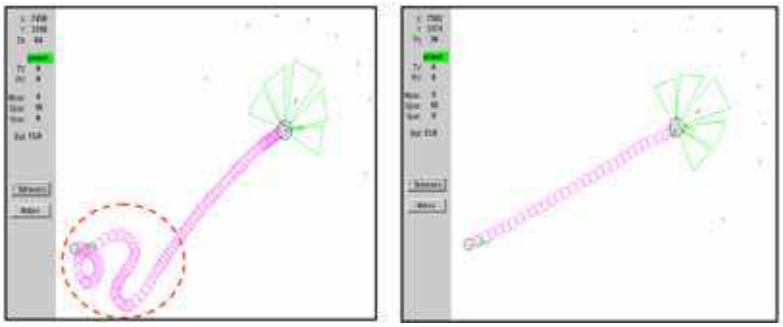

Fig. 6. Go to goal: Training/Validation

Figure (7) shows the convergence of the fitness values of the genetic algorithms for the two output variables *Vrot_CB* and *Vtran_CB* obtained during experimental test.

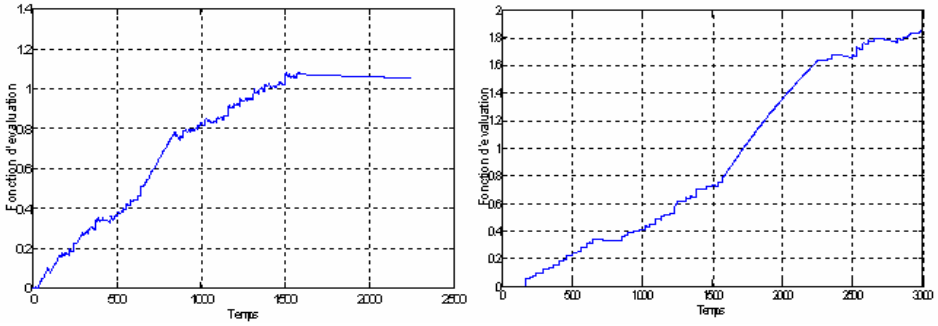

Fig. 7. fitness functions for the two output variables

B. « Obstacles Avoidance » behaviour

The Inputs of the FIS are the distances provided by the ultrasounds sensors from the three directions (Frontal, Left and Right) and defined by the three fuzzy subsets: near (N), means (M), and far (F) (fig.8).

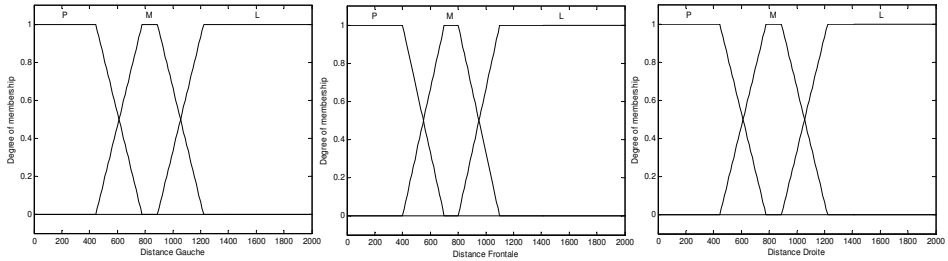

Fig. 8. Memberships Functions of the inputs variables

The conclusions (rotation speeds) are initialized randomly. The translation speed of *Vtran_EO* is given analytically; it is linearly proportional to the frontal distance:

$$Vtran\_EO = \frac{V\max}{D\max}.(Dis\_F - Ds) \tag{11}$$

$V\max$ is the maximum speed of the robot equal to 350mm/s.

$D\max$ is the maximum value measured by the frontal sensors and estimated to 2000mm and Ds is a safe distance witch is fixed at 250mm.

