

REINFORCEMENT LEARNING OF FUZZY LOGIC CONTROLLERS FOR QUADRUPED WALKING ROBOTS

Dongbing Gu and Huosheng Hu

*Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, UK
Email: dgu@essex.ac.uk, hhu@essex.ac.uk*

Abstract: This paper presents a fuzzy logic controller (FLC) for the implementation of some behaviour of Sony legged robots. The adaptive heuristic Critic (AHC) reinforcement learning is employed to refine the FLC. The actor part of AHC is a conventional FLC in which the parameters of input membership functions are learned by an immediate internal reinforcement signal. This internal reinforcement signal comes from a prediction of the evaluation value of a policy and the external reinforcement signal. The evaluation value of a policy is learned by temporal difference (TD) learning in the critic part that is also represented by a FLC. A genetic algorithm (GA) is employed for learning internal reinforcement of the actor part because it is more efficient in searching than other trial and error search approaches.
Copyright © 2002 IFAC

Keywords: Fuzzy logic controller, Learning algorithm, Robot control.

1. INTRODUCTION

Autonomous robots operating in an unknown and uncertain environment have to cope with changes in the environment and constraints imposed by some difficult tasks. There is an increasing tendency to build up the mapping of sensory and action pairs by Fuzzy Logic Controllers (FLC) (Beom, *et al.*, 1995), Neural Networks (NN) (Reignier, *et al.*, 1997), and Reinforcement Learning (RL) (Asada, *et al.*, 1997)(Baltes and Lin, 2000) to make robots be reactive and adaptive.

The mechanism of a FLC is that the uncertainty is represented by fuzzy sets and an action is generated co-operatively by several rules that are triggered to some degree, and produce smooth and robust control outputs. The NN and RL control schemes are based on the collection of a large number of sensory and action data pairs, which can be sampled by extensive exploration of state-action space. In many robotic applications, such as the RoboCup discussed in this paper, the training data is difficult to collect and the heuristic rules are available from our previous knowledge or experience. Therefore the FLC scheme is more suitable than NN or RL control schemes.

The problems in the design of a FLC are the setting of parameters of membership functions and the composition of fuzzy rules. They are classified into two types: structure identification and parameter adjustment (Mitra and Hayashi, 2000). The structure identification of a FLC includes the partition of its input space, the selection of antecedent and consequent variables, the determination of the number of *IF-THEN* rules, and the initial position of membership functions. The parameter adjustment determines the parameters of membership functions. Many learning approaches have been proposed to refine or modify a FLC, including NN based (Jang, 1993), RL based (Berenji and Khedkar, 1992)(Lin and Jou, 2000), and GA based (Bonarini and Khedkar, 1992)(Jung, *et al.*, 2000).

The NN-based FLC automatically determines or modifies its structure and parameters with unsupervised or supervised learning by representing a FLC in a connectionist way. The problem for the NN-based FLC is that enough data pairs have to be provided to train the networks before being used. The GA-based and RL-based FLCs are two equivalent learning schemes which need a scalar response from the environment to show the action

performance (Kaelbling and Moor, 1996). Although the response from the environment for actions is needed, it can be a scalar value that is easier to collect than the desired-output data pairs in the real robot application and can be in any form without the differentiable requirement.

The difference between the GA-based and RL-based FLCs lies in the manner of state-action space searching. The GA-based FLC is a population based approach that encodes the structure and/or parameter of each FLC into chromosomes to form an individual, and evolves individuals across generations with genetic operators to find the best one. The RL-based FLC uses statistical techniques and dynamic programming methods to evaluate the value of FLC actions in the states of the world. However, the pure GA-based FLC can not proceed to the next generation until the arrival of the external reinforcement signal (Lin and Jou, 2000). In contrast, the RL-based FLC can be employed to deal with the delayed reinforcement signal that appears in many situations.

