ScienceAsia 27 (2001) : **177-183**

Genetic Reinforcement Learning with Updating Table of Q-value Function: Obstacle Avoidance Robot

T Leephakpreeda*, S Limpichotipong and C Netramai
School of Industrial and Mechanical Engineering, Sirindhorn International Institute of Technology, Thammasat University
Corresponding author, E-mail: thanan@siit.tu.ac.th

Received 31 Mar 2000
Accepted 24 Apr 2001

ABSTRACT This paper presents an alternative approach of the machine learning- the genetic reinforcement learning with the updating table of the Q-value function. The proposed method in updating table is implemented to obtain the reinforcement values of the Q-value function for given state/action pairs corresponding to any policies during exploring environment. To search optimal policies, the fitness of a set of policies for genetic algorithm is defined in terms of the value of the Q-value function. The genetic algorithm and the reinforcement learning are then applied in conjunction to optimize the final control system performance. The effectiveness of the proposed methodology is demonstrated on a real application of the obstacle avoidance robot.

KEYWORDS: Genetic reinforcement learning, Q-value function, Obstacle avoidance robot.

INTRODUCTION

Recently, many researchers in the area of machine intelligence have attempted to develop strategies for which a computer has the ability to learn solving sophisticated problems from its experience of performing a control task. [1] Among practically uses is reinforcement learning, which is a computational approach to learn a mapping from states (situations) to actions by trial-and-error interactions with a complex and uncertain environment. In this learning strategy, the concept of reinforcement function is defined as the performance measurement in the form of a scalar value. [2] In other words, the reinforcement function is the value function mapping state/action pairs to the reinforcements after performing actions in given states according to the given policy. It is called the Q-value function. [3] However, the Q-value function is usually represented by using the universal model of the neural network. As widely known, this mathematical model requires computing derivatives for learning rule, ie, gradient descent method in backpropagation training which may not be suitable to be implemented in real time due to the burden of computation time. Therefore, genetic reinforcement learning with the updating table of Q-value function is proposed here to eliminate such difficulty.

GENETIC REINFORCEMENT LEARNING

In high-level control task, a control system is required to interact with changeable environment in which it is performed. Therefore, a strategy by which the controller is able to decide on the control input by itself required to perform the specified functions while interacting with a dynamic environment has been of interest in intelligent control research. Actually, learning of a machine to interact in real-time with complex and uncertain environments has direct roots in the learning of animals in nature. It is based on the concept that the tendency of actions followed by a satisfactory or an improved affair is reinforced. This form of learning is called Reinforced Learning (RL). Fig 1 shows the block diagram of the reinforcement learning. In RL, the controller is considered an agent or a learner, which not only takes an action a at each state x of the

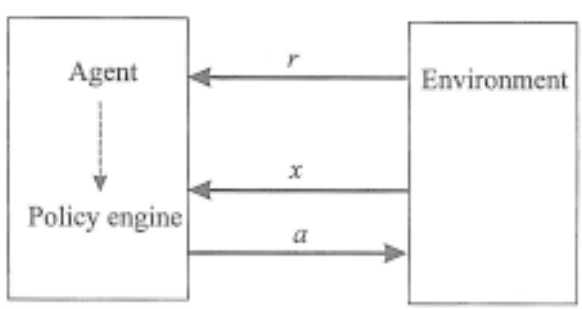

Fig 1. Block diagram of reinforcement learning.

environment but also receives a reward or a reinforcement $r(x, a)$ value from environment as the measure of its performance. For an example, a value of reward parameter r is assigned to be 1 if the success signal is received; the reward value is assigned to be -1 if the failure signal is received and otherwise 0. The policy engine determines which action should be performed in each state. In other words, the policy engine is a function of mapping from states to actions. The dashed-line arrow presents a learning process of the adaptation in the policy engine according to the reinforcement value r. In our proposed methodology, this reinforcement learning is to use a value function of the reinforcement in constructing an optimal policy, which is expected to be the most suitable for interacting with the controller's environment. The reinforcement learning algorithm can be considered as follows.

Given a policy π, which determines action that should be performed in each state, the Q-value function of the state/action pair is defined as the sum of the reinforcement value received when starting in that state and following some fixed policy to a terminal state.

$$Q_\pi(t) = Q_\pi(x(t), a(t)) \quad (1)$$

where t is the index of a time sequence.

