

# Accuracy Based Fuzzy Q-Learning for Robot Behaviours

Dongbing Gu and Huosheng Hu  
Department of Computer Science  
University of Essex  
Wivenhoe Park  
Colchester, CO4 3SQ, UK  
Email: [dgu@essex.ac.uk](mailto:dgu@essex.ac.uk), [hhu@essex.ac.uk](mailto:hhu@essex.ac.uk)

**Abstract**—This paper presents a learning approach to fuzzy classifier systems. Q-learning algorithm is employed to implement credit assignment of the learning. GA operators are used as an action selection mechanism of the learning. The learning approaches can be viewed as a fuzzy learning classifier system or a Q-learning algorithm that adopts fuzzy logic to generalise Q-learning results. Rule accuracies are treated as rule fitness values. The learning algorithm is applied to a control robot behaviour.

## I. INTRODUCTION

In the learning of classifier systems (CSs), environment payoffs are apportioned to rule strengths. Credit assignment is conducted through updating the rule strengths by the bucket brigade algorithm or temporal sequence (TD) algorithm. A rule strength value is regarded as a fitness value. Through evolving rules, optimal or sub-optimal CSs can be obtained by choosing the rules with high strength values. This is called the rule strength based approach, i.e. the payoffs is assigned to individual rules according to their activations [3][4].

In the learning of fuzzy CSs (FCSs), the strength-based approach becomes more complex since outputs of FCSs depend on the activated fuzzy rules at each input state, not a single rule. By using a fuzzy state definition [1][2][10], payoffs apportioned to each rule depend on rule's contribution in fuzzy inference to final outputs. A rule's contribution in each fuzzy state is proportional to the rule activated degree in the state. This method was applied to robot control in many researches [2][7][12].

Based on the fuzzy state definition, FCSs can be described by a Markov Decision Process (MDP) model, where each rule corresponds to a fuzzy state-action pair. A utility value  $q$  can be defined for each fuzzy rule to estimate accumulated payoffs for fuzzy state-action pair of the rule. The learning of FCSs is then converted to the learning of utility values. The optimal or sub-optimal FCS can be found

by choosing the fuzzy action with the highest  $q$  value for each rule. The credit assignment is achieved by Q-learning algorithms by updating  $q$  values. This was called the fuzzy Q-learning in [6][9]. Researches in [5][2] showed the fuzzy Q-learning is an approach to generalising crisp CSs to continuous cases.

In these strength-based learning approaches (either CSs or FCSs), A rule strength value is a division of an estimated payoff and serves as a rule's fitness value for GA learning. However, the division of estimated payoffs may not be equivalent to the rule fitness value, as different environment niches would produce different payoff levels. A low strength rule may be the best one for its niche [18]. Various rules solve different parts of a problem. The rules in the same niches should compete to perform same task and earn the same rewards, while rules in different niches should cooperate in their tasks. Use of the strength-based classifier learning would lead to population domination of the rules in high-payoff niches and finally lead to the learning converging to local optima.

An accuracy-based classifier learning approach was proposed in [18], which still maintains rule strengths, but uses a measure of the strength accuracy as a rule's fitness value. The strength accuracy indicates how well a rule estimates payoffs whenever the rule fires. Those rules that receive high payoffs in one niche are not luck to receive high payoffs in other niches. Use of the rule accuracy as the rule fitness value in GA learning can distinguish between accurate rules and inconsistent-payoff rules.

The accuracy-based classifier learning was applied in FCSs in [13][14]. The research in [14] was a fuzzy version of crisp CSs, in which payoffs are apportioned to fuzzy rules based on rule's activation by the brigade bucket algorithms. The learning algorithm in [13] adopted as a rule fitness value an inverse of an absolute error. The absolute error was acquired by comparing a CS output with a given correct output. Its credit assignment mechanism distributes payoffs to rules proportional to rule's activation.

This paper proposes an accuracy-based learning approach for FCSs based on a Q-learning credit assignment strategy. This approach adopts an inverse measure of the rule accuracy as a rule's fitness value. Each rule maintains a  $q$  value, but it is no longer an estimation of accumulated payoffs. It is only used for calculating the rule accuracy. This  $q$  value is updated by the Q-learning mechanism. An on-line Q-learning algorithm, fuzzy Q-learning algorithm, is developed for the credit assignment. The max operator in the standard Q-learning is not used since the rules that have maximum  $q$  values no longer represent rules with the best payoffs. The action selection mechanism employs a niching GA to select actions in sub-populations.