This paper describes the design of a FLC for Sony legged robots to play soccer. The main behaviour is to move towards the ball guided by vision. There is no global visual system to provide relative position of the ball and the robot for immediate rewards. The only information provided from the environment is the ball's local visual information that may disappear in some cases because of the dynamic motion of the legged robots or the occlusion of other robots. The response in this task is a kind of delayed reward for learning. Therefore, FLC with the learning ability by adaptive heuristic critic (AHC) reinforcement learning (Barto, *et al.*, 1983) is investigated in this paper for the behaviour control.

In the rest of this paper, a handcrafted FLC is formulated for Sony legged robots to move towards the ball in section II. The architecture of the FLC with the AHC reinforcement learning is shown in section III. Section IV provides learning mechanisms for the actor and critic part. The simulation and experimental results are given in section V. Finally section VI presents conclusions and future work.

II. FUZZY LOGIC CONTROLLER

For Sony legged robots, the output action is the discrete command set, each of which can make the robot move single steps in different directions. We adopt FLC for both actor part and critic part with the same input partition by heuristic rules or experience (Gu and Hu, 2000). The learning aims to refine those handcrafted settlements to make learning work more effectively since the robot becomes increasingly aware of better control rules and capable of interacting with the environment as the learning progresses.

A reactive control scheme is employed for Sony legged robots to approach the ball in a game. There are two state variables: the orientation relative to the ball represented by θ and the distance to the ball by d , which are important for this behaviour due to the lack of global co-ordination (see figure 1). The

tracking algorithm has been implemented for the robot head to track the ball based on the colour detection (Hu and Gu, 2001). It leads θ to be expressed by the head's pan angle that can be easily read from the pan encoder. Although there is an infrared range-finder for the measurement of distance, it is difficult to use. Instead, we use the image pixels of the ball to approximate the distance (d). So, the input state vector is $S = [s_1, s_2]^T = [\theta, d]^T$. This behaviour is to control the robot to approach the ball by taking action such as *MOVE FORWARD*, *LEFT FORWARD*, *RIGHT FORWARD*, *LEFT TURN*, or *RIGHT TURN*, which are provided by low-level walking software.

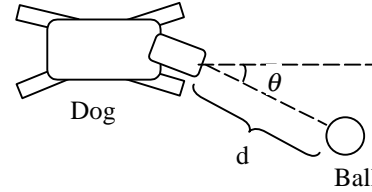


Figure 1. Approaching ball behaviour.

Both sensory information and actions are imprecise, incomplete, and vague. For example, the ball size measured in pixels does not accurately correspond to the distance from the robot to the ball, but with fuzzy concepts such as small (S), middle (M), or Large (L). The pan angle measured in degrees from a head motor encoder is accurate to some extent, but it is not the precise direction from Sony dog to the ball since the ball may not stay in the image centre. So we have fuzzy concepts, left (L), zero (Z), or right (R), for the pan angle. The output of this behaviour is one of the five one-step motion commands that can not be precisely represented in position and orientation due to non-perfect actuators or slippage on the ground.

We define F_n^j as the j th fuzzy set ($j=1\dots l_n$) and l_n the number of fuzzy sets for the input state variable s_n . A quadruple of (a, b, c, d) is used to represent the triangle or trapezoid membership function of the fuzzy set as shown in figure 2 where $b = c$ for triangle shape. The output action a is the crisp value that can be seen as a fuzzy singleton c^m ($m=1\dots M$) in a FLC. The membership degree for fuzzy set F_n^j is expressed as:

$$\mu_{F_n^j}(s_n) = \begin{cases} \max \left[0, \frac{d - s_n}{d - c} \right] & s_n > c \\ \max \left[0, \frac{s_n - a}{b - a} \right] & s_n < b \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

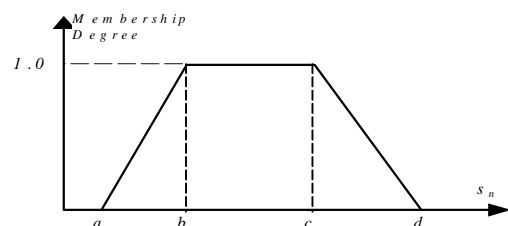


Figure 2. Membership function of a fuzzy set.