Hence, the optimal policy is the mapping from states to actions that maximizes the Q-value function.

$$\pi^* = \arg\max_{\pi \in \Pi} \{Q_\pi(x(t), a(t))\} \quad (2)$$

where Π is the set of all admissible policies.

The corresponding optimal Q-value function $Q_{\pi^*}(x(t), a(t))$ for the optimal policy π^* is the sum of the reinforcements when starting from its state and performing optimal actions until a terminal state is reached.

By the definition of the Q-value function, a relationship between the Q-value function of two successive states, ie, $x(t)$ and $x(t+1)$, can be expressed as follows.

$$Q_\pi(x(t), a(t)) = r(x(t), a(t)) + Q_\pi(x(t+1), a(t+1)) \quad (3)$$

where $r(x, a)$ is the reinforcement value.

In general, the Q-value function of any states is unknown before learning process is complete. Hence, let's define the approximation of the Q-value function of state/action pairs to be $\hat{Q}_\pi(x(t), a(t))$. This means that the true Q-value function is equal to summation of the approximation of the Q-value function and a residual error $e(x(t), a(t))$.

$$Q_\pi(x(t), a(t)) = \hat{Q}_\pi(x(t), a(t)) + e(x(t), a(t)) \quad (4)$$

The expression in Eq (4) can also be true for the optimal policy. The relation can be written as:

$$Q_{\pi^*}(x(t), a(t)) = \hat{Q}_{\pi^*}(x(t), a(t)) + e(x(t), a(t)) \quad (5)$$

or, by using definition of the Q-value function in Eq (3), the residual error can be defined by:

$$e(x(t), a(t)) = \max_{\pi \in \Pi} \{r(x(t), a(t)) + Q_\pi(x(t+1), a(t+1))\} - \hat{Q}_{\pi^*}(x(t), a(t)) \quad (6)$$

To obtain the optimal Q-value function, the minimization-maximization problem is established as follows.

$$\text{minimize} \left\{ \max_{\pi \in \Pi} \{r(x(t), a(t)) + Q_\pi(x(t+1), a(t+1))\} - \hat{Q}_{\pi^*}(x(t), a(t)) \right\} \quad (7)$$

With Eq (7), it can be interpreted that learning is accomplished when each update of the approximation of the optimal Q-value function reduces the value of the residual error to zero. At the same time, the optimal policy can be obtained when the maximum values of the Q-value function at each state/action pairs are found. It should be noted that without searching the optimal policy (do not maximize), as the residual error approaches zero, the approximation of the Q-value function will converge to the true Q-value function. The true Q-value function obtained corresponds the performance of one policy among other policies in a set of admirable policies. Hence, if the maximization in Eq (7) can also be accomplished, the optimal policy is then obtained. The difficult issue of the problem statement in Eq (7) is that the minimization and maximization must be achieved simultaneously in order to obtain the optimal policy. To solve the problem, the updating table as discussed later on is proposed in order to provide the values of the Q-

value function in a way that the condition on $e(t)=0$ is always satisfied. With this result, the problem in Eq (7) can be reduced to Eq (8).

$$\underset{\pi\in\Pi}{\text{maximize}}\ \left\{r(x(t),a(t))+Q_{\pi}(x(t+1),a(t+1))\right\} \quad (8)$$

In this study, the principle of genetic algorithm [4] is used to search the optimal policy π^{*} on the set of the policies by stochastically creating a new policy Π and exploring the state-action space. The Q-value function for given state/action pairs are computed to measure the fitness of each policies. During iteration, the "best" one that has the highest fitness will be chosen as offspring to generate the new generation of the policies. This procedure continues during the exploration of the agent. The arrangement of the genetic reinforcement learning can be illustrated by Fig 2.