In the following, section II introduces the fuzzy Q-learning method. The accuracy-based learning is addressed in section III. In section IV, the proposed learning algorithm is described. In section V, the implementation is conducted in a simulator and a real robot. A chase-ball behaviour is evaluated. Finally a summary for this paper is provided in section VI.

## II. STRENGTH BASED FUZZY Q-LEARNING

Reinforcement learning of robots investigates the interaction between a robot and its environment, concerning robot states, actions, and payoffs. A MDP can be used to model the interaction. A state space  $S$  is defined on sensor information  $s_k$ . An action space  $A$  is defined on robot motion commands  $a_k$ . The MDP model is expressed as  $II=(S, A, P, R)$  and a data set  $(s_k, a_k, r_k, s_{k+1}, a_{k+1})$  represents the interaction at time step  $k$ .

In Q-learning, an optimal policy is found by maximising payoffs received over time. Q-learning defines a  $Q(s, a)$  value for each pair of state  $s$  and action  $a$ .

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

where  $\gamma$  is a discount factor ( $0 \leq \gamma \leq 1$ ),  $r_t$  is a payoff value at time  $t$ , and  $\alpha$  is a learning rate.

For discrete applications, Q values can be listed in a table, in which each item represents a state-action pair. For continuous applications, the learning becomes more difficult, as it is impossible to list all continuous values in a table. Many generalisation techniques have been used to approximate the Q table, for example, neural networks in [15], CMAC in [17], locally weighted regression in [16], and fuzzy logic in [7].

A FCS can map a state  $s_t$  to an action  $a_t$ . It can also map a state-action pair  $(s_t, a_t)$  to a  $Q(s_t, a_t)$  value in a continuous state space. The following expression is a rule in a FCS for the mapping:

$$R_m: \text{IF } s_t \text{ is } F_1^m \text{ AND } \dots \text{ AND } s_N \text{ is } F_N^m$$

$$\text{THEN } a \text{ is } c_m^k \text{ with } q_m^k \quad (2)$$

where  $q_m^k$  is the  $k$ th  $q$  value in a rule  $m$ . and  $F$  denotes a fuzzy set. Following fuzzy inference, a  $Q$  value is calculated as follows:

$$Q(s_t, a_t) = \frac{\sum_{m=1}^M \delta_m(s_t) \cdot q_m^k}{\sum_{m=1}^M \delta_m(s_t)} \quad (3)$$

where  $M$  is the number of rules and  $\delta$  is the rule activation degree. It is expected to find the changes of  $q$  values in order to update them. This is achieved through finding changes of  $Q$  values in (1) first, and then calculating them based on (3). From (1), we have:

$$\Delta Q(s_t, a_t) = \alpha(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (4)$$

The changes of  $q$  values can be found by calculating the gradient in (3):

$$\Delta q_m^k = \Delta Q \cdot \frac{\partial Q(s_t, a_t)}{\partial q_m^k} = \Delta Q \cdot \frac{\delta_m(s_t)}{\sum_{m=1}^M \delta_m(s_t)} \quad (5)$$

For multi-step Q-learning, an eligibility trace is maintained for each  $q$  value. The eligibility trace is defined as a replacement type:

$$e(m, k) = \begin{cases} 1 & m = m_t, k = k_t \\ \lambda \gamma e(m, k) & \text{otherwise} \end{cases} \quad (6)$$

where  $m_t$  and  $k_t$  denote the rule number and action number at time step  $t$ . The updating formula for  $q$  values becomes as follows ( $\beta$  is an updating rate):

$$q_m^k = q_m^k + \beta \Delta q_m^k \cdot e(m, k) \quad (7)$$

## III. ACCURACY BASED FCS LEARNING

The  $q$  values in (2) are calculated according to total accumulated payoffs and rule's activation (see (5)). They are no longer the estimated accumulative payoffs of rules. Therefore it is not suitable to use them as the fitness values in the learning. The accuracy-based learning employs their changes as the fitness values. In the accuracy-based learning, the fuzzy rule in (2) can be rewritten as follows:

$$R_m: \text{IF } s_t \text{ is } F_1^m \text{ AND } \dots \text{ AND } s_N \text{ is } F_N^m \\ \text{THEN } a \text{ is } c_m^k \text{ with } q_m^k \text{ and } \Delta q_m^k \quad (8)$$