The function of reinforcement is defined as follows [5]:

$$r_{Vrot\_CB}(t) = \begin{cases} 1\times Signe\ (Vrot\_EO) & if\ Dis\_R < Dis\_L\ or\ Dis\_F<Dis\_L \\ & and\ D\min<800 \\ -1\times Signe\ (Vrot\_EO) & and\ Dis\_L < Dis\_R\ or\ Dis\_F<Dis\_R \\ & and\ D\min<800 \\ 0 & elsewhere \end{cases} \tag{12}$$

with *Dmin=min (Dis_L, Dis_F, Dis_R)*
The parameters of FQL algorithm and the genetic algorithms are identical to the preceding behaviour except for the sizes of the chromosomes *Lp*=12 and *Lc*=27.
Figure (9) represents the trajectories of the robot during the learning phase with FQL algorithm and a random initialization of the consequents part for the 27 fuzzy rules. Several situations of failures are observed, this example show the limits of traditional FQL algorithm when priory knowledge about the system are not available.

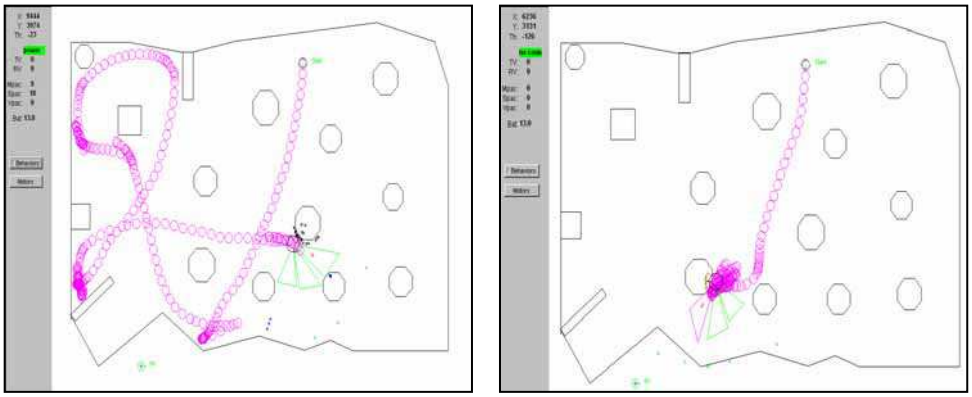

Fig. 9. Trajectories of the robot obtained by FQL algorithm using a random initialization of parameters

The trajectories of figure (10) show the effectiveness of the association of the reinforcement learning FQL and the genetic algorithm as stochastic tool for exploration. FQLGA Algorithm enables to find an optimal FIS for the desired behaviour (obstacles avoidance). The duration of learning depend to the genetic algorithm parameters and the obstruction density of the environment. We observe that after each generation the quality of the FIS (sum of local qualities) increases, which give more chance to the best individuals to be in the next generations.

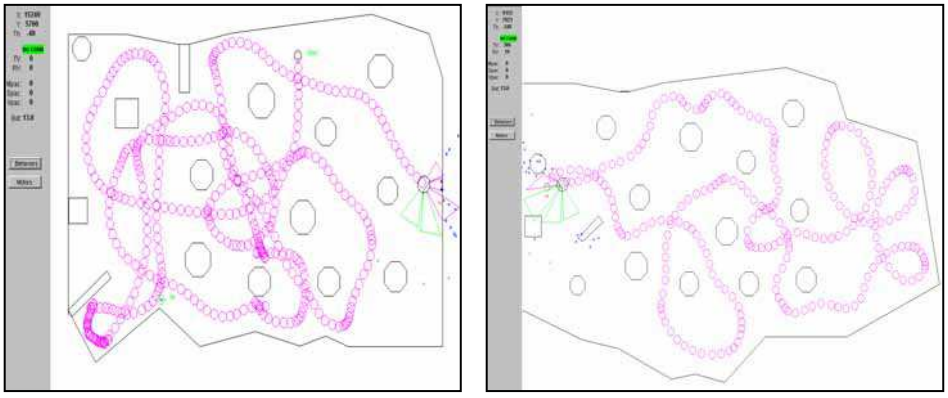

Fig. 10. Learning/Validation Trajectories of the robot with FQLGA algorithm for various environments

Figure (11) shows the performances of FQLGA algorithm compared to the FQL algorithm which can be blocked in a local minimum when the optimal solution is not present in the randomly generated set of actions. On the other hand FQLGA algorithm converges towards the optimal solution independently of the initialized values.