Updating Table Method

As mentioned in the previous section, the values of Q-value function are used as the fitness indexes of the policies for the optimization with the genetic algorithm. The updating table method is thus proposed to provide the values of the Q-value function of the state/action pairs corresponding to a given policy in simple way. Referring to Eq (3), the successive values of Q-value function can be obtained by calculating the present value of Q-value function backward to the values of Q-value function of the transition state/action pairs and finally to the one of the initial state/action pair. For an example, suppose that the agent performs the actions according to the policy π from the initial state $x(0)$ until the final state $x(N)$. The corresponding reinforcement values $r(x(t),a(t))$ for each state/action pairs have been credited from environment to the agent. For a given value of $Q_{\pi}(x(N),a(N))$, all the values of $Q_{\pi}(x(t),a(t))$ for $t = N - 1, ..., 2, 1, 0$ can be determined by Eq (9).

$$\begin{aligned}
Q_{\pi}(x(N-1),a(N-1)) &= r(x(N-1),\\
&\quad a(N-1))+Q_{\pi}(x(N),a(N))\\
&\vdots\\
Q_{\pi}(x(1),a(1)) &= r(x(1),a(1))+Q_{\pi}(x(2),a(2))\\
Q_{\pi}(x(0),a(0)) &= r(x(0),a(0))+Q_{\pi}(x(1),a(1))
\end{aligned} \quad (9)$$

According to the concept above, the architecture of the updating table can be illustrated as Fig 3. The rows and columns of the updating table are classified by the initial state and the transition state respectively. The dimension of the updating table is defined by the number of the possible states in environ-ment. Let's assume that there are all n different types of the states, which the agent can observe from environment.

$$x=\left\{x_1, x_2, ..., x_n\right\} \quad (10)$$

Additionally, the agent has the choices of actions.

$$a=\left\{a_1, a_2, ..., a_m\right\} \quad (11)$$

After the agent performs series of actions with a given initial state, all the values of $Q_{\pi}(x(t),a(t))$ for $t = 0, 1, 2, ..., N$ determined from Eq (9) are stored at the same corresponding row. Each of the values of $Q_{\pi}(x(t),a(t))$ and the corresponding choices of the actions for $t = 0, 1, 2, ..., N$ are filled up to their columns according to the route of the transition states before the final state is reached. Therefore, if the agent is perturbed and the deviation from the final state occurs to another state. That state is regarded as the initial state. The registration of the values $Q_{\pi}(x(t),a(t))$ of and the choices of the actions are handled as mentioned above after the final state is found again.

To get more concrete understanding, an example is given as the case that the three possible states can be monitored and the two actions can be made by

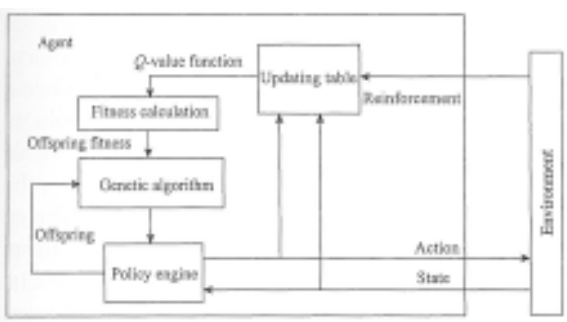

Fig 2. Arrangement of genetic reinforcement learning.

Initial state \ Transition state	x_1	x_2	...	x_n
x_1				
x_2				
⋮				
x_n				

Fig 3. Architecture of updating table.

the agent. The updating table can be constructed as shown in Fig 4. The final state, which is desired to reach, is the state x_1. Before starting to explore the environment, the updating table is blank. The policy is first defined by randomly mapping the state to the action until the agent finds the final state, that is the state x_1. At each time step, if the present state is not the final state x_1, the constant reinforcement value is given by -1. This penalty of reinforcement undergoes until the final state x_1 is obtained. Now, let's consider two cases with two initial states x_2 and x_3.

$$x_2(0) \overset{-1}{\Rightarrow} x_1(1)$$

$$x_3(0) \overset{-1}{\Rightarrow} x_2(1) \overset{-1}{\Rightarrow} x_1(0)$$

In case of the initial state x_2, the agent can reach the final state x_1 with one action a_1 while the agent starts exploring environment at the state x_3 with the action a_1, passes through the state x_2 with the action a_2, and then reaches the final state x_1. Corresponding to the policy given in the updating table, all the values of $Q_\pi(x(t), a(t))$ can be calculated backward by Eq (9) and $Q_\pi(x_1) = 0$ as shown in Fig. 4. In this case, it should be noted that the maximum value of $Q_{\pi^*}(x(0), a(0))$ indicates the shortest path from that initial state to the final state.

Procedures in Genetic Reinforcement Learning

The following steps of the genetic reinforcement learning proceed during the exploration of the agent.

Performing phase

Step 1.