We have experiences of using heuristic methods to control the robot to approach the ball, e.g. the robot should turn left or right when the pan angle is large and the ball size is large, etc. They are represented as the linguistic fuzzy rule:

Ri: IF s_1 is $F_1^{j_1}$ **AND** s_2 is $F_2^{j_2}$, **THEN** a is c^{m_i}

There are $N(N = l_1 \cdot l_2)$ rules in total. The true value for the i th rule activated by the input state vector S_t is calculated by Mamdani's minimum fuzzy implication:

$$\alpha_{R_i}(S_t) = \min(\mu_{F_1^{j_1}}(s_1), \mu_{F_2^{j_2}}(s_2))$$

The crisp output a , stimulated by the input state S after fuzzy reasoning, is calculated by the centre of area method (COA):

$$a(S_t) = \frac{\sum_{i=1}^N \alpha_{R_i}(S_t) \cdot c^{m_i}}{\sum_{i=1}^N \alpha_{R_i}(S_t)}$$

or

$$a(S_t) = \sum_{i=1}^N p_i(S_t) \cdot c^{m_i} \quad (2)$$

The output action a will change if the robot visits different states S_t . The change manner is determined by parameters p_i when the commands c^m are fixed. Further, p_i depends on the parameters of the fuzzy membership function (a, b, c, d) when the input partitions (l_1, l_2) are set up. In the next sections, we will investigate how to let robots autonomously learn all quadruples (a, b, c, d) of input fuzzy membership functions in order to adapt the change manner to the dynamic and uncertain environment.

III. FUZZY LOGIC CONTROLLER WITH THE AHC REINFORCEMENT LEARNING

Reinforcement learning is achieved when an optimal policy is found by maximising the payoffs (or rewards) received over time, which is a good indication of the optimal action to take at each step. The payoff model normally takes the form of the expected sum of infinite horizon discounted payoffs, called the evaluation value of a policy:

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

where γ is the discount factor ($0 \leq \gamma \leq 1$) and r_t is the payoff or the external reinforcement signal received at time t . This evaluation value depends on the state S_t at time t and is represented as $V(S_t)$. The optimal policy π that consists of a series of actions (a) can be found from the following equation:

$$V^*(S_t) = \max_{\pi} E\left(\sum_{k=t}^{\infty} \gamma^{k-t} r_k\right) \quad (3)$$

The payoff model also indicates that the most desirable action is not the one that gives the highest immediate payoff. In other words, it should be credited for improvement upon the completion of a long series of actions. This is called the credit assignment problem that is especially acute when the payoff occurs infrequently (Barto, *et al.*, 1983).

A simple way to learning under this model is to wait until the goal is achieved and then assign the credit to each action taken. For example, for the behaviour of approaching the ball, after the ball size in the image is large enough or the time is long enough, the learning starts to improve the actions at each time step. However, this method will result in large sensory and action space searching, and a great deal of memory. Therefore, it is important to make maximum use of the information gathered in order to reduce the number of trials to be performed.

The evaluation value of a policy in (3) can also be written as an iteration equation according to Bellman's dynamic programming in a Markovian environment:

$$V^*(S_t) = \max_a \left(r_t + \gamma \sum_{S_{t+1}} P(S_{t+1} | S_t, a) \cdot V^*(S_{t+1}) \right) \quad (4)$$

It shows that the next state S_{t+1} is achieved after gathering the payoffs r_t from the environment by executing action a at the current state S_t with the state transition probability $P(S_{t+1} | S_t, a)$. Accordingly, the optimal policy can be found from (4) if Markovian model ($r_t, P(S_{t+1} | S_t, a)$) is known *a priori*. However, this is not the case for most applications. The learning should proceed through exploring the environment to gather more knowledge about the Markovian model and exploiting the knowledge accumulated by (4) to reduce the number of trials.