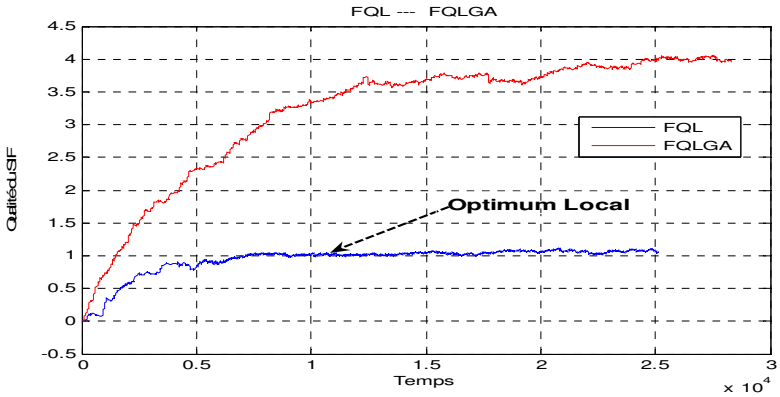

Fig. 11. Evolution of the quality of the Fuzzy controller with FQL and FQLGA algorithms

C. Experimental results with the real robot Pioneer II

Figure (12) represents the results of the on line learning of the robot Pioneer II for the behaviour "Go to goal". During the learning phase, the robot does not follow a rectilinear trajectory (represented in green) between the starting point and the goal point because several actions are tested (exploration). Finally the algorithm could find the good actions, and the robot converges towards the goal marked in red colour, the necessary time to find these good actions is estimated at 2mn. Each generation is generated after having noted twenty (20) failures. The learning process requires respectively 32 and 38 generations for GA to determine rotation and translation speed.

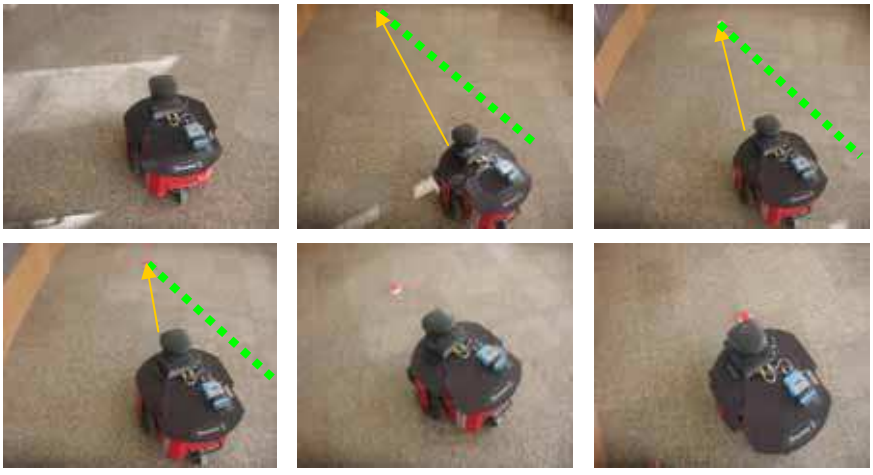

Fig.12**.** On line learning of the real robot Pioneer II, "Go to goal" behaviour

Figure (13) represents the results of the on line learning of the "Obstacles Avoidance" robot behaviour. For reasons of safety, we consider that the minimal distances detected by the frontal sonars and of left/right are respectively 408 mm and of 345 mm a lower value than these distances is considered then as a failure and involves a retreat (represented in green) of the mobile robot. A generation is created after 50 failures. The genetic algorithms require 14 generations to optimize the conclusions and 24 generations to optimize the parameters of the antecedents. The duration of on line learning is estimated at 20 min, this time is acceptable vis-à-vis the heaviness of the traditional genetic algorithms.