(1.1) Check whether the policies are available at the initial state where the agent is.

Initial state \ Transition state	x_1	x_2	x_3
x_1	0		
x_2	0	-1 a_1	
x_n	0	-1 a_2	-2 a_1

Fig 4. Values of Q value-function with three states and two actions.

- If not, go to Step 2.
- If the policies are available, the agent performs according to them.

(1.2) Check whether the value is greater than other values in the same column of the updating table.

- If not (the policies are not optimal), use them as a parent for generating the policies for new learning. Go to Step 2.
- If its value is greater than the others, go to (1.3).

(1.3) Check whether that the sequence of state is similar to one in the updating table in the manner of state by state.

- If not (the environment may be changed), go to Step 2.
- If the sequence of state is similar, the agent continues performing until the final state is found.

Learning phase

Step 2. Check whether the parents in the genetic algorithm are available for generating the policy.

- If not, initiate the policy randomly.
- If the parents are available, use them to generate the offspring for the policies. Go to Step 3.

Step 3. Continue on learning according to the generated policies in Step2 and record the values of $Q_\pi(x(t), a(t))$ and the actions for new policies in the updating table accordingly when the final state is reached.

During the exploration of the agent, check whether the limit cycle occurs or the transition state is repeated.

- If not, go to Step 1.
- If the limit cycle occurs, go to Step 1 by setting the current transition state to be the initial state.

Results and Discussions

The viability of the proposed method is demonstrated by solving the obstacle avoidance problem of the mobile robot. The robot has to produce a "best" sequence of control actions or set of policies by itself in order to move forward when possible and avoid obstacles. As illustrated in Fig 5 (a), the robot with 5-inch diameter was independently driven by two DC motors. Six infrared proximity sensors were placed around the front of robot to perceive its environment. The location of the sensors and two DC motors is shown as Fig 5(b). The micro-

controller of ATMEL-AVR90S8535 was used to handle the input-output signals. I1, I2,..., I6 and M1 as well as M2 represent the infrared sensors and the DC motor. The two successive infrared sensors were grouped to provide a wider range of the detection. The possible eight states (2^3=8) can be defined in combination of S1, S2 and S3 in Table 1. There are five specified actions of the robot, that is, move forward a_1, rotate in counter-clockwise direction a_2, rotate in clockwise direction a_3, turn left with the rotation of the left wheel backward a_4 and turn right with the rotation of the right wheel backward a_5. It can be expected that the robot will try to make rotations and turns when the obstacle is founded. After the state x_8 is reached or the obstacle is not found, the robot will move forward with action a_1. Fig 6 shows the obstacle environment of the wall arrangement in learning. Its dimension of the square field is 1.2x1.2 m^2. Based on eight states in Table 1, the updating table has the eight rows and eight columns. To avoid the unavailable policies at the beginning, the values of Q-value function and the policies for given initial states are first filled up randomly. For an example of one experiment, the initial updating table is presented as Fig 7. For a given initial state, the fitness of a set of the policies in the same row is defined with the value of Q-value function of that initial state. This definition is based on the hypothesis that if each fitness in each initial state is maximized, the fitness of overall policies is also maximized. The performance index is defined as the summation of fitness values for all initial states. Intuitively, the performance index of the fitness-value summation from the genetic reinforcement can be used to quantify the capacity of the robot in avoiding the obstacles. This means that the more the performance index increases, the less the number of the penalties due to the unsuitable turns for the obstacle avoidance. Referring to Fig 7, the initial value of the performance index is equal to -49. Next, the proposed procedures in genetic reinforcement learning is implemented during the exploration of the robot. During first 100 seconds, the performance index against time is plotted in Fig 8. The performance