Still the calculations for  $q$  and  $\Delta q$  in (5) and (7) hold. The fitness value for a rule or the rule accuracy is an inverse measure of  $\Delta q$ . By using the fitness value calculation in [18], a predicted rule accuracy  $\kappa$  at time step  $j$  is defined as follows:

$$\kappa_j = \begin{cases} \eta e^{-(\Delta q_j - \Delta q_0)} & \Delta q_j > \Delta q_0 \\ \eta & \text{otherwise} \end{cases} \quad (9)$$

The accuracy falls off exponentially for  $\Delta q_j > \Delta q_0$ .  $\Delta q_0$  is an initial value. The predicted accuracy in (9) can be used to adjust rule's fitness value  $f_j$  using the standard Widrow-Hoff delta rule:

$$f_j = f_j + \chi(\kappa_j - f_j) \quad (10)$$

where  $\chi$  is an adjust rate of fitness values.

The niche GAs can prevent the population from the premature convergence or the genetic drift resulting from the selection operator [11]. The niching GAs maintain population diversity and promote the formation of sub-populations in the neighbourhood of local optimal solutions [8]. In rule-based systems, the way to accomplish niching is to force competing rules to share fitness. The explicit fitness sharing method divides a fitness value into the individuals within a fixed niche radius [11]. In a FCS, the fitness sharing is implicitly implemented by assigning fitness values to the activated rules based on their contributions. The fuzzy rule antecedent constitutes an evolving niche or sub-population where the fuzzy rules with the same antecedent share similar environment states. The rule consequences or actions need to compete for survival within a niche, while the rules from different niches co-operate to generate the output.

In the definition of a fuzzy rule in (8), a fuzzy rule can be defined as a sub-population and the rule actions are encoded as individuals in sub-population. If there are  $M$  rules in a FCS, there will be  $M$  sub-populations. As shown in figure 1, in each learning step, the payoff from the environment is apportioned to the rules that are activated in the previous step. The rule's fitness values are accordingly updated in the form of (10). The arrows between the payoff and sub-populations show the spatial credit assignment. There is a winner action in each sub-population and the winner actions from all sub-populations are formed a FCS. The selection of the winners in sub-population is implemented by the niching GA. The arrows between the sub-populations and FCS represent the action selection. The niche GA uses two operators to select the actions:

- Reproduce operator: individuals in each sub-population are selected as winners in terms of their fitness values. The roulette wheel selection is used.

- Mutation operator: the mutation is taken for each sub-population with a mutation probability  $p_m$ . The operator chooses an individual from sub-population randomly to replace a winner in the sub-population.

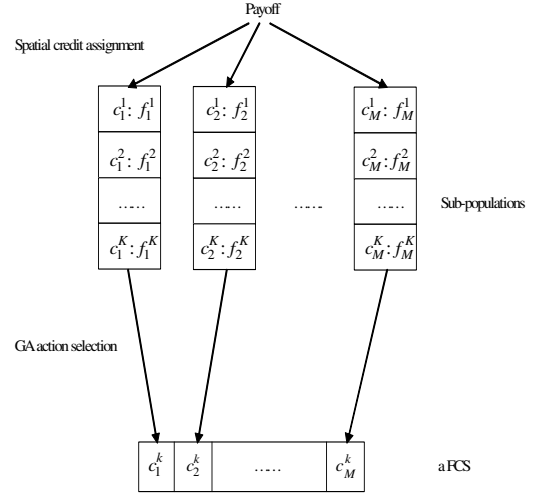


Fig. 1. Learning mechanism

#### IV. LEARNING ALGORITHM

The accuracy-based learning architecture is shown in figure 2. The environment provides current states and reinforcement payoffs to the learning system. The learning system produces actions to perform in the environment. The system includes a performance component, a reinforcement component, and a discovery component.

The performance component is a FCS that reads states from environment through physical sensors, calculates activation degrees of fuzzy rules, and generates an action. The FCS is only composed of those rules selected by the GA. The action is then executed by the learning robot. The robot moves into next state and receives evaluating payoff from the environment for its action.