Fig. 13. On line learning of the real robot Pioneer II, behaviour "Obstacle Avoidance»

Figure (14) represents the evolution of the fitness function obtained during this experimentation.

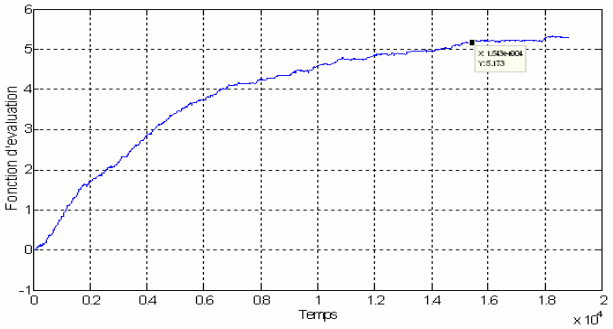

Fig.14 Fitness function evolution, "Obstacles avoidance" robot behaviour

6. Dynamic Fuzzy Q-Learning Genetic Algorithm (DFQLGA)

A. Algorithm DFQL

The algorithm Dynamic Fuzzy Q-Learning (DFQL) proposed by Meng Joo Er, et Chang Deng is an extension of the original Q-learning method into a fuzzy environment. State-space coding is realized by the input variable fuzzy sets. A state described by a vector of fuzzy variables is called a fuzzy state. A learner may partially visit a fuzzy state, in the sense that real-valued descriptions of the state of the system may be matched by a fuzzy state description with a degree less than one. Since more than one fuzzy state may be visited at the same time, possibly with different degrees, we have a smooth transition between a state and its neighbours, and, consequently, smooth changes of actions done in the different states. Both the actions and the Q-function are represented by a FIS whose fuzzy logic rules can be self-constructed based on the -completeness of fuzzy rules and the TD error.

1) ε – *Completeness* criteria for Rule Generation

Definition: -Completeness of fuzzy rules C. C. Lee: For any input in the operating range, there exists at least one fuzzy rule so that the match degree (or firing strength) is no less than ε. In fuzzy applications, the minimum value of ε is usually selected as $\varepsilon = 0{,}5$.
From the viewpoint of fuzzy rules, a fuzzy rule is a local representation over a region defined in the input space. If a new pattern satisfies ε – *Completeness*, the DFQL will not generate a new rule but accommodate the new sample by updating the parameters of existing rules.

2) TD Error Criterion for Rule Generation

It is not sufficient to consider -completeness of fuzzy rules as the criterion of rule generation only. New rules need to be generated in regions of the input fuzzy subspace where the approximation performance of the DFQL is unsatisfactory. Here, we introduce a separate performance index ξ^i, for each fuzzy subspace which enables the discovery of "problematic" regions in the input space. The performance index is updated as follows Meng Joo Er,et Chang Deng:

$$\xi_{t+1}^{i} = [(K - \alpha_t^i)\xi_t^i + \alpha_t^i(\tilde{\varepsilon}_{t+1})^2]/K \quad \text{avec} \quad K > 0$$

By developing the preceding equation, the errors TD coefficients are expressed as follows:

$$\begin{cases} \xi_1^i = \dfrac{\alpha_0^i}{K}\varepsilon_1^2 \\ \xi_2^i = (\dfrac{K-\alpha_1^i}{K})\cdot(\dfrac{\alpha_0^i}{K})\varepsilon_1^2 + (\dfrac{\alpha_1^i}{K})\varepsilon_2^2 \\ \xi_3^i = (\dfrac{K-\alpha_2^i}{K})\cdot(\dfrac{K-\alpha_1^i}{K})\cdot(\dfrac{\alpha_0^i}{K})\varepsilon_1^2 + (\dfrac{K-\alpha_2^i}{K})\cdot(\dfrac{\alpha_1^i}{K})\varepsilon_2^2 + (\dfrac{\alpha_2^i}{K})\varepsilon_3^2 \\ \vdots \end{cases} \tag{13}$$

After each iteration, all preceding errors TD are balanced by the term $\dfrac{K-\alpha_{t-1}^i}{K} < 1$.