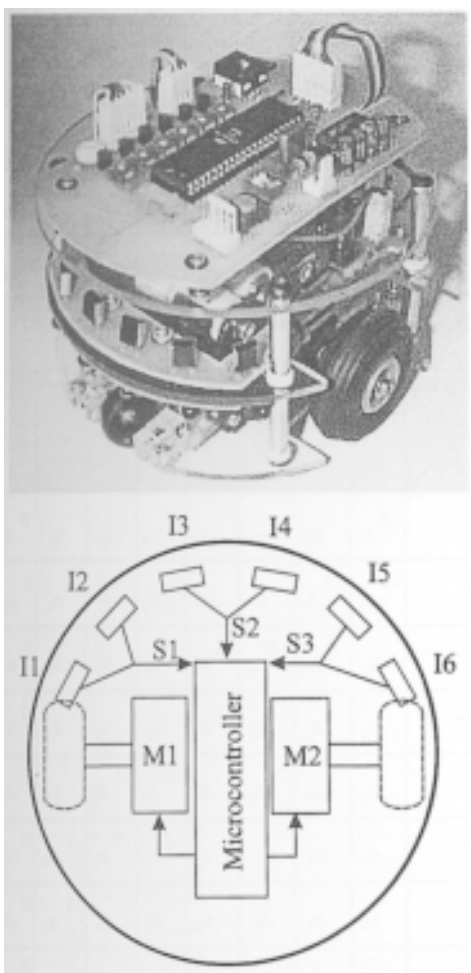

Fig 5. Diagram of mobile robot: (a) miniature mobile robot and (b) location of sensors and DC motors.

Table 1. Definition of observed states.

States	S1 (I1+I2)	S2 (I3+I4)	S3 (I5+I6)
x_1	1	1	1
x_2	1	1	0
x_3	1	0	1
x_4	1	0	0
x_5	0	1	1
x_6	0	1	0
x_7	0	0	1
x_8	0	0	0

1-found obstacle; 0- not found obstacle.

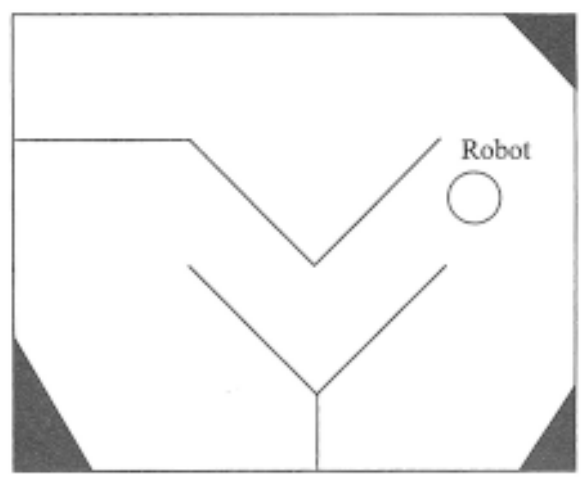

Fig 6. Obstacle environment of wall arrangement in top view.

index increases during the exploration of the robot. This means that the improved policies for environment in Fig 6 can be obtained. After 60 seconds, the performance index converges to the value of -17. During that time, the updating table at time of 30 seconds is picked as shown in Fig 9. Only the initial states x_5, x_6 and x_7 are found and are learnt. The corresponding performance index is -35. It is observed from experiment that after the 15-minute exploring of the robot is taken for finding all initial states, the value of the performance index is equal to -14. The optimal policies are shown as Fig 10. It should be noted that learning may take longer time to reach the optimal policies in the cases of more complicated environment. Furthermore, it can be interpreted from the experiment results in the case that the state x_6 is the initital state as follows. When the robot starts at the initial state x_6, the robot will choose to turn to the right. The state x_5 is then found. Rotation counter-clockwise is performed to avoid the obstacle to the right. Next, the state x_7 is founded. The robot makes a left turn. Finallly, the final state x_8 is reached. However, at initial state x_6, another feasible action is clockwise rotation which can be

Initial state \ Transition state	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	-7 a_5	-6 a_5	-5 a_2	-4 a_5	-3 a_3	-2 a_3	-1 a_3	0 a_1
x_2	-6 a_2	-7 a_2	-5 a_3	-4 a_2	-3 a_3	-2 a_2	-1 a_2	0 a_1
x_3	-5 a_3	-6 a_2	-7 a_4	-4 a_2	-3 a_4	-2 a_5	-1 a_4	0 a_1
x_4	-4 a_5	-5 a_5	-6 a_4	-7 a_4	-3 a_4	-2 a_4	-1 a_3	0 a_1
x_5	-3 a_2	-4 a_2	-5 a_4	-6 a_5	-7 a_2	-2 a_4	-1 a_5	0 a_1
x_6	-2 a_3	-3 a_2	-4 a_4	-5 a_2	-6 a_4	-7 a_2	-1 a_4	0 a_1
x_7	-1 a_5	-2 a_2	-3 a_2	-4 a_4	-5 a_5	-6 a_3	-7 a_5	0 a_1
x_8								0 a_1

Fig 7. Initial updating table of values of Q-value function with eight states and five actions.