The discovery component plays an action selection role. Two genetic operators are used to implement the selection. Finally, a set of rule actions is selected for the performance component.

The reinforcement component serves to assign the payoff to the individual rules that are activated by current state. For a continuous state space, it is impractical to find the next step Q values on-line. An on-line algorithm can be achieved by replacing the optimal Q value in next step ( $\max_{a_t} Q(s_t, a_t)$ ) with a predicted value  $Q(s_{t+1}, a_{t+1})$  (SARSA algorithm) [15]. The updating formula can be expressed as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (11)$$

And the change of the  $Q$  value becomes:

$$\Delta Q(s_t, a_t) = \alpha(r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (12)$$

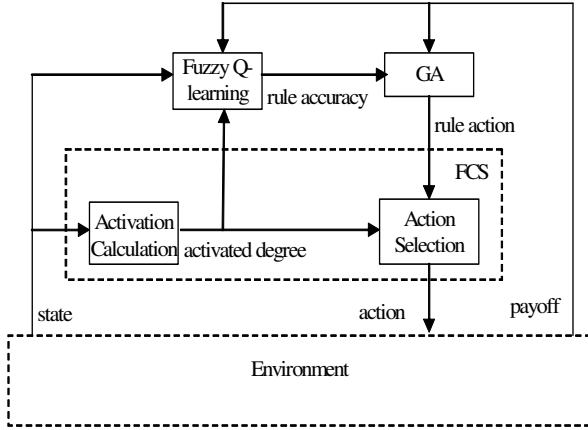


Fig. 2. Fuzzy Q-learning architecture

Actually,  $\max_{a_t} Q(s_t, a_t)$  does not provide the optimal estimation for accumulated payoffs before final convergence. In the early stage of learning, it is specifically true since the robot does not sufficiently explore its space.

The algorithm is summarised as:

1. Initialise  $q_m^k$ .
2. Read a current state  $s_t$ .
3. Select an action  $a_t$  by using an  $\varepsilon$ -greedy algorithm.
4. Calculate  $Q(s_t, a_t)$  by using (11).
5. Run a action  $a_t$ , read next state  $s_{t+1}$ , and receive a reward  $r_t$ .
6. Select an action  $a_{t+1}$ , by using the  $\varepsilon$ -greedy algorithm.
7. Calculate  $Q(s_{t+1}, a_{t+1})$  by using (11).
8. Calculate  $\Delta Q(s_t, a_t)$  by using (12).
9. Calculate  $\Delta q_m^k$  and  $f_i$  by using (5)(9)(10).
10. Update  $q_m^k$  by using (7).
11.  $a_t = a_{t+1}$ ,  $s_t = s_{t+1}$ ,  $Q(s_t, a_t) = Q(s_{t+1}, a_{t+1})$ .
12. Go to step 5 until the terminal conditions are met.

Figure 3. fuzzy Q-learning architecture

## V. EXPERIMENT RESULTS

The experiment evaluation is conducted on a *chase-ball* behaviour of a soccer playing robot. The behaviour target is to move behind a ball. Input states include the distance to a ball, the angle related to a ball, and the angle related to a goal. Three fuzzy sets are defined for each state. There are 27 rules in total for the behaviour. Six walking commands are used for the robots, including *MOVE FORWARD*, *LFFT*

*FORWARD*, *RIGHT FORWARD*, *LEFT TURN*, *RIGHT TURN*, and *STOP*.

The experimental environment consists of a football pitch used for the robot. The robot has its own vision camera to find its input states. Additionally, a global CCD colour camera is mounted on the ceiling to provide global position information to the proposed learning system as payoffs. The global monitor can provide robot localisation and ball position via wireless Internet to the robot.

Figure 4 (a) and (b) are two test positions for the *chase-ball* behaviours in simulations and real robots (the two poses are (50cm, 110cm, -30°) and (30cm, 140cm, 40°) respectively). The robot and the ball are in turn placed in the two positions during the learning process. A trial is executed until the ball is big enough, indicating that the robot is near to the ball or time is up. A learning run is defined as 10 trials. Finally, a FCS is tested to verify if the behaviour is learned. This FCS is the best one chosen from the last run.

The mutation probability in the GA is 0.1. In the simulation, 30 runs are carried out. In the real robot, 20 runs are carried out.