Using the squared TD error as the criterion, the rule firing strength α^i determines how much the fuzzy rule R_i affects the TD error. It should be noted that (13) acts as a digital low-pass filter. In this way, TD errors in the past are gradually "forgotten" as time passes, but are never completely lost. The more recent the TD error is received, the more it affects the value of ξ. The initial value of ξ^i is set to zero. The parameter K which controls the overall

behaviour of ξ is usually selected between ten and 100. A small value of K makes ξ adapt very rapidly and a large value makes ξ more stable in a noisy environment. Thus, if ξ^i is lower than a certain threshold k_e, further segmentations should be considered for this fuzzy subspace at least. The figure (15) represents the global flowchart of the DFQL algorithm Meng Joo Er,et Chang Deng.

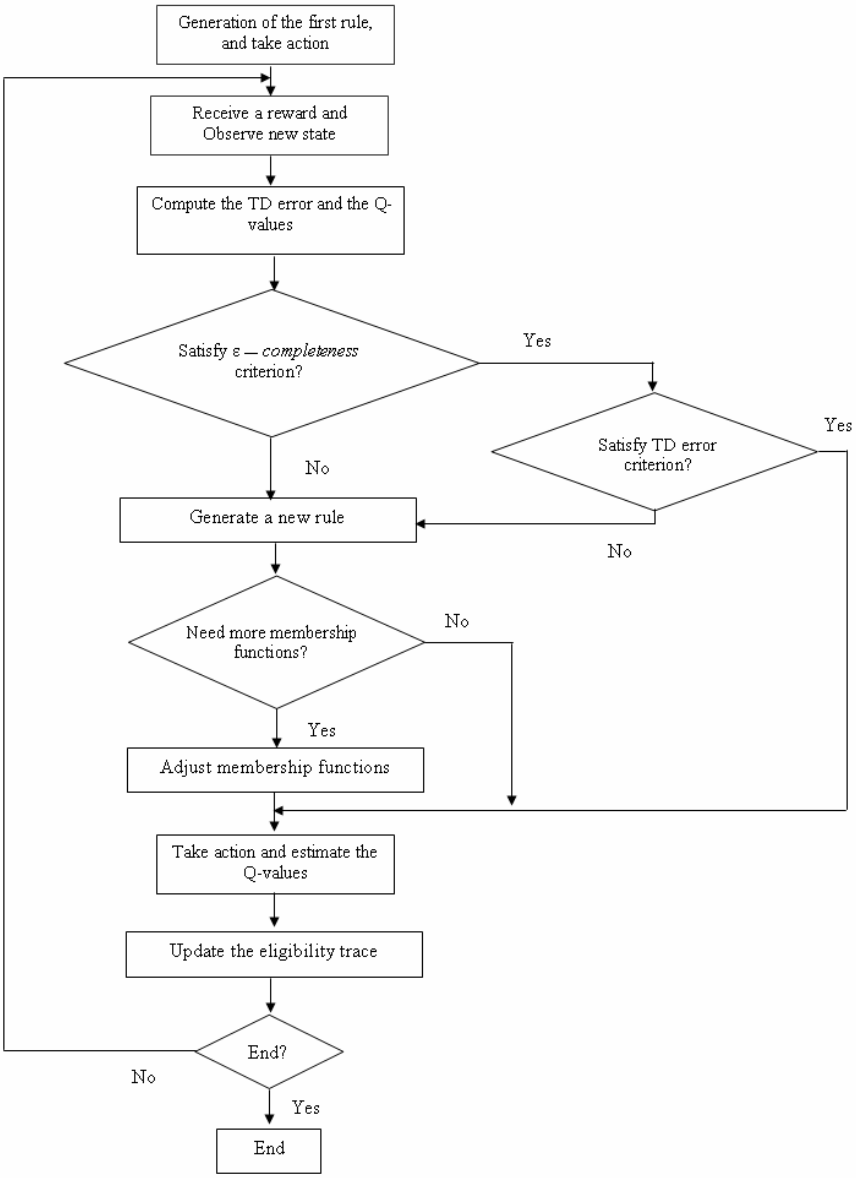

Fig. 15. DFQL algorithm flowchart

B. Dynamic Fuzzy Q-Learning Genetic Algorithm (DFQLGA)

The structural and parametric optimization of a fuzzy inference system without any *priori* knowledge, is the objective of several research, several approaches suggested in the literature are based on the reinforcement genetic method.

The Algorithm FQLGA, presented in A. Nemra H. Rezine, ensures a parametric optimization; however the optimization of the conclusions and premises separately doesn't exploit effectively the power of the genetic algorithm. The SEFC Algorithm proposed by Chia-Feng Juang, Jiann-Yow Lin, and Chin-Teng Lin, permit to solve this problem. Using the Michigan approach, when a fuzzy rule (premises & conclusions) is regarded as an individual and the FIS is built by the combination of several individuals.

However for these two algorithms the structure of the FIS (the number of fuzzy rules and membership function) is predefined by the human expert, which is not always easy, especially in the case of the complexes systems. For the latter an automatic design of the FIS becomes necessary, which is the objective of the learning algorithm DFQL which generates the rules and the membership functions automatically during the learning according to