In the model free RL (Kaelbling and Moor, 1996), instead of learning the model first and then finding an optimal policy, the policy and its evaluation value are learned directly. To do so, the optimal evaluation value $V^*(S_t)$ in (4) is approximated by a prediction value $V(S_t)$ with the error:

$$\varepsilon_t = V^*(S_t) - V(S_t)$$

This error can be approximated by its predicted value according to (4):

$$\hat{\varepsilon}_t = r_t + \gamma \cdot V(S_{t+1}) - V(S_t) \quad (5)$$

where $V(S_{t+1})$ is the predicted value for $\sum_{S_{t+1}} P(S_{t+1} | S_t, a) \cdot V^*(S_{t+1})$ and represents the accumulated knowledge.

In the AHC scheme, the predicted evaluation value $V(S_t)$ of a policy is learned by TD method that is responsible for the *credit assignment problem* in the critic part, while the learning in the actor part is to select actions that lead to the convergence of the predicted evaluation value and the true optimal value. Minimising $\hat{\varepsilon}_t$ by different methods is an easy way to optimise the actor part. But this kind of greedy algorithm leads to the learning losing the capability of exploration of the sensory and action space. Therefore, the algorithms that can deal with the tradeoffs between exploration and exploitation are necessary to the learning in the actor part. Normally, some *ad hoc* algorithms are used for this purpose, such as \mathcal{E} -greedy algorithm (Jouffe, 1998), Boltzmann or Gaussian randomised strategy.

The FLC with the AHC reinforcement learning proposed in this paper, as shown in figure 3, can be regarded as a controller modified by RL from control

engineering points of view. It can also be thought of as the AHC reinforcement learning with fuzzy logic as the generalisation of actor and critic parts from machine learning points of view. Finding the optimal policy in RL corresponds to find optimal FLC in the actor part, while each action in the policy corresponds to the activated rules in FLC.

The actor part in figure 3 is a FLC expressed in (2), whose parameters are learned by an internal reinforcement signal expressed in (5). The GA is used for learning and exploration.

The FLC in the critic part has the same input states S_t as in the actor part. The output is the evaluation value of a policy $V(S_t)$ that is a scalar real value. We also have heuristic experience to build up Takagi-Sugeno fuzzy rules mapping S_t into $V(S_t)$. The same input partition and the number of the fuzzy rules are used to yield a similar expression as (2):

$$V(S_t) = \sum_{i=1}^N q_i(S_t) \cdot v^i \quad (6)$$

The difference with the actor part lies in the learning parameters of input and output. We fix the input membership functions and improve the output value through learning in the critic part. This means the prediction of the true evaluation value of a policy can be learned by improving v_i when q_i is fixed given the input states S_t in formula (6). TD learning is used for the critic part and the learning details will be described in the next section.

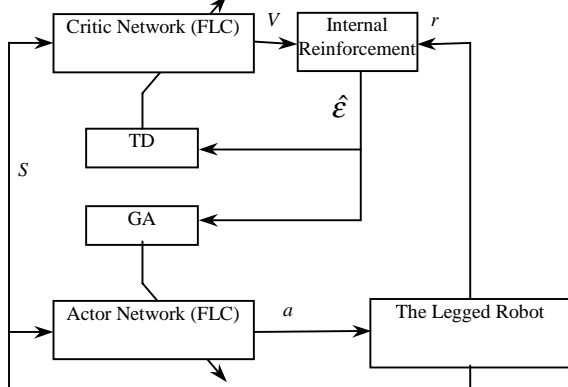


Figure 3. Learning FLC Architecture.