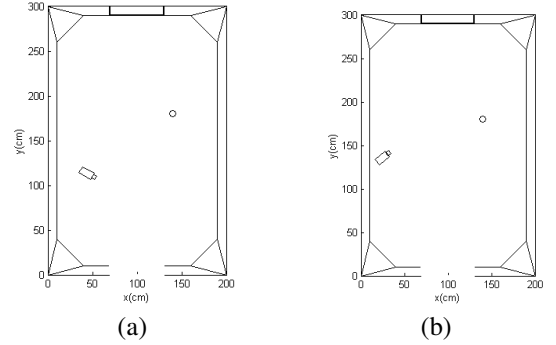


Fig. 4. Two test positions

The environment payoffs obtained from the simulations are normalised to 1 and shown in figure 5. They indicate that the payoffs gradually increased and finally converged to a high value. The dropdowns in the middle of the curve indicates the solution exploration procedures by the mutation operator.

The best FCS was picked up from the last run to do a test. Figure 6 shows the behaviour is successfully evolved. The robot can move to a ball in both positions.

In a real robot, the parameters are the same as in the simulation, but fewer runs are tested. The environment payoffs are normalised to 1 and shown in figure 7. It shows they converged to a high values. Figure 8 are two tests of the behaviour in a real robot with the best FCS in the last run. The behaviour was successfully acquired.