the two criterias mentioned before. The results obtained by algorithm DFQL are considered to be satisfactory if the predefined set of actions is correct and suitable for the desired task, nevertheless, if it is not the case (no priori knowledge) then the apprentice is likely to fail to do its task appropriately. Then as a solution for this problem we develop a new algorithm the DFQLGA.

C. Principle of the DFQLGA algorithm

The new learning algorithm suggested is called Dynamic Fuzzy Q-Learning Genetic Algorithm (DFQLGA) is a combination of the algorithm DFQL and the genetic algorithm (GA) developed previously. With this algorithm, the fuzzy rules and the memberships functions of the premise part are generated automatically by the DFQL algorithm Meng Joo Er, Chang Deng according to *ε-Completeness* and *error TD* criteria. The conclusion part of the fuzzy rules, as it is randomly initialized (a random set of actions for each fuzzy rule), then the genetic algorithm is used to determine the best conclusions of the rules previously generated.

D. Optimization of the conclusions by GA

The considered Fuzzy Inference System (FIS) is a zero order Takagi-Sugeno type, composed by one fuzzy rule only, initialized randomly as follow:

$Rule\ Ri: if\ \ S1\ is\ L_1^i\ \ and.....and\ S_{Ni}\ is\ L_{Ni}^i\ then\ Y\ is\ a_j^i\ \ with\ q_j^i$ a_j^i is a vector representing the discrete set of K conclusions generated randomly for the output variable Y of the rule R_i with which a vector representing the quality q_j^i is associated with each action. The initialization of the fuzzy rule is random for the conclusion part, the membership functions of the premises are of type Gaussian and has unspecified parameters (Means, Standard deviation).

The update of local quality q_j^i is given by the equation (eq.6), the algorithm DFQLGA uses as fitness function the sum of local qualities Q given by (fig.16):

$$f(Ind_j) = Q(S_{t+1}, SIF_{t+1}) = \sum_{i=1}^{N} q_j^i \tag{14}$$

To accelerate the time of learning, the quality matrix Q is not initialized at any iteration, but undergoes the same genetic operations as those applied on the population of the individuals