IV. GA BASED REINFORCEMENT LEARNING

There are two salient features for reinforcement learning compared with other learning methods. One is the reward or payoff, which can be chosen as a scalar value and need not be differentiable. This is a key point in this paper since it is impossible, or very difficult even if it is possible, to collect the corresponding stimulate-response data pairs. Another key point is the temporal credit assignment problem (Sutton, 1988) since immediate reward sometimes can not be received in a dynamic environment.

The aim of the critic part is to approximate the true evaluation value of a policy by learning. The approximated error plays the role of an immediate internal reinforcement that guides learning not only in the critic part but also in the actor part. The delayed rewards from the environment force the

critic part may take a long sequence of actions until the significant external reinforcement arrives and is assigned to each of the actions in the sequence. In the environment modelled as Markovian decision process (MDP), where the state transitions are independent of any previous environment states or actions, TD learning has a capability to allow the current TD error (or the internal reinforcement) to update all previous predictions of the evaluation value (Sutton, 1998). This is done by making use of eligibility traces that memorise the previously visited state-action pairs or the triggered rules in our case.

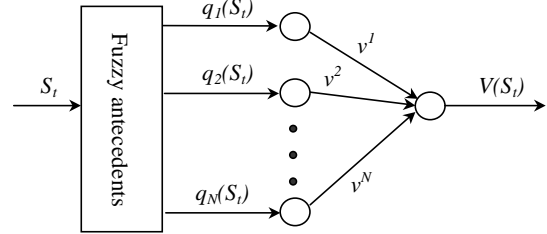


Figure 4. The critic part

The FLC in the critic part formulated in (6) can be regarded as a Proceptron shown as in figure 4. The weights v_i can be updated by the delta rule with learning rate β :

$$v^i(t+1) = v^i(t) + \beta \cdot \hat{\epsilon}_t \cdot q_i(S_t) \quad (7)$$

This is the TD(0) algorithm that looks only one step ahead when learning (Sutton, 1988). The general case is TD(λ) algorithm:

$$v^u(t+1) = v^u(t) + \beta \cdot \hat{\epsilon}_t \cdot q_u(S_t) \cdot e_t(u) \quad (8)$$

where $e_t(u)$ is the *eligibility traces* for the rule u that presents the rule triggered previously, not just the immediate previous rule. It is defined to be the accumulated sum:

$$e_t(u) = \sum_{k=1}^t (\lambda \cdot \gamma)^{t-k} \delta_{u,u_k} \cdot \delta_{u,u_k} = \begin{cases} 1, & u = u_k \\ 0, & \text{otherwise} \end{cases}$$

or updated version:

$$e_{t+1}(u) = \begin{cases} \gamma \cdot \lambda \cdot e_t(u) + 1, & u = u_t \\ \gamma \cdot \lambda \cdot e_t(u), & \text{otherwise} \end{cases} \quad (9)$$

where λ determines the trace decay rate, called *eligibility rate* and u_t is the current triggered rules. When $\lambda = 1$, it will update all the states visited previously.

The success of RL learning also depends largely on the exploration and exploitation of state-action space. Most of statistical approaches explore the space at the beginning of learning and exploit it near the end. The compromise between them can balance the generalisation and efficiency of learning. GA appears to be a better solution to this problem due to its global optimisation capability [19]. The actor part employs GA to update its input membership functions and search the space at the same time.

Conventional GAs encode a solution (in our case, it is a FLC) in the form of a discrete or binary string called a chromosome or an individual [16][20][27]. This kind of encoding results in very long strings that slow down the evolution or it is difficult to find an optimal solution if the string length is not long