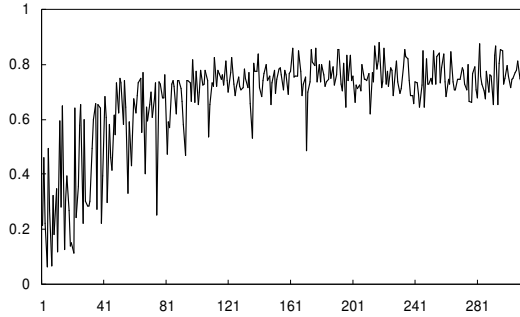


Fig. 5. Learning in a simulation

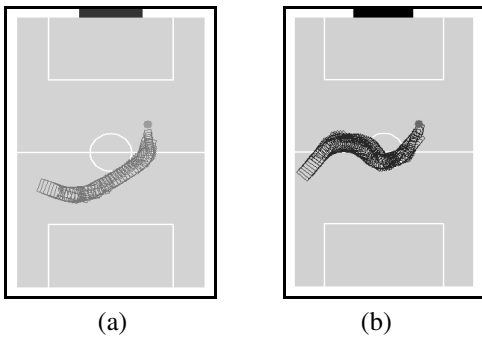


Fig. 6. Tests in a simulation

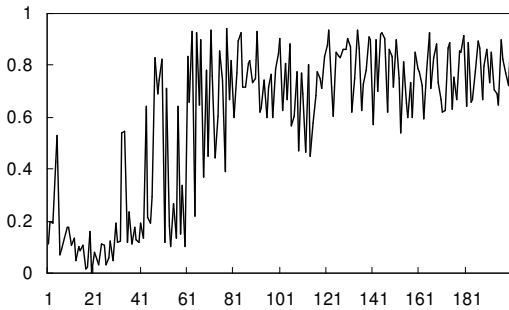


Fig. 7. Learning of the real robot

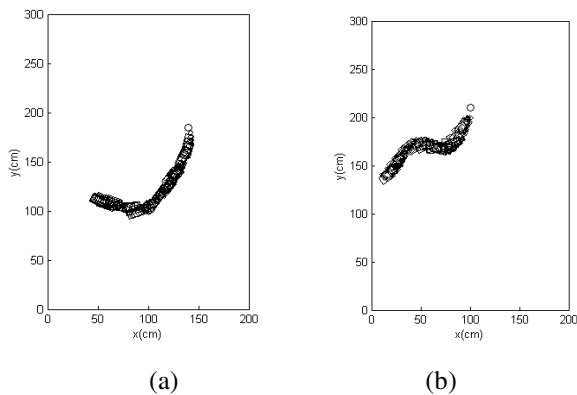


Fig. 8. Tests of the real robot

## VI. CONCLUSIONS

This paper formulates the FCS learning from two prospects. One prospect is from the Q-learning view. The learning purpose is to find a utility value for each action-state pair. Due to continuous states of robot applications, fuzzy logic was used to generalise the discrete Q-learning. A action selection mechanism was implemented by a niching GA. Another prospect is from the learning CSs view. The fuzzy Q-learning algorithm acts as a reinforcement learning component and the GA as a discovery component. Although these two prospects have originated from different research communities, they are gradually convergent to one entity since the action predication of Q-learning and the action exploration of GA have positive effects on GA and Q-learning respectively. An accuracy-based fitness is used in the discovery component. In the niching GA, the payoff sharing is implicitly conducted among the activated fuzzy rules. In learning CSs, rule deletion and covering mechanisms are important for the learning. The implementation of these two mechanisms has been listed as future work.

## ACKNOWLEDGMENT

This research is funded by the Engineering and Physical Sciences Research Council (EPSRC) under grant GR/S45508/01.

## REFERENCES

- [1] Berenji, H. R. and Khedkar, P., Learning and Tuning Fuzzy Logic Controllers through Reinforcements, *IEEE Trans. on Neural Networks*, Vol. 3, No. 5, 1992, pages 724-740.
- [2] Bonarini, A., Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier Systems, *Proceedings of Int. Conf. on Learning Classifier Systems*, Morgan Kaufmann, San Francisco, CA, 1999, pages 228-235.
- [3] Booker, L. B., Goldberg, D. E., Holland, J. H., Classifier System that Learn Their Internal Models, *Machine Learning*, Vol. 3, 1988, pages 161-192.
- [4] De Jong, K. A., Learning with Genetic Algorithm: An Overview, *Machine Learning*, VOL. 3, 1988, pages 121-138.
- [5] Dorigo, M. and Bersini, H., A Comparison of Q-Learning and Classifier Systems, *Proceedings of from Animals to Animats, the 3th International Conference on Simulation of Adaptive Behaviour (SAB94)*, Brighton, UK, August 8-12, 1994.
- [6] Glorenec, P. Y. and Jouffe, L., Fuzzy Q-Learning, *Proceeding of the 6<sup>th</sup> IEEE Int. Conf. on Fuzzy Systems*, 1997, pages 659-662.
- [7] Grefenstette, J. and Schultz, A., An Evolutionary Approach to Learning in Robots, *Machine Learning Workshop on Robot Learning*, New Brunswick, NJ, 1994.
- [8] Horn, J. and Goldberg, D. E., Natural Niching for Evolving Cooperative Classifiers, In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (Eds), *Genetic Programming*, The MIT Press, 1996, pages 553-564.
- [9] Jouffe, L., Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Trans. on SMC-Part B*, Vol. 28, No. 3, August 1998, pages 338-355.
- [10] Lin, C. T. and Jou, C. P., GA-Based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System, *IEEE Trans. on SMC-Part B*, Vol. 30, No. 2, April 2000, pages 276-289.

- [11] Mahfoud, S. W., Niching Methods for Genetic Algorithms, PhD Thesis, University of Illinois at Urbana-Champaign, 1995.
- [12] Matellan, V., Fernandez, C., and Molina, J. M., Genetic Learning of Fuzzy Reactive Controllers, *Robotics and Autonomous Systems* 25 (1998), pages 33-41.
- [13] Pipe, A. G. and Carse, B., Autonomous Acquisition of Fuzzy Rules for Mobile Robot Control: First Results From Two Evolutionary Computation Based Approaches, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)*, Morgan-Kaufman, San Francisco, CA, 2000, pages 849-856.
- [14] Rendon, V. M., Reinforcement Learning in the Fuzzy Classifier Systems, *Expert Systems and Applications*, Vol. 14, 1997, pages 237-247.
- [15] Rummery, G. A., On-Line Q-Learning Using Connectionist Systems, PhD Thesis, Cambridge University, 1994.
- [16] Smart, W. D., Making Reinforcement Learning Work on Real Robots, PhD Thesis, brown University, 2002.
- [17] Sutton, R. S., Generalisation in Reinforcement Learning: Successful Example Using Sparse Coarse Coding, *Advances in Neural Network Processing Systems*, Vol. 8, 1996, pages 1038-1044.
- [18] Wilson, S. W., Classifier Fitness Based on Accuracy, *Evolutionary Computation*, Vol. 3, No. 2, 1994.