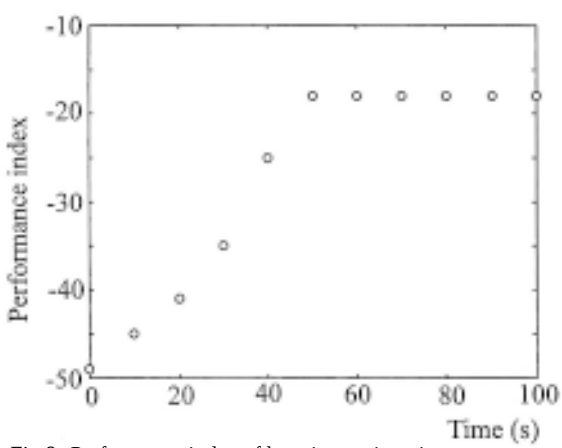

Fig 8. Performance index of learning against time.

Initial state \ Transition state	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	-7 a_5	-6 a_5	-5 a_2	-4 a_5	-3 a_3	-2 a_3	-1 a_3	0 a_1
x_2	-6 a_2	-7 a_2	-5 a_3	-4 a_2	-3 a_3	-2 a_2	-1 a_2	0 a_1
x_3	-5 a_3	-6 a_2	-7 a_4	-4 a_2	-3 a_4	-2 a_5	-1 a_4	0 a_1
x_4	-4 a_5	-5 a_5	-6 a_4	-7 a_4	-3 a_4	-2 a_4	-1 a_3	0 a_1
x_5	a_2	a_5	a_4	a_2	-2 a_4	a_3	-1 a_3	0 a_1
x_6	-3 a_2	a_5	a_4	a_2	-2 a_4	-4 a_4	-1 a_5	0 a_1
x_7	a_5	a_4	a_3	a_2	a_3	a_2	-1 a_4	0 a_1
x_8								0 a_1

Fig 9. Updating table with policies at 30 seconds.

Initial state \ Transition state	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	-2 a_4	a_4	a_3	a_2	-1 a_3	a_4	a_3	0 a_1
x_2	-2 a_2	-3 a_4	a_2	a_3	-1 a_3	a_3	a_5	0 a_1
x_3	a_2	a_2	-2 a_2	-1 a_3	a_4	a_2	a_3	0 a_1
x_4	a_5	a_5	a_5	-1 a_3	a_4	a_2	a_3	0 a_1
x_5	a_3	a_5	a_2	a_2	-2 a_2	-1 a_5	a_5	0 a_1
x_6	a_5	a_4	a_3	a_5	-2 a_2	-3 a_5	-1 a_4	0 a_1
x_7	a_2	a_3	a_2	a_5	a_4	a_5	-1 a_2	0 a_1
x_8								0 a_1

Fig 10. Updating table with optimal policies at 900 second.

performed to avoid the obstacle. If it is such case, the sequence of action and states will be changed accordingly. Also, it is found that the robot can always avoid the wall obstacles during longer experiment run. This control scheme is thus quite robust to the variations in environment such as the friction and slipping of the wheel.

Conclusion

In this paper, the genetic reinforcement learning with the updating table of the Q-value function has been described. The genetic algorithm provides a systematic way of creating new policies for efficiently and effectively exploring the state/action space, which is required in reinforcement learning scheme. Without the computation burden, the updating table is exploited to yield the value of the fitness of the policies in such a way that the optimal policies are eventually obtained. According to the experimental results, it can be shown that the proposed methodology can be effectively used to solve the obstacle avoidance problem in real-time implementation. The robot learns and improves the policies for better performance in driving itself from an arbitrary state to a specified final state within a finite time.

References

1. Vepa R (1993) A Review of Techniques for Machine Learning of Real-time Control Strategies, Intell Syst Eng 2, 77-90.
2. Mitchell TM (1997) Machine Learning, The McGraw-Hill Publishing Companies, Inc, USA.
3. Watkins JCH and Dayan P (1992) Technical Note: -learning, *Mach Learn* 8, 279-292.
4. Dawid H (1997) Adaptive Learning by Genetic Algorithms, The Springer-Verlag Berlin Heidelberg Publishing Company, New York.