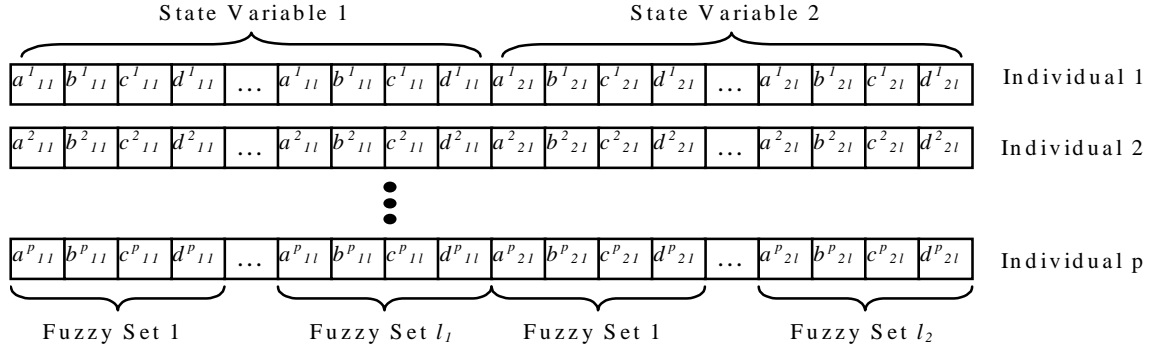


Figure 5. Encoding of a FLC.

enough. For the FLC, the quadruple (a,b,c,d) of its input membership functions are real values. It is better to manipulate them directly in the real value space instead of discrete space [15][19]. The real value encoding of a FLC for one generation with p individuals is shown in figure 5 where one individual represents one FLC.

The fitness function is defined by:

$$f(t) = \frac{1}{\hat{\epsilon}_t}$$

There are three genetic operators employed: reproduction, crossover and mutation.

- **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 p_c . One point crossover is used to exchange the genes.
- **Mutation:** The real value mutation is done by adding a certain amount of noise to new individuals to produce the offspring with the mutation probability p_m . For the i th variable in j th individual, it can be expressed as:

$$a_i^j = a_i^j + \sigma_i \cdot N(0,1)$$

where $N(0,1)$ is normal distribution. σ_i is standard deviation for the i th variable.

The *temporal credit assignment* problem is solved in the critic part by $TD(\lambda)$. Over many iterations, the evaluation value is distributed across the sequence of actions by propagating the credit. In contrast, there is no explicit evidence to show how to deal with it in the actor part by GA learning. However, some researchers show that the credit assignment in GA for an action is made implicitly, since policies that prescribe poor actions will have fewer offspring in future generations (Moriarty, *et al.*, 1999). Furthermore, with the convergence of the critic part in the process of $TD(\lambda)$ learning, the predicted evaluation value is much closer to the true value, which will provide a better immediate fitness value for GA to evolve.

The learning procedure starts from the random formation of an initial generation of a FLC in the actor part and handcraft of one FLC in the critic part. $TD(\lambda)$ learning proceeds in the critic part for each individual of the current generation in the actor part.

After completing the current generation, the GA evolves into a new generation and the learning repeats again in the critic part until the GA terminal condition is met.

V. EXPERIMENTAL RESULTS

A FLC is developed from heuristic experiences of approaching ball behaviour. Figure 6(a) (b) (c) are the membership functions for the pan angle, ball size and signal command output respectively. Figure 6(d) is the rule base for the FLC according to (2).

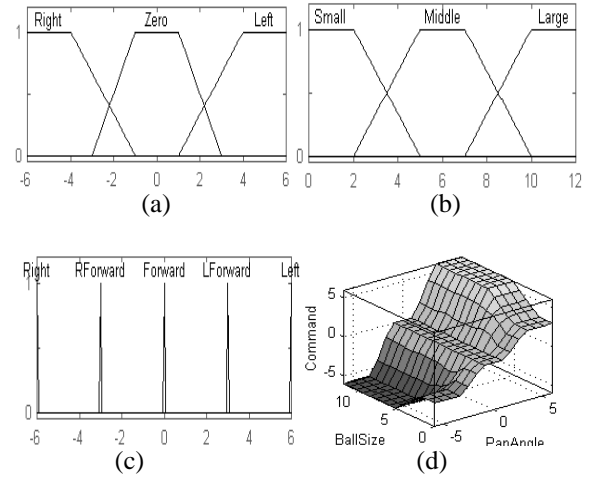


Figure 6 (a) Pan angle membership, (b) Ball size membership, (c) Output command membership, (d) The control rule surface

The payoff from the environment after executing the chosen action is defined by the ball size (d).

$$r = d / (88 \times 60)$$

The image size is 88x60. If the ball disappears in the image for a short time due to motion of the robot or occlusion of other objects, there is no immediate external reward for the action ($r=0$).

The quadruple (a,b,c,d) for each input fuzzy label should be constrained by their geometric shapes during GA learning.

$$a \leq b \leq c \leq d$$

Some constraints should also be considered for the relative position relationship between two neighbour fuzzy sets in order to keep their legibility.

$$\sum_{j=1}^{l_n} \mu_{F_n^j}(s_n) = 1$$

It implies that there are no more than two fuzzy sets being triggered by one input state and any of the input states should trigger at least one fuzzy set.

The eligibility truces are used to record past triggered rules in the critic network. The accumulating eligibility truce is implemented for the triggered rule (see figure 7) as expressed in (9) [12]. A part of memory is kept for the truce record of each rule at each sample time. The rules will be updated according to (9). However, truces are recorded in the memory only for a period of time in order to reduce the burden of time consumed in a learning process. Note that truces will be too small to keep after a period of time.

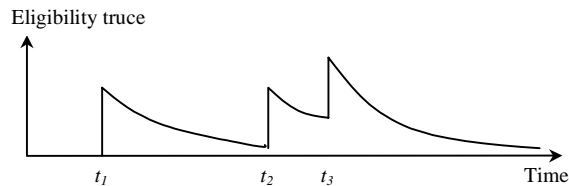


Figure 7. Accumulating Eligibility truce, t_1 , t_2 , and t_3 are the times when rules are triggered.

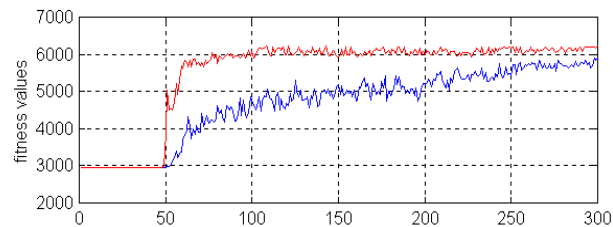


Figure 8 GA learning process

The GA learning process is shown in figure 8 after evolving 300 generations in simulation. The upper curve is the maximal fitness values in each generation. The lower curve presents the average fitness values in each generation and shows that the average fitness values converge to the maximum as the generation increases. The experiments are tested in the real environment where a Sony walking robot is commanded to approach the ball and score a goal, as shown in figure 9.

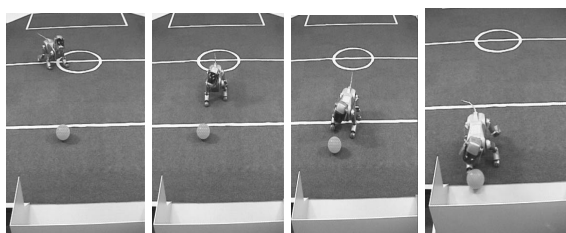


Figure 9 The results from a real robot

VI. CONCLUSIONS AND FUTURE WORK

The AHC reinforcement learning of a FLC for Sony legged robots is presented in this paper. The strategy is to make use of the heuristic experience and autonomous exploration of an environment to yield a good performance. This controller is able to evolve by scalar rewards from an environment, adapt to the uncertainty in sensing and action, and react to dynamic changes in the environment.

The experimental results show the FLC can be learned by the proposed reinforcement learning scheme. The simulation is tested first due to the consideration of the time consumption. The learning continued in real robots has proved that the simulation learning is useful and time-saving because of the reduction of the search space.

Further research will focus on the structure identification in a FLC, such as input partition, selection of antecedent and consequent variables, determination of the number of IF-THEN rules, etc. The convergence of reinforcement learning is still an open issue for discussion.

REFERENCES

- Beom, H. R., Cho, H. S. (1995). A Sensor-based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning, *IEEE Trans. on SMC*, Vol. 25, No. 3, pp464-477.
- Reignier, P., Hansen, V., and Crowley, J. L. (1997). Incremental Supervised Learning for Mobile Robot Reactive Control, *Robotics and Autonomous Systems*, 19, pages 247-257.
- Asada, M., Noda, M., and Tawaratumida, S. (1996). Purposive Behaviour Acquisition for a Real Robot by Vision Based Reinforcement Learning, *Machine Learning*, 23:279-303.
- Baltes, J., Lin, Y. (2000). Path Tracking Control of Non-holonomic Car-like Robot with Reinforcement Learning, Veloso, M., Pagello, E., Kitano, H. (Eds.), *Lecture Notes in Artificial Intelligence*, 1856, RoboCup-99: Robot Soccer World Cup III, Springer.
- Mitra, S., Hayashi, Y. (2000). Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework, *IEEE Trans. on Neural Networks*, Vol. 11, No. 3.
- Jang J. R. (1993). ANFIS: Adaptive-Neural-Based Fuzzy Inference System, *IEEE Trans. on SMC*, Vol. 23, pages 665-685.
- Berenji, H. R., Khedkar, P. (1992). Learning and Tuning Fuzzy Logic Controller through Reinforcements, *IEEE Trans. On Neural Networks*, Vol. 3, No. 5, pages 724-740.
- Lin, C. T., Jou, C. P. (2000). GA-Based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System, *IEEE Trans. On SMC-Part B*, Vol. 30, No. 2, pages 276-289.
- Bonarini, A. (1997). Evolutionary Learning of Fuzzy rules: competition and cooperation, In Pedrycz, W. (Ed.), *Fuzzy Modelling: Paradigms and Practice*, Kluwer Academic Press, Norwell, MA, 265 – 284.
- Juang, C. F., Lin, J.Y., and Lin, C. T. (2000). Genetic Reinforcement learning through Symbiotic Evolution for Fuzzy Controller Design, *IEEE Transactions on SMC-Part B*, Vol. 30, No. 2, pages 290-301.
- Kaelbling, L. P., Moor, A. W. (1996). Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research*, Vol. 4, pages 237-285.
- Barto, A. G., Sutton, R., Anderson, C. W., *Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems*, *IEEE Trans. On SMC*, Vol. 13, No. 5, Sept. 1983, pages 834-846.
- Gu, D. and Hu, H. (2000). Towards Learning and Evolving of a Team of Sony Legged Robots, *Proceedings of the Workshop on Recent Advances on Mobile Robots*, Leicester, UK.
- Hu, H. and Gu D. (2001). Reactive Behaviours and Agent Architecture for Sony Legged Robots to Play Football, *International Journal of Industrial Robot*, Vol. 28, No. 1, ISSN 0143-991X, pages 45-53.
- Jouffe, L. (1998). Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Trans. On SMC-Part B*, Vol. 28, No. 3, pages 338-355.
- Sutton, R. S. (1998) Learning to Predict by the Method of Temporal Differences, *Machine Learning*, 3(1), pages 9-14
- Moriarty, D. E., Schultz, A. C., Grefenstette, J. J. (1999). Evolutionary Algorithms for Reinforcement Learning, *Journal of Artificial Intelligent Research*, 11, pages 241-276.
- Saffiotti, A., Ruspini, E. H., and Konolige, K. (1999). Using Fuzzy Logic for Mobile Robot Control, in Zimmermann, H. J., editor, *Practical Applications of Fuzzy Technologies*, Kluwer Academic Publisher, pages 